Práctica 1

Técnicas de Búsqueda Local y Algoritmos Greedy

Problema del Agrupamiento con Restricciones (PAR)

Metaheurística GII 19-20

Grupo MH1 - Miércoles 17:30h

Cumbreras Torrente, Paula

49087324-B

paulacumbreras@correo.ugr.es

ugr | Universidad de Granada

“Un algoritmo debe ser visto para ser creído”
-Donald Ervin Knuth

Índice de contenidos

Descripción del problema

El problema del agrupamiento con restricciones (PAR) consiste en un problema de agrupamiento o clustering al que se le ha añadido un conjunto de restricciones. El clustering es una técnica de aprendizaje no supervisada cuyo objetivo es agrupar los elementos que tengan alguna similitud entre ellos de un conjunto de datos en varios clusters, simplificando así la información de dichos datos.

Este problema suele ser aplicado en aplicaciones de minería de datos, aplicaciones Web, marketing, diagnóstico médico, análisis de ADN, biología computacional, análisis de plantas y animales, farmacología…

El problema elegido consiste en una generalización del ya mencionado. PAR es una técnica de aprendizaje semi-supervisada. Las restricciones expresan un requerimiento del usuario y describen propiedades que han de contener los grupos de elementos. El conjunto de restricciones que se aplicarán a nuestro problema pueden ser clasificadas en:

- Restricciones fuertes (hard): todas las restricciones de esta clase han de ser cumplidas para que una solución sea factible. Las restricciones fuertes tratadas en nuestro problema serán:
    - Todos los clusters deben contener al menos una instancia:
    - Cada instancia debe pertenecer a un solo cluster:
    - La unión de los clusters debe ser el conjunto de datos X:
- Restricciones débiles (soft): se busca que el número de restricciones violadas de esta clase sea minimizado. Las restricciones débiles a aplicar serán restricciones de instancia:
    - Must-Link: dos instancias dadas han de ser asignadas al mismo cluster.
    - Can't-Link: dos instancias dadas no podrán agruparse en el mismo cluster.

Se nos han proporcionado 3 conjuntos de datos para la realización de del problema:

- Iris: 150 datos con 4 propiedades a agrupar en 3 clases.
- Ecoli: 336 datos con 7 propiedades a agrupar en 8 clases.
- Rand: 150 datos con 2 propiedades a agrupar en 3 clases.

Para cada conjunto de datos se estudiarán 2 instancias distintas de PAR a partir de los conjuntos de restricciones, también proporcionados, los cuales corresponderán al 10% y 20% del total de restricciones posibles.

Para esta práctica se estudiarán los algoritmos Greedy y Búsqueda Local. Se realizará una comparación de ambos en función de la evaluación de los resultados obtenidos y del tiempo obtenido de la ejecución de los mismos con diferentes semillas de generación de números aleatorios.

Descripción de la aplicación de los algoritmos empleados

El esquema de estructuras de datos empleado para el problema es el siguiente:

- Los elementos se almacenan en Nodos, estructuras formadas por un entero sin signo cluster donde se registrará el cluster al que pertenecen (por defecto -1), y un vector de double datos donde se almacenan las distintas propiedades del elemento.
- El conjunto de todos los Nodos se almacena en un vector de Nodos X en el orden de lectura.
- Los Cluster quedan representados como una estructura compuesta por un vector de double donde se almacenará su centroide y por un vector de enteros sin signos donde se almacenará el índice en X de los distintos Nodos que lo compongan.
- El conjunto de todos los Clusters se almacena en un vector de Cluster C.
- Las restricciones se almacenan en una matriz (vector de vector) de enteros M.

Una vez se haya ejecutado el algoritmo seleccionado, aparecerán por pantalla una serie de evaluaciones de la partición obtenida:

- Tasa_C: Desviación general de la partición.

Desviación general

Para todos los clusters
  calcular distancia media intra-cluster
  añadir el valor obtenido al valor de una variable desviación
Devolver desviación entre el número de clusters
{Almacenar distancias medias en un vector}

Distancia media intra-cluster

Para cada elemento i del cluster
  para cada propiedad j del elemento
  añadir en variable[j] la distancia entre el dato j del elemento i y el dato j del centroide del cluster
Para cada propiedad del vector variable
  añadir su valor a una variable distancia
Devolver raíz cuadrada de distancia entre el número de elementos del cluster

- Tasa_Inf: Infactibilidad o número de violaciones de restricciones

Infactibilidad

Para cada fila de M
  para cada columna de M
  si M[i][j] es 1 y el cluster del elemento i y el del elemento j son distintos
  incrementar infactibilidad en 1
  si M[i][j] es -1 y el cluster del elemento i y el del elemento j son el mismo
  incrementar infactibilidad en 1
Devolver infactibilidad
{Almacenar número total de restricciones en una variable}

- Agregado: Función objetivo o función a minimizar

Función objetivo
Calcular infactibilidad {número total de restricciones} Calcular desviación {vector de distancias medias} Para cada elemento del vector de distancias   Buscar la mayor distancia Lambda es la distancia máxima entre el número total de restricciones Devolver desviación más infactibilidad por lambda

- Tiempo: Calculado desde el inicio del algoritmo hasta el final del mismo, excluyendo la lectura de ficheros e inicialización de datos. Devuelto en nanosegundos.

Como parámetro de entrada habrá que indicar si se desea que se muestre por pantalla el resultado de la partición C. En caso afirmativo se mostrarán los índices en X de los elementos almacenados en cada cluster; un ejemplo de salida sería:

```
Semilla = 8

Tasa_C|Tasa_i|Agreg|Tiempo - repetición 1
0.106822  0  0.106822  457676653

Cluster 0:
{ 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124
125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148
149 }
Cluster 1:
{ 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
38 39 40 41 42 43 44 45 46 47 48 49 0 2 }
Cluster 2:
{ 51 52 53 54 55 56 57 58 59 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82
83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 50 60 }
```

A parte de los ya mencionados, otros operadores que han sido necesarios para la implementación de los algoritmos son:

- Cálculo de centroides: Asigna a cada cluster el vector promedio de las instancias de X que los componen

Calculo centroides

Para cada cluster de la partición
  para cada elemento i del cluster
  para cada propiedad j del elemento
  añadir a variable [i] la propiedad j del elemento i
  para cada propiedad i del vector centroide
  asignar a variable [i] el valor que tenía dividido entre el número de elementos del cluster
  asignar el vector variable al centroide del cluster

Pseudocódigo del método de búsqueda

El método de búsqueda de esta práctica es la Búsqueda Local, un proceso iterativo que parte de una solución aleatoria y la mejora realizando modificaciones locales respecto a la función objetivo.

```
Generación de solución aleatoria
Para cada elemento i de X
  asignar un cluster aleatorio a cluster
  asignar a dicho cluster la posición i
Calcular centroides
Función objetivo

Inicio iteraciones
Mientras haya cambios y contador < 100000
  cambios = false
  para cada elemento i de X
    calcular centroides
    función objetivo

Modificación local - primero el mejor
      para cada cluster j mientras no se encuentre una mejora
        si cluster no está vacío
          calcular nuevo cluster
          si probar vecino
            eliminar el elemento i del cluster al que pertenecía
            asignar el elemento i al nuevo cluster
            modificar el valor de cluster del elemento i
            mejora encontrada y cambios=true
  aumentar contador

Fin iteraciones
Calcular centroides
```

Probar vecino: Indica si la función objetivo al cambiar un elemento a un determinado cluster sería reducida

Probar vecino
Eliminar el elemento del cluster al que pertenecía Asignar el elemento al nuevo cluster Cálculo de centroides Si función objetivo es menor que la función objetivo antes devolver true Si no devolver false

Pseudocódigo del algoritmo de comparación

El algoritmo de comparación elegido es un algoritmo Greedy COPKM. Los algoritmos greedy o voraces son aquellos que, para resolver el problema, eligen la opción óptima local en cada paso tratando así de llegar a una solución general óptima.

```
Inicialización

Infactibilidad
Calcular centroides aleatorios

Inicio iteraciones
Mientras haya cambios
  para cada elemento de X

Comparación de restricciones
    para cada cluster j de C
      si probar cluster es menor que máximo auxiliar
        vaciar lista de empates
        modificar máximo auxiliar
        añadir j a la lista de empates
      si son iguales añadir j a la lista de empates

Modificación local - elección del óptimo
      si la lista de empates tiene más de 1 cluster
        si algún cluster k de la lista está vacío, nuevo cluster = k
        si no nuevo cluster = desempatar por distancia
      si no nuevo cluster = el cluster de la lista
      si no es la primera iteración eliminar el elemento del cluster al que pertenecía
      asignar el elemento a nuevo cluster
      modificar cluster del elemento
  si infactibilidad es igual a infactibilidad antes, diferencia = false
  aumentar contador

Fin iteraciones
Calcular centroides
```

Las funciones necesarias para la implementación de este algoritmo son:

- Desempatar por desviación: Dada una lista de Cluster, devuelve aquel cuyo centroide sea menos lejano a un Nodo dado

Desempatar por distancia

Para cada elemento de la lista
  calcular distancia a elemento
  buscar distancia mínima
Devolver cluster con distancia mínima

Calcular distancia a elemento

Para cada propiedad j del elemento
  añadir en variable[j] la distancia entre el dato j del elemento y el dato j del centroide del cluster
Para cada propiedad del vector variable
  añadir su valor a una variable distancia
Devolver raíz cuadrada de distancia

- Probar cluster: Devuelve la infactibilidad que se obtendría al asignar un elemento a un determinado cluster

Probar cluster
Si no es la primera iteración   eliminar el elemento del cluster al que pertenecía Asignar el elemento al cluster Devolver infactibilidad

- Cálculo de centroides aleatorios: Asigna a cada cluster un vector de double generado aleatoriamente

Cálculo centroides aleatorios

Para cada cluster de la partición
  para cada elemento del cluster
  para cada propiedad j del elemento
  añadir a variable [i] un valor aleatorio
  asignar el vector variable al centroide del cluster

Procedimiento y manual de uso

Para el desarrollo de esta práctica no se ha partido de ningún framework de metaheurísticas ni de un código proporcionado, más allá de la librería random dada para la generación de números pseudoaleatorios. La estructura empleada en la práctica consiste en una librería llamada util donde se han incluido todas las funcionalidades necesarias para ambos algoritmos y en un main, alojado en "cluster.cpp" desde donde se llaman a las funciones necesarias para la ejecución según los parámetros que se indiquen. Para las estructuras de datos se han usado struct y la librería std.

Se ha incluido un archivo makefile en la carpeta de software para la compilación del programa. Se ha creado un único ejecutable para ambos algoritmos, llamado cluster; éste se encuentra en el directorio bin. Para ejecutar el programa, es necesario pasar como parámetros, en el orden indicado, los siguientes datos:

- Modo: hace referencia al algoritmo que se ejecutará. En esta práctica, los parámetros aceptados serán "COPKM" para la ejecución del algoritmo Greedy y "BL" para la ejecución del algoritmo de Búsqueda Local
- Conjunto: hace referencia al conjunto de datos sobre el que se ejecutará el problema. Los parámetros aceptados serán "I" para el conjunto Iris, "E" para el conjunto Ecoli y "R" para el conjunto Rand. Para evitar tener que introducir la ruta completa de todos los ficheros en cada ejecución, éstos han de encontrarse dentro de un directorio llamado "datos", en el mismo directorio desde donde se llama a la función.
- Porcentaje restricciones: hace referencia el porcentaje del total de restricciones posibles que se aplicarán al problema. Los parámetros aceptados serán "10" y "20"
- Nº repeticiones: cantidad de veces que deseemos que se ejecute el algoritmo
- Resultado: si deseamos que se muestre, además de la salida estándar, el reparto de elementos en los distintos clusters, este parámetro será "S"; en caso contrario será "N"
- [Semillas]: hace referencia a las semillas de generación de números aleatorios. Se pueden introducir tantas como se deseen. Si no se introduce ninguna, se considerarán las semillas 1, 5, 10, 20 y 25.

Experimentos y análisis de resultados

Descripción de los casos del problema empleados y de los valores de los parámetros considerados en las ejecuciones de cada algoritmo

Para ambos algoritmos se han realizado 50 ejecuciones por conjunto de datos, es decir, 25 por cada conjunto de restricciones. Se han elegido 5 semillas, en nuestro caso {1, 5, 10, 20 y 25}, y para todas ellas se ha ejecutado el programa 5 veces.

Resultados obtenidos según el formato especificado

Se han incluido los resultados de todas las ejecuciones de los algoritmos Greedy y Búsqueda Local en formato pdf en unos archivos con el mismo nombre, respectivamente, alojados en el directorio resultados. También se ha incluido un archivo de nombre "BusquedaLocalVSGreedy.pdf" donde se realiza la comparación, y es éste el que será mostrado en el presente documento.

A continuación se presentan los resultados medios obtenidos en las 5 ejecuciones para cada una de las semillas. También se añade otra media aritmética para estos resultados.

Resultados obtenidos por el algoritmo BL con 10% de restricciones

	Iris				Ecoli				Rand			
BL 10%	*Tasa_C*	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
S=1	0,10484800	13	0,10614500	1897696011	2,33554200	639	2,86500000	94606768011	0,12527900	0	0,12527900	1385735871
S=5	0,10484800	13	0,10614500	1635234068	2,73964000	760	3,35865200	89506435241	0,12527900	0	0,12527900	1193764048
S=10	0,10484800	13	0,10614500	1921683361	2,72237400	790	3,38943200	79241580935	0,12527900	0	0,12527900	1654009979
S=20	0,10394340	60	0,11001720	1988175787	2,65276600	650	3,21350200	86682213825	0,12527900	0	0,12527900	1399232927
S=25	0,10426240	61	0,11026820	1993319449	2,68349600	866	3,41466800	87833912422	0,12527900	0	0,12527900	1523433674
Media	0,10454996	32	0,10774408	1887221735	2,62676360	741	3,24825080	87574182087	0,12527900	0	0,12527900	1431235300

Resultados obtenidos por el algoritmo BL con 20% de restricciones

	Iris				Ecoli				Rand			
BL 20%	*Tasa_C*	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
S=1	0,10545200	21	0,10657600	1692754726	2,23892400	1672	2,92701400	88679301878	0,12527900	0	0,12527900	1597538007
S=5	0,10545200	21	0,10657600	1702008832	2,63770600	1462	3,32961800	77495205879	0,12527900	0	0,12527900	1464822834
S=10	0,21574720	119	0,26783760	2020515844	2,33678800	1429	2,93412200	84301828494	0,12527900	0	0,12527900	1807955687
S=20	0,10545200	21	0,10657600	1770763536	2,31329400	1022	2,75954000	81210009637	0,12527900	0	0,12527900	1331706269
S=25	0,21574720	119	0,26783760	2015805843	3,71264000	1248	3,37029600	78079875331	0,12527900	0	0,12527900	1608349599
Media	0,14957008	60	0,17108064	1840369756	2,64787040	1367	3,06411800	81953244244	0,12527900	0	0,12527900	1562074479

Resultados obtenidos por el algoritmo COPKM con 10% de restricciones

	Iris				Ecoli				Rand			
COPKM 10%	*Tasa_C*	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
S=1	0,09000000	344	0,13000000	598221560	3,76000000	2696	11,08084000	98773251983	0,14793328	286	0,20147920	876250094
S=5	0,12000000	549	0,23000000	787841772	2,26000000	2797	6,68319000	73719608222	0,18878428	432	0,29787380	905064885
S=10	0,11000000	463	0,18000000	599154848	5,62000000	2823	13,96226600	11790516633	0,14973148	373	0,22012980	416852772
S=20	0,10000000	474	0,17000000	833957892	3,08000000	2603	7,36499400	16900344256	0,15728700	267	0,21109040	661344144
S=25	0,06000000	463	0,12000000	1020769352	3,74000000	2732	8,72719800	14621354147	0,17584260	369	0,25087180	561160113
Media	0,09600000	459	0,16600000	767989085	3,69200000	2730	9,56369760	43161015048	0,16391573	345	0,23628900	684134402

Resultados obtenidos por el algoritmo COPKM con 20% de restricciones

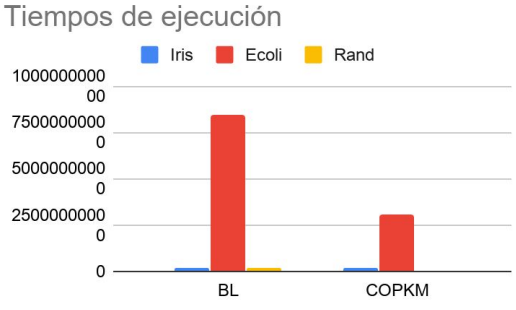
	Iris				Ecoli				Rand			
COPKM 20%	*Tasa_C*	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
S=1	0,09000000	672	0,13000000	2911102243	3,76000000	5392	18,39830000	20636062192	0,14787208	599	0,20721100	415062645
S=5	0,12000000	1100	0,24000000	1299209044	2,26000000	5594	11,10451400	15324537649	0,18872788	901	0,30915480	1081235834
S=10	0,11000000	945	0,19000000	1079664143	5,63000000	5644	22,30444000	11523673257	0,14973148	773	0,22721280	417159895
S=20	0,10000000	945	0,18000000	3357344397	3,74000000	5464	13,70982800	23791383469	0,15734780	545	0,21576380	498714534
S=25	0,06000000	928	0,13000000	4959091629	3,64000000	5478	16,48558200	20051020491	0,17584260	731	0,25455620	636809517
Media	0,09600000	918	0,17400000	2721282291	3,80600000	5514	16,40053280	18265335412	0,16390437	710	0,24277972	609796485

Análisis de resultados

Para facilitar el análisis, se ha realizado una comparativa entre los resultados medios obtenidos con los dos algoritmos estudiados, que a continuación se muestra

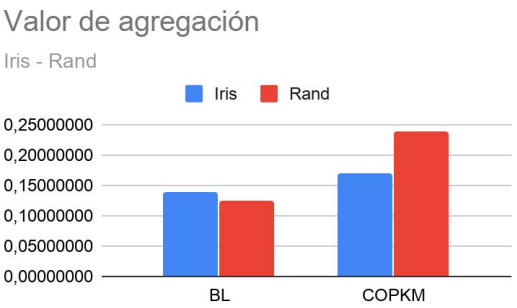
	Iris				Ecoli				Rand			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
BL	0,12706002	46	0,13941236	1863795746	2,63731700	1054	3,15618440	84763713165	0,12527900	0	0,12527900	1496654890
COPKM	0,09600000	688	0,17000000	1744635688	3,74900000	4122	12,98211520	30713175230	0,16391005	528	0,23953436	646965443

A simple vista, resulta fácil observar dos diferencias destacables entre los resultados: el algoritmo Greedy es más rápido que el algoritmo de Búsqueda Local pero nos devuelve un valor agregado mayor, es decir, es menos eficaz.

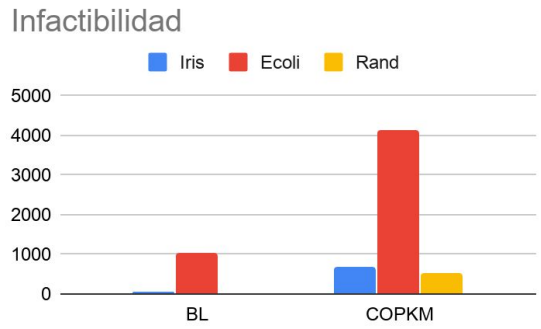


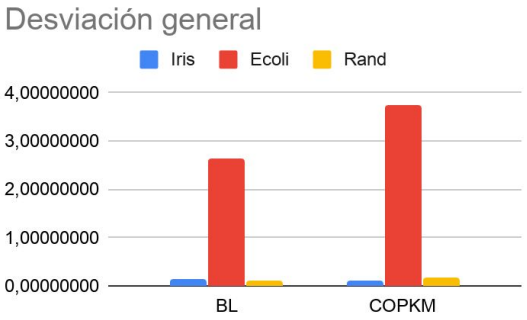
En la gráfica de la izquierda se hace notoria esta diferencia de velocidad entre los algoritmos mencionados anteriormente, aunque destaque más la enorme diferencia entre los tiempos de ejecución con el algoritmo BL del conjunto Ecoli y de los demás. Esto es debido a que el espacio de búsqueda para este conjunto de datos es mayor, por tanto podríamos afirmar que no es un algoritmo adecuado para grandes conjuntos de datos.

En la siguiente gráfica se muestra la comparación del valor de agregación obtenido para los conjuntos Iris y Rand (se ha omitido el conjunto Ecoli pues presentaba un valor de agregación tan elevado que hacía que los valores de los otros dos conjuntos fueran incomparables). A diferencia del algoritmo Greedy, se obtiene con BL un menor valor de agregación para el conjunto Rand. El algoritmo de Búsqueda Local resulta más efectivo para conjuntos de datos cuyo espacio de búsqueda sea pequeño



Aunque el algoritmo greedy implementado diera prioridad a la infactibilidad antes que a la desviación general, en los resultados se muestra que el algoritmo de búsqueda obtiene mejores tasas para ambos factores:





Bibliografía

- https://es.wikipedia.org-*"Análisis de grupos-Wikipedia, la enciclopedia libre"*
- https://www.ecured.cu- *"Clustering-EcuRed"*
- https://ccc.inaoep.mx- *"4.2 Búsqueda Local (Local Search)"*
- https://thales.cica.es- *"Algoritmos voraces"*