Práctica 3

Técnicas de Búsqueda basadas en Trayectorias

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

"La simplicidad es un prerrequisito para la fiabilidad"
-E. W. Dijkstra

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 4 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.
- Newthyroid: 215 datos con 5 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 que estudiarán 4 algoritmos basados en trayectorias. Se compararán los resultados obtenidos con dichos algoritmos, evaluados en función de la desviación general, de la infactibilidad y del tiempo obtenido con diferentes semillas de aleatoriedad, con los resultados obtenidos con los algoritmos de la primera práctica (Búsqueda Local y Greedy).

Descripción de la aplicación de los algoritmos empleados

El esquema de estructuras de datos empleado para el problema es el mismo que el empleado para la práctica 2:

- 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.
- Las restricciones se almacenan en una matriz (vector de vector) de enteros M.
- Un cromosoma o solución equivaldrá a una partición, consistente en un vector de tantos enteros sin signo como elementos tenga el conjunto seleccionado. En cada posición de este vector se almacenará el número del cluster al que ha sido asignado el elemento de X correspondiente a dicha posición.

Para permitir la reutilización del código de la primera práctica, se ha añadido una función que permite adaptar una solución a la estructura de datos anterior:

- 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.

Conversión de solución a antigua estructura de datos (práctica 2)

Para todos las posiciones i de la solución
  asignar a la variable cluster del elemento X[i] el valor de solución[i]
  añadir al vector datos del cluster C[solución[i]] la posición i
Fin para
Actualizar centroides
Devolver C (X ha sido pasado por referencia)

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 (práctica 1)

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 (práctica 1)

```
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 (práctica 2)

```
Para cada fila de M
  para cada columna de M
    si M[i][j] es 1 y los elementos i y j de la solución son distintos
      incrementar infactibilidad en 1
    si M[i][j] es -1 y los elementos i y j de la solución 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 (práctica 2)

```
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
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 }
```

Aparte 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 (práctica 1)

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

- Operador para evitar que una solución tenga algún cluster vacío: Realiza modificaciones aleatorias hasta que esta restricción sea cumplida

No vacío

Generar vector check de tamaño num_clus
Para cada elemento i de la solución
  Aumentar check[i] en una unidad
Fin para
Mientras no se cumpla la restricción
  Para cada i de num_cluster
  si check[i] vale 0
  seleccionar aleatoriamente un cluster rand
  almacenar valor de solución[rand] en aux
  modificar el valor de solución[rand] a i
  aumentar check[i] en una unidad
  reducir check[aux] en una unidad
  salir del para
  si ningún elemento de check vale 0
  se cumple la restricción
  Fin para
Fin mientras
Devolver solución

- Operador de selección: Devuelve la mejor solución entre dos (no aleatorias)

Torneo
Almacenar función objetivo de la primera solución Almacenar función objetivo de la segunda solución Devolver solución con mejor función objetivo

- Búsqueda local: adaptación de la búsqueda local a las características de esta práctica

Búsqueda local adaptada

Generar una nueva solución e igualarla a la actual
Generar un vector orden de tamaño num elem y desordenarlo
Almacenar función objetivo de la solución actual
Mientras haya cambios y contador < evaluaciones máximas
  cambios = false
  para cada i de orden
  aplicar operador cambio cluster sobre nueva solución

aumentar contador
almacenar la función objetivo de la nueva solución
si la función objetivo de la nueva solución es mejor a la anterior
solución es nueva solución
actualizar función objetivo
cambios = true
desordenar orden
salir de para
fin para
Fin mientras
Devolver solución

Cambio cluster (generador de vecinos)

Obtener un nuevo cluster aleatoriamente distinto a solución[elemento].
Asignar a solución[elemento] el nuevo cluster
Modificar hijo para que ningún cluster esté vacío (no vacío)
Devolver solución

Pseudocódigo del método de búsqueda

Enfriamiento Simulado (ES)

El algoritmo del Enfriamiento Simulado es un algoritmo de búsqueda caracterizado por el uso de una variable temperatura, altamente inicializada, la cual será reducida iterativamente mediante un mecanismo de enfriamiento. En cada iteración se generan soluciones vecinas y, a diferencia de la búsqueda local, aun si la función objetivo se la solución vecina es peor que la actual, todavía cabe la posibilidad de que sustituya a la solución actual; de esta forma el algoritmo podrá salir de óptimos locales en los que quedaría atrapados si se tratará de una búsqueda local.

```
Generar solución actual aleatoriamente
Generar mejor solución igual a solución actual
Almacenar la función objetivo de ambas soluciones
Aumentar el contador de evaluaciones
Inicializar parámetros *
Mientras que la temperatura inicial sea menor que temperatura fin
  multiplicar temperatura inicial por temperatura fin
Fin mientras
Mientras haya éxito y temperatura sea mayor que temperatura fin
  mientras éxitos sea menos que max éxitos y vecinos sea menor que max vecinos
    Generar solución vecina cambiando un cluster aleatorio de la solución actual
    almacenar su función objetivo
    aumentar contador de vecinos
    almacenar diferencia entre la función objetivo vecino y la actual
    generar un número aleatorio prob entre 0 y 1
    si diferencia < 0 o exponencial de - (diferencia / temperatura) es mayor o igual a prob
      la solución actual pasa a ser la vecina
      actualizar función objetivo
      si la función objetivo nueva es menor que la función objetivo de mejor solución
        la mejor solución pasa a ser la solución actual
        actualizar función objetivo
        aumentar contador de éxitos
    aumentar evaluaciones
    si evaluaciones es mayor o igual que max evaluaciones salir del mientras
  fin mientras
  aplicar esquema de enfriamiento de Cauchy modificado **
  si evaluaciones es mayor o igual que max evaluaciones salir del mientras
Fin mientras
Devolver mejor solución
```

Inicialización de parámetros (*)
Max vecinos = 10 * num elementos Max exitos = 0.1 * max vecinos Num enfriamientos = $\frac{max\ evaluaciones}{max\ vecinos}$ Temperatura actual = temperatura inicial = $\frac{\mu * función\ objetivo}{log(\mu)}$

$\beta = \frac{temperatura\ ini - temperatura\ fin}{num\ enfriamientos * temperatura\ ini * temperatura\ fin}$
Esquema de Cauchy (**)
temperatura = $\frac{temperatura}{1 + \beta * temperatura}$

Búsqueda Multiarranque Básica (BMB)

El algoritmo Búsqueda Multiarranque es un algoritmo basado en trayectorias que alterna una fase de generación de soluciones candidatas con una fase de optimización. Para ello, genera 10 soluciones aleatorias sobre las que aplica la búsqueda local, y selecciona la que mejor función objetivo presente.

```
Generar solución actual aleatoriamente
Inicializar mínimo (1000)
Generar solución auxiliar
Para cada i de iteraciones (10)
  generar aleatoriamente los valores de solución auxiliar
  aplicar búsqueda local sobre solución auxiliar
  almacenar su función objetivo
  si la función objetivo es menor que el mínimo
    mínimo pasa a ser la función objetivo
    solución actual pasa a ser la solución auxiliar
  limpiar auxiliar
Fin para
Devolver solución
```

Búsqueda Local Reiterada (ILS)

El algoritmo Búsqueda Local Reiterada es un algoritmo basado en trayectorias que, a partir de una solución inicial, obtiene un mínimo local; sobre éste aplica un proceso de mutación y se volverá a buscar un mínimo local. Este proceso se repetirá varias veces (10), seleccionando siempre la mejor solución (optimizada o no) para mutar y optimizar.

```
Generar solución actual aleatoriamente
Inicializar mínimo (1000)
Aplicar búsqueda local sobre solución actual
Generar solución auxiliar
Para cada i de iteraciones (10)
   solución auxiliar es igual a solución actual
   aplicar cambio segmento sobre solución actual
   aplicar búsqueda local sobre solución actual
   solución actual es igual a mejor entre solución auxiliar y solución actual (torneo)
Fin para
Devolver actual
```

- Operador de mutación: Modifica aleatoriamente los clusters de aquellos elementos comprendidos en un segmento de tamaño fijo a partir de una solución dada

Cambio segmento

Obtener índice inicio aleatoriamente
Tam_segemento es tamaño de la solución por probabilidad de mutación
Añadir a vector seleccionados las posiciones comprendidas entre inicio e inicio+tam_segmento módulo num_elementos
Para cada i de tam_segmentos
  obtener un nuevo cluster aleatoriamente
  modificar el valor de solución[seleccionados[i]] al nuevo cluster
Fin para
Modificar hijo para que ningún cluster esté vacío (no vacío)
Devolver solución

Hibridación (ILS-ES)

El algoritmo de Hibridación consiste en una Búsqueda Local Reiterada (ILS) cuyo algoritmo se búsqueda será el algoritmo Enfriamiento Simulado (ES).

Generar solución actual aleatoriamente
Inicializar mínimo (1000)
Aplicar enfriamiento simulado sobre solución actual
Generar solución auxiliar
Para cada i de iteraciones (10)
  solución auxiliar es igual a solución actual
  aplicar cambio segmento sobre solución actual
  aplicar enfriamiento simulado sobre solución actual
  solución actual es igual a mejor entre solución auxiliar y solución actual (torneo)
Fin para
Devolver actual

Pseudocódigo del algoritmo de comparación

Los algoritmos de comparación empleados han sido los algoritmos de Búsqueda Local y Greedy COPKM empleados para la práctica anterior.

- La Búsqueda Local es un proceso iterativo que parte de una solución aleatoria y la mejora realizando modificaciones locales respecto a la función objetivo.

Búsqueda Local

```
Para cada elemento i de X
  asignar un cluster aleatorio a cluster
  asignar a dicho cluster la posición i
  modificar hasta que ningún cluster quede vacío
Calcular centroides
Almacenar función objetivo
Mientras haya cambios y contador < 100000
  cambios = false
  para cada elemento i de X
    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
    fin para
  fin para
  aumentar contador en una unidad
Fin mientras
```

- 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.

Greedy

```
Almacenar infactibilidad
Calcular centroides aleatorios
Mientras haya cambios
  Para cada elemento de X
    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
    Fin para
    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
    Fin para
    Si infactibilidad es igual a infactibilidad antes, diferencia = false
    Aumentar contador
Fin mientras
```

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 "trayectoria.h" donde se han incluido todas las funcionalidades necesarias para los algoritmos de esta práctica 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. También se hace uso de la librería "util.h" creada para la primera práctica. Para las estructuras de datos se ha usado 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 todos los 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
    - "BL" para la ejecución del algoritmo de Búsqueda Local
    - "ES" para el algoritmo Enfriamiento Simulado
    - "BMB" para el algoritmo Búsqueda Multiarranque Básica
    - "ILS" para el algoritmo Búsqueda Local Reiterada
    - "ILS-ES" para el algoritmo Híbrido ILS y ES
- Conjunto: hace referencia al conjunto de datos sobre el que se ejecutará el problema. 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. Los parámetros aceptados serán:
    - "I" para el conjunto Iris
    - "E" para el conjunto Ecoli
    - "R" para el conjunto Rand
    - "NT" para el conjunto Newthyroid
- 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 cada 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 en formato ods en archivos individuales con el mismo nombre que el parámetro especificado en la sección anterior para cada algoritmo en el directorio "ejecuciones". En el mismo directorio se han añadido 3 archivos de comparación:

- "comparacion_BL_COPKM.ods" muestra una comparación entre los resultados medios obtenidos para cada semilla con los algoritmos de la primera práctica.
- "comparacion_trayectoria.ods" muestra una comparación entre los resultados medios obtenidos para cada semilla con los cuatro algoritmos de esta práctica.
- "comparacion.ods" muestra la comparación entre las medias de los resultados obtenidos para todas las semillas con todos los algoritmos evaluados (práctica 1 y práctica 3). Aunque los tiempos se han obtenido en nanosegundos con una mayor precisión, en esta comparación se muestran en segundos para facilitar la comprensión.

Comparación de todos los algoritmos con 10% de restricciones

10,00 %	Iris				Ecoli			
Algoritmo	Tasa_C	Tasa_inf	Agregado	Tiempo	Tasa_C	Tasa_inf	Agregado	Tiempo
COPKM	0,09556858	378,2	0,1475478	0,767989085	2,3718714	2638	6,73392	43,16101505
BL	0,104848	13	0,106145	1,848027832	2,5326792	633,8	3,0743884	88,04224211
ES	0,2878606	466,4	0,3981334	0,602299425	1,642324	402,8	1,960232	42,7847003
BMB	0,296891	477,6	0,411563	0,177810547	7,472144	1823,6	9,858702	2,847494624
ILS	0,2997574	466,4	0,4124678	0,181502605	7,480302	1815,4	9,889738	2,829392974
ILS-ES	0,1223562	160,2	0,1414084	2,510821335	3,925832	1353,6	5,152976	44,18902378

10,00 %	Rand				Newthyroid			
Algoritmo	Tasa_C	Tasa_inf	Agregado	Tiempo	Tasa_C	Tasa_inf	Agregado	Tiempo
COPKM	0,09524704	609,6	0,2109796	0,684134402	1,2921686	1001,4	2,5783302	4,919350623
BL	0,125279	0	0,125279	1,341885525	0,6809574	1130,8	1,2526174	1,68194946
ES	0,3940468	473	0,5448494	0,429834004	1,675708	1168,6	2,558968	2,011553417
BMB	0,4028432	481	0,5595422	0,19881342	2,012414	1165,6	2,966818	0,459959677
ILS	0,4061652	473,6	0,5597098	0,17547463	1,993814	1161,8	2,956038	0,415400804
ILS-ES	0,117006	0	0,117006	2,446102816	0,600867	1125	1,15348	16,90341048

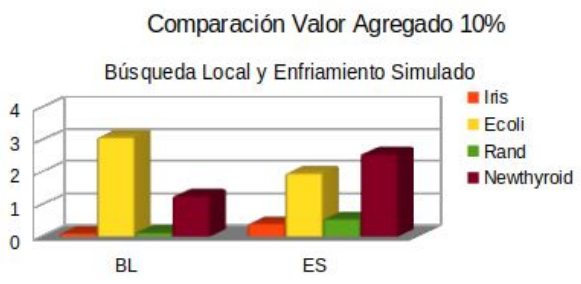
Comparación de todos los algoritmos con 20% de restricciones

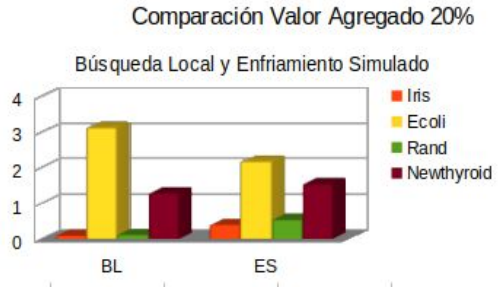
20,00 %	Iris				Ecoli			
Algoritmo	Tasa_C	Tasa_inf	Agregado	Tiempo	Tasa_C	Tasa_inf	Agregado	Tiempo
COPKM	0,09561876	750,4	0,15028562	2,721282291	3,0555152	5008,8	11,936774	18,26533541
BL	0,105452	21	0,106576	1,840369756	2,572108	1195	3,129222	81,95324424
ES	0,2901504	950	0,407923	0,525187726	1,850818	783,2	2,183482	43,79667244
BMB	0,295879	965,4	0,4189026	0,22465542	7,468858	3620,2	9,91157	3,885959512
ILS	0,297097	949	0,4186606	0,193860338	7,47343	3619,4	9,955826	2,66755436
ILS-ES	0,1449928	373,2	0,177646	2,589549109	3,859818	2862,6	5,172022	49,1339583

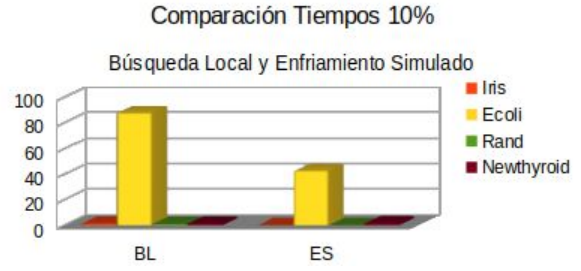
20,00 %	Rand				Newthyroid			
Algoritmo	Tasa_C	Tasa_inf	Agregado	Tiempo	Tasa_C	Tasa_inf	Agregado	Tiempo
COPKM	0,09518544	1234,2	0,2194204	0,609796485	1,292019	2031,6	2,6550182	2,807681082
BL	0,125279	0	0,125279	1,562074479	0,712057	2220,6	1,299056	1,781757686
ES	0,3947178	930,6	0,5523612	0,456780183	0,9294762	2083,6	1,5531252	5,539835767
BMB	0,4010894	945,8	0,5656966	0,229665134	2,011978	2328,8	3,011912	0,525957112
ILS	0,4048538	983	0,5744396	0,194126264	2,010576	2320,6	3,008138	0,443787785
ILS-ES	0,117006	0	0,117006	2,635057875	0,7143204	2104,6	1,261554	19,70342138

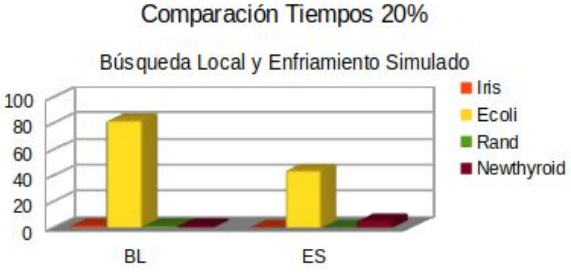
Análisis de resultados

El algoritmo Enfriamiento Simulado presenta los mejores valores obtenidos para el conjunto Ecoli, el de mayor tamaño, aunque sin llegar a su mínimo global. No ocurre lo mismo con los otros conjuntos, para los cuales obtiene resultados peores que los obtenidos con el algoritmo de Búsqueda Local. En cuanto al tiempo empleado, ES ha sido el doble de rápido que BL, exceptuando el conjunto Newthyroid. Basándonos en estas observaciones, parece evidente que BL es idóneo para conjuntos pequeños mientras que para conjuntos grandes ES es una buena alternativa, pues obtiene mejores valores en un tiempo razonable.









Mientras desarrollaba la práctica, intuía que el algoritmo Búsqueda Local Reiterada obtendría mejores prestaciones que al algoritmo de Búsqueda Multiarranque Básica, por la propia naturaleza de ambos algoritmos. Sin embargo, una vez obtenidos los resultados, se puede ver que las diferencias son insignificantes. ILS suele obtener mejores resultados que

BMB aunque no ocurre en todos los casos. También es el primero ligeramente más rápido que el segundo, aunque la diferencia no es nunca superior a un segundo. Tan minúsculas son las diferencias que no he considerado oportuno mostrarlas en una gráfica. En las tablas que se muestran a continuación se pueden observar los resultados obtenidos por ambos algoritmos por semilla. Al añadir esto, trato de resaltar el hecho de que hay mayor diferencia entre los resultados con las diferentes semillas con el algoritmo ILS, ya que depende mucho de la aleatoriedad en la fase de mutación. Al depender tanto, obtiene resultados bastante mejores que BMB, pero también resultados peores que hacen que el resultado medio esté igualado.

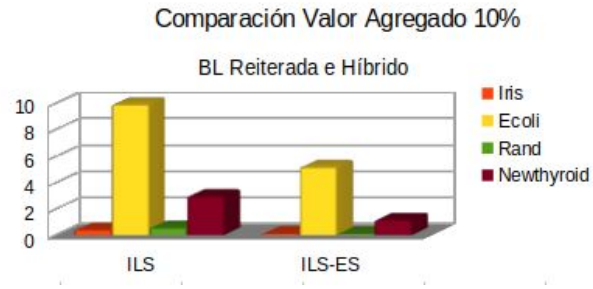
10,00 %	Iris				Ecoli				Rand				Newthyroid			
BMB	Tasa_C	Tasa_inf	Agregado	Tiempo	Tasa_C	Tasa_inf	Agregado	Tiempo	Tasa_C	Tasa_inf	Agregado	Tiempo	Tasa_C	Tasa_inf	Agregado	Tiempo
Semilla 1	0,296625	478	0,412932	164626995,4	7,47132	1804	9,87646	2654389315	0,402978	465	0,556165	195578675,2	2,02152	1192	2,98228	447731885,2
Semilla 5	0,293774	490	0,408528	174682183,2	7,46553	1832	9,84288	2575427657	0,404954	488	0,566738	211536239,6	2,01022	1172	2,9812	457571734
Semilla 10	0,299849	478	0,413789	166470343,2	7,46571	1810	9,79374	2694863588	0,40594	479	0,56014	194154854,2	2,00417	1145	2,93132	489155669,4
Semilla 20	0,297262	482	0,411236	191456983,2	7,50897	1829	9,92402	3127490827	0,397211	475	0,548703	193115761,8	2,01304	1150	2,96419	452775933,2
Semilla 25	0,296945	460	0,41133	191816228,8	7,44919	1843	9,85641	3185301735	0,403133	498	0,565965	199681571,2	2,01312	1169	2,9751	452563161,4
Media	0,296891	477,6	0,411563	0,177810547	7,472144	1823,6	9,858702	2,847494624	0,4028432	481	0,5595422	0,19881342	2,012414	1165,6	2,966818	0,459959677
ILS																
Semilla 1	0,300525	466	0,412957	169774690	7,46226	1857	9,96496	3008991340	0,407182	485	0,564469	152550738,2	2,01341	1143	3,00646	431311154,2
Semilla 5	0,300658	479	0,417597	205483839,2	7,46464	1835	9,82836	2587362563	0,400352	463	0,547373	183319598,4	1,9819	1170	2,97062	414489223,8
Semilla 10	0,301032	450	0,412666	164783262	7,45984	1801	9,81627	3737333082	0,406497	479	0,564059	188345004	1,95913	1163	2,89966	440029883,2
Semilla 20	0,295974	477	0,408533	180525167,6	7,5093	1787	9,90589	2139512676	0,408274	478	0,564448	185857869,2	1,99843	1167	2,94772	408041149,6
Semilla 25	0,300598	460	0,410586	186946066	7,50547	1797	9,93321	2673765208	0,408521	463	0,5582	167299937,8	2,0162	1166	2,95573	383132606,8
Media	0,2997574	466,4	0,4124678	0,181502605	7,480302	1815,4	9,889738	2,829392974	0,4061652	473,6	0,5597098	0,17547463	1,993814	1161,8	2,956038	0,415400804

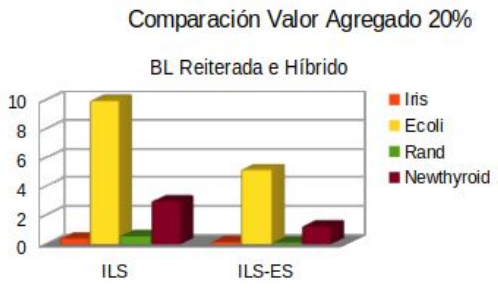
20,00 %	Iris				Ecoli				Rand				Newthyroid			
BMB	Tasa_C	Tasa_inf	Agregado	Tiempo	Tasa_C	Tasa_inf	Agregado	Tiempo	Tasa_C	Tasa_inf	Agregado	Tiempo	Tasa_C	Tasa_inf	Agregado	Tiempo
Semilla 1	0,296576	970	0,421647	210225803,8	7,47049	3618	9,86092	4058420808	0,404896	959	0,572456	202027771,6	2,02649	2331	3,0143	532166511,2
Semilla 5	0,296891	941	0,416452	255062648,2	7,45173	3588	9,91303	4044913102	0,406609	957	0,579044	236453534	1,99896	2313	2,98129	518696913,2
Semilla 10	0,295708	1003	0,424915	218676358,4	7,45685	3645	9,938	2745535693	0,402941	965	0,571176	226165123,8	2,00809	2308	3,03215	542587796,2
Semilla 20	0,297826	959	0,418064	212306623,2	7,46496	3643	9,9103	4109426091	0,394876	908	0,548105	258591415,4	1,99897	2351	2,99868	501940620,4
Semilla 25	0,292394	954	0,413435	227005664	7,50026	3607	9,9356	4471501867	0,396125	940	0,557702	225087823,8	2,02738	2341	3,03314	534393720,2
Media	0,295879	965,4	0,4189026	0,22465542	7,468858	3620,2	9,91157	3,885959512	0,4010894	945,8	0,5656966	0,229665134	2,011978	2328,8	3,011912	0,525957112
ILS																
Semilla 1	0,298757	966	0,420911	193715952,6	7,504	3663	10,0628	2769702751	0,408434	982	0,578959	175212994,2	2,02009	2316	3,00281	428213559
Semilla 5	0,291283	930	0,413576	185568730,6	7,46586	3640	10,0488	2372303465	0,402549	953	0,565414	188851593,8	2,0093	2311	2,98348	438666627
Semilla 10	0,296987	950	0,419109	196106919,2	7,43092	3593	9,84402	2264602100	0,405656	1021	0,582331	203748374,6	1,99722	2299	2,98652	473710993,2
Semilla 20	0,299044	940	0,418657	194202227,8	7,43945	3594	9,83928	2797779477	0,404973	958	0,573031	190947381,2	1,99568	2327	2,97383	444064877,2
Semilla 25	0,299414	959	0,42105	199707861	7,52692	3607	9,98423	3133384009	0,402657	1001	0,572463	211870975,6	2,03059	2350	3,09405	434282869,8
Media	0,297097	949	0,4186606	0,193860338	7,47343	3619,4	9,955826	2,66755436	0,4048538	983	0,5744396	0,194126264	2,010576	2320,6	3,008138	0,443787785

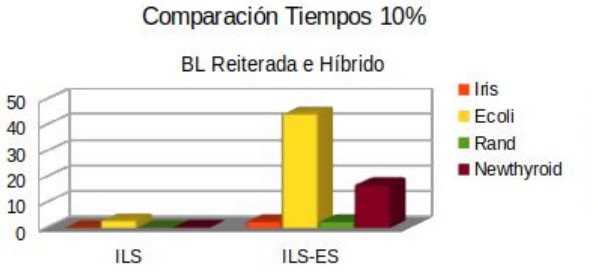
Nota: Los tiempos medios de cada semilla se expresan en nanosegundos. En las medias de éstos han sido convertidos a segundos

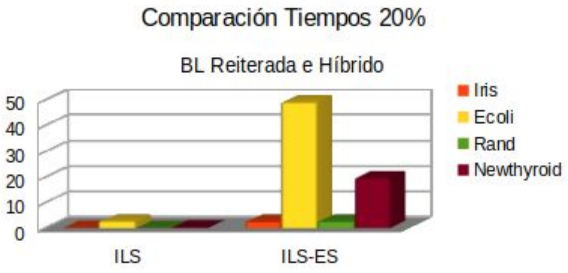
Los valores agregados obtenidos por ambos algoritmos no son buenos si los comparamos con los otros presentados en esta práctica. Con tan pocas evaluaciones (10000 frente a 100000), la búsqueda local no es capaz de obtener un resultado óptimo para ninguno de los conjuntos. La diferencia abismal entre los tiempos de BL y de éstos dos algoritmos (llegando a tardar apenas 3 segundos en ejecutarse sobre Ecoli frente a los 80 segundos que tarda la BL) hace evidente que la mayor parte del tiempo empleado por la BL se pierde en la exploración del vecindario en las últimas iteraciones. También es en las últimas iteraciones donde mejora su valor agregado.

El algoritmo ILS, al hibridarse con ES, mejora considerablemente los resultados obtenidos, aunque sacrificando el tiempo de ejecución. Podemos afirmar entonces que no sucede con ES lo mencionado anteriormente sobre BL. ES, aún reduciendo el número máximo de evaluaciones a un 10% del mismo, sigue obteniendo buenos resultados, incluso mejores a los obtenidos con un máximo de 100000 evaluaciones; esto es debido a que ES no rechaza vecinos que obtengan una peor función objetivo que la actual con el fin de salir de mínimos locales. Aunque esto funcione en el caso del conjunto Ecoli, con el resto de conjuntos ha supuesto un empeoramiento en los resultados. También el tiempo empleado parece no ser exponencial con respecto al número de iteraciones realizadas, como supusimos con BL. ILS-ES obtiene entonces mejores valores agregados pero peores tiempos que ILS con BL.

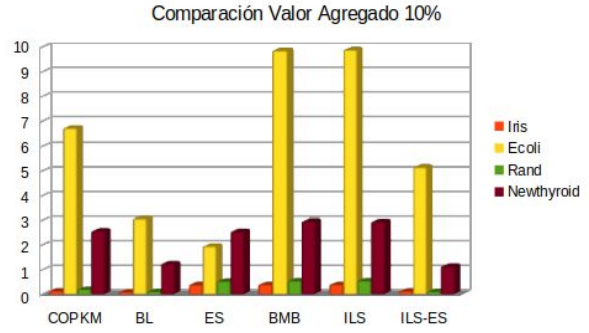


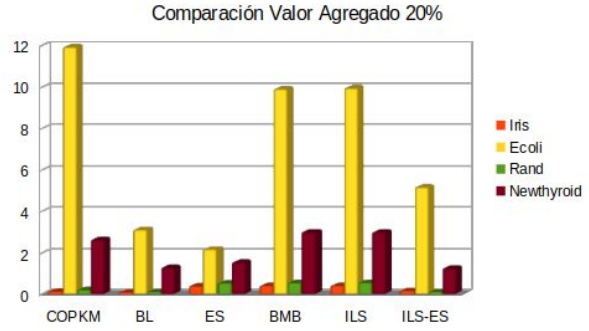


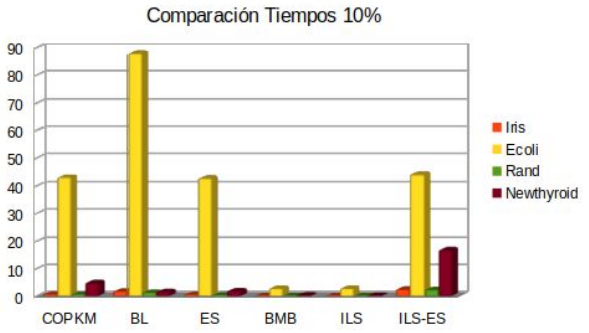


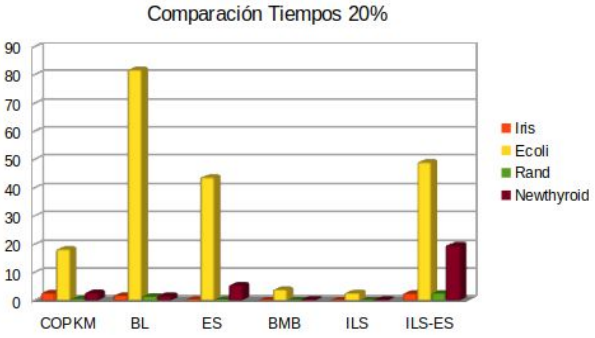


Para los conjuntos Iris y Ecoli (10 y 20% de restricciones) BL sigue obteniendo mejores valores agregados frente a ILS-ES, sin embargo, si hubiera que seleccionar un algoritmo para los cuatro conjuntos, basándonos en la relación valor agregado - tiempo, el algoritmo ILS-ES es el mejor de todos los presentados.









Bibliografía

- https://sci2s.ugr.es
- http://personales.upv.es/vyepesp/06myx01.pdf
- https://es.coursera.org
- https://ccc.inaoep.mx
- https://upcommons.upc.edu

"Se trata más de ser suficientemente bueno que de ser bueno o malo"
-James Bach