UNIVERSIDAD
DE GRANADA

METAHEURÍSTICA

Práctica 3.

Problema del Agrupamiento con Restricciones

Lukas Häring García

1. Introducción.

Existen problemas que actualmente no se conoce ninguna estrategia para la búsqueda de una solución óptima en tiempo polinómico, es por ello que autores han ideado nuevas estrategias, cada vez más abstracta, para encontrar o aproximar la solución óptima en tiempo polinómico.

El problema propuesto por la asignatura es conocido como "Problema del Agrupamiento", este problema se puede resolver en un tiempo $O(n^{dk+1} log\ n)$ [1], tal que son fijados "k", el número de clusters, y "d", la dimensión. "n" el número de entidades. Este tiempo está categorizado como polinómicos, es por ello que ha sido modificado para éste convertirse en un problema duro.

Ahora, el problema presenta dos tipos de restricciones, deben unirse **(*"Must-Link"*)** y no pueden unirse **("*Not-Link*")**. Utilizando los parámetros anteriores sobre unos *datasets*, vamos a proponer diferentes métodos para la aproximación de la solución óptima: La primera, la estrategia *codiciosa* **(Greedy)**, esta será el pilar para poder comparar los demás métodos; La segunda, la búsqueda local, en vez de explorar todo el conjunto de soluciones (que no es posible en tiempo polinómico), vamos a explorar un subconjunto de elementos e intentar converger a una solución aceptable; Otros métodos más ambiciosos inspirados en la naturaleza...

El propósito de esta práctica es la experimentación con algoritmos no rudimentarios para obtener una aproximación más o menos óptima en un tiempo aceptable, además de importar algoritmos bioinspirados para resolver algoritmos complejos.

En la segunda práctica hemos utilizado algoritmos bio-inspirados para obtener soluciones aceptables, utilizamos la técnica Darwinista evolutiva y una técnica híbrida.

Para las técnicas evolutivas, vamos a probar diferentes técnicas de cruce y selección, veremos que estas técnicas se ven favorecidas según la naturaleza del problema, siendo conveniente probar diferentes algoritmos.

En esta última práctica trata de acercarnos a las búsquedas basadas en trayectorias, estas parten de una solución inicial, de manera iterativa, la van reemplazando por otra de mejor calidad.

2. Descripción de los algoritmos.

Para el desarrollo de la siguiente práctica, vamos a suponer que se conoce todo el contenido expuesto en la práctica anterior, los cálculos de la distancia intra-cluster, desviación general, cálculo del factor lambda, etc.

La representación de los datos será igual a la anterior, vamos a utilizar un vector en representación entera.

La función objetivo está explicada en la práctica anterior, basta con recordar que se calcula de la siguiente forma y que no ha sido modificada para esta práctica:

$$f(A) = \lambda \cdot infeasibility(A) + general\_deviation(A)$$

Para la generación de la población inicial, vamos a generar soluciones aleatorias, que tienen la estructura que ya se ha comentado en la práctica anterior, estas soluciones deben cumplir la restricción de que todos los grupos tienen que tener al menos un elemento.

```
function Poblacion_inicial(num_grupos, ||X||, tamaño_poblacion)
   poblacion_inicial = []
   desde i=0 hasta num_grupos - 1, hacer:
     elemento_inicial = vector_aleatorio(num_grupos, ||X||)
     mientras incumple_restriccion(elemento_inicial, num_grupos):
         elemento_inicial = vector_aleatorio(num_grupos, ||X||)
     añadir(poblacion_inicial, elemento_inicial)
    devolver población_inicial
```

Vemos que incumple_restricciones, devolverá cierto o no, si existe un grupo vacío. Este devolverá un grupo de elementos (no tienen por que ser distintos) de posibles soluciones, que será el primer grupo en pasar nuestro algoritmo genético.

Finalmente, el elemento común a estos algoritmos es la búsqueda utilizada, en concreto una búsqueda local primer mejor, explicada en la primera práctica. Aunque vamos a recordar de forma general como trabajaba.
Se genera una población de pares de vecinos (índice, vecino), que indica en la posición del vector, que vecino (clúster) es el siguiente a recorrer. Vamos a ir visitando cada vecino hasta encontrar uno que reduzca el valor de la función objetivo, en ese momento, lo devolvemos.

Vamos a recordar el algoritmo de generación de vecinos, donde siempre se va a generar un par de vecinos válidos (cumpliendo las restricciones de clúster).

```
function Vecinos(padre, numero_clusters):
    vecinos = []
    para cluster,indice en padre, hacer:
      -- Miramos los vecinos de la izquierda
      para i en [0, cluster - 1]:
         padre[indice] = i
         si restriccion(padre, numero_clusters) entonces
            añadir [i, indice] en vecinos
         padre[indice] = cluster
     -- Miramos los vecinos de la derecha
     para i en [cluster - 1, numero_clusters]:
         padre[indice] = i
         si restriccion(padre, numero_clusters) entonces
            añadir [i, indice] en vecinos
         padre[indice] = cluster

    devolver permutados(vecinos)
```

La función “Restricción” está definida en la primera práctica, esta comprueba que no deja ningún cluster vacío, si esa así, se añadirá el par correspondiente (cluster, índice).

La lista de candidatos inicial se trata de un único elemento, de este elemento se genera un vecindario, con un criterio de selección primero mejor, esto quiere decir que de aquellos vecinos desordenados válidos generados por la función definida anteriormente, vamos sacando de forma ordenada y si existe uno mejor que el generador, repetimos iterativamente sobre estos, así hasta alcanzar el número máximo de llamadas a la función objetivo.

2.1. Enfriamiento Simulado (ES).

Como su propio nombre indica, se trata de un algoritmo de un sistema térmico, en concreto, de enfriamiento, por lo que $T_o > T_f$, es por ello que vamos a definir dichos puntos de temperatura así como su función de enfriamiento:

$T_o = \mu \frac{f(Initial\ Population)}{-log(\varphi)}$, $\mu = \varphi = 0.3 \Rightarrow T_o \approx 0.25 \cdot f(Initial\ Element)$ y $T_f = 10^{-3}$

Y la función de enfriamiento de Cauchy, definida como:
$T_{k+1} = \frac{T_k}{1+\beta \cdot T_k}$ con $\beta = \frac{\Delta T}{M \cdot T_o \cdot T_f}$, $lim_{T_f \to 0^+} \beta = \infty$ y $M = 0.1 \cdot \frac{f_{max}}{||X||}$
Vemos que β tiene que ser necesariamente grande, mientras que para T_{k+1} podemos resolver su recurrencia, obteniendo:
$T_k = \frac{T_o}{\beta \cdot k \cdot T_o + 1}$, vemos que es estrictamente decreciente por lo que llegará un punto en el que se aproxime a 0.

Vamos a utilizar como población inicial, un único elemento "*Initial Element*", a partir de este elemento, vamos a transformarlo en otro tal que sea una posible solución (o local) al problema.

Para cada etapa de enfriamiento, existen unas iteraciones de exploración, en las que se va a modificar la solución actual e ir visitando sus vecinos, aquellos cambios (a vecinos) serán tomados como válidos si minimizan la función del que vienen, pero esto tiene una desventaja, que puedan quedarse estancados en un mínimo local, por eso se le añade un factor aleatorio para la exploración:
$[0,1] < exp(-\frac{\Delta f}{kT})$, k es el factor de velocidad que fijamos en "1", mientras que T de manera natural, converge a 0. Para cada iteración donde nos aproximamos a 0, $-\frac{\Delta f}{T} \to -\infty$, esto hace que $exp(-\frac{\Delta f}{T}) \to 0$ o lo que es lo mismo, cuanto más nos aproximamos al "frío", menos probable es que vayamos a salir de un mínimo local (explotar), consiguiendo también así que no escapemos de un posible mínimo global.

Definimos ahora el número de iteraciones en las que se dedica a la explotación, para ello vamos a crear dos umbrales de generación y otro para los aciertos, es considerado un acierto si se consigue disminuir la función objetivo. Estos umbrales están prefijados al comienzo, donde el umbral de generación es $10 \cdot ||X||$, es decir, 10 veces más que el número de elementos en el problema, mientras que el umbral

de aciertos $0.10 \cdot ||X||$ o un diez por ciento del total de elementos, es decir, dos umbrales dependientes del problema.

Las condiciones de parada, la temperatura del sistema siempre tiene que ser mayor a la temperatura final, el número de evaluaciones de la función objetivo debe ser menor al umbral impuesto 100.000 y siempre debe haber al menos un acierto en la fase de enfriamiento.

2.2. Búsqueda Multiarranque Básica (BMB).

Esta técnica es la más simple y se trata de un acondicionamiento del algoritmo utilizado en la primera práctica, para ello, vamos a utilizar realizar una leve modificación al algoritmo de búsqueda local, donde éste ahora va a recibir como parámetro el padre al que aplicar la estrategia, anteriormente, este mismo también generaba la solución inicial, ahora no.

Inicialmente vamos a realizar una generar una solución válida para la primera iteración, que será el punto de mira inicial para las demás iteraciones y ver, si es posible, encontrar una solución mejor a este.

Destacar que el algoritmo de búsqueda local se ve reducido un 10%, a 10.000 ejecuciones de la función objetivo. Una vez aplicado dicho algoritmo, devolverá la solución optimizada.

Posteriormente, se realizará el mismo esquema 9 veces más, generará solución válida (no incumpla la restricción) y aplica búsqueda local, si el elemento encontrado por la técnica de búsqueda local, obtiene un valor de la función objetivo menor al mejor de las iteraciones anteriores, este será reemplazado.

Finalmente, se devolverá la mejor solución encontrada hasta el momento. Esta técnica es útil ya que explora 10 veces posibles soluciones, aunque no suficiente en problemas complejos y finalmente explora sobre las soluciones.

2.3. Búsqueda Local Reiterada (ILS).

Esta técnica hace uso de un algoritmo de búsqueda, por lo que se proponen dos, una búsqueda local y otra, una búsqueda por enfriamiento. Observamos claramente una técnica híbrida.

Esta técnica además hace uso de una técnica de mutación que es utilizada para la exploración y además de la técnica de búsqueda que es utilizada para la explotación.

Parecido a la técnica anterior, vamos a realizar un total de 10 pasadas, pero esta vez solo vamos a generar inicialmente una única solución, en la primera pasada vamos a generar la solución inicial y vamos a aplicar la técnica de búsqueda elegida, a partir del resultado obtenido, en las siguientes 9 iteraciones vamos a realizar una mutación sobre la solución (exploración) y una búsqueda sobre esta (explotación), si el resultado obtenido es mejor, será reemplazado, en caso contrario, mantendremos el anterior, se trata una mejora del algoritmo BMB ya que esta exploración es “relativamente” local ya que no se ve modificado al completo la solución.

La mutación utilizada se trata de una mutación de segmento, esto quiere decir que se coge un segmento de madera modular de tamaño 10% de la solución actual y se actualizan los valores a otros nuevos pero siempre comprobando que no se incumplan las restricciones:

```
function MutacionILS(padre, numero_clusters):
    solucion = copia padre
    inicio_segmento = aleatorio(0, ||padre||)
    tamanio_segmento = 0.1 * ||padre||
    mientras True:
        para i en tamanio_segmento
            solucion[(inicio_segmento + i) % ||padre||] = aleatorio(0, numero_clusters)
        si restriccion(padre) y solucion != padre Romper Bucle
    devolver solucion
```

Esto va a obligar que la solución propuesta sea obligatoriamente distinta (en al menos un elemento de ese 10%) de la que viene además de cumplir las restricciones.

Como hemos comentado antes, esta estrategia ayudará en la exploración cercana a la solución anterior, aunque tiene una desventaja, esta exploración (como todas las anteriores) pueden conseguir que escapemos de un mínimo global.

3. Análisis de los resultados

3.1 Dataset “Rand”.

El factor obtenido es $\lambda = 0.0072$, está definido el problema para 3 clusters.

3.1.1. Análisis Tabla de Resultados Greedy.

10% COPKM	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	1.00	280.00	3.01	1.15s
Ejecución 2	1.00	226.00	2.63	0.77s
Ejecución 3	0.92	158.00	2.05	1.12s
Ejecución 4	1.00	200.00	2.44	1.53s
Ejecución 5	1.21	238.00	2.92	1.91s
Media	1.02	220.40	2.61	1.30s

20% COPKM	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.76	68.00	1.01	1.16s
Ejecución 2	1.02	408.00	2.49	1.17s
Ejecución 3	0.86	302.00	1.95	1.59s
Ejecución 4	0.77	156.00	1.33	1.52s
Ejecución 5	0.85	256.00	1.77	1.57s
Media	0.85	238.00	1.71	1.40s

3.1.2. Análisis Tabla de Resultados Búsqueda Local.

10% BL	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.72	0	0.72	3.02s (1522 its)
Ejecución 2	0.72	0	0.72	3.28s (1671 its)
Ejecución 3	0.72	0	0.72	3.17s (1613 its)
Ejecución 4	0.72	0	0.72	2.63s (1353 its)
Ejecución 5	0.72	0	0.72	2.90s (1484 its)
Media	0.72	0	0.72	3.00s (1529 its)

20% BL	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.72	0	0.72	3.62s (1818 its)
Ejecución 2	0.72	0	0.72	3.25s (1618 its)
Ejecución 3	0.72	0	0.72	3.25s (1640 its)
Ejecución 4	0.72	0	0.72	3.03s (1575 its)
Ejecución 5	0.72	0	0.72	2.27s (1123 its)
Media	0.72	0	0.72	3.08s (1555 its)

3.1.3. Análisis Tabla de Enfriamiento Simulado.

10% ES	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.72	0	0.72	123.86s
Ejecución 2	0.72	0	0.72	124.70s
Ejecución 3	0.72	0	0.72	123.31s
Ejecución 4	0.72	0	0.72	125.23s
Ejecución 5	0.72	0	0.72	123.18s
Media	0.72	0	0.72	124.06s

20% ES	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.72	0	0.72	91.17s
Ejecución 2	0.72	0	0.72	89.35s
Ejecución 3	0.72	0	0.72	88.90s
Ejecución 4	0.72	0	0.72	94.34s
Ejecución 5	0.72	0	0.72	93.97s
Media	0.72	0	0.72	91.55s

3.1.4. Análisis Tabla de Búsqueda Multiarranque Básica.

10% BMB	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.72	0	0.72	91.72s
Ejecución 2	0.72	0	0.72	88.09s
Ejecución 3	0.72	0	0.72	87.51s
Ejecución 4	0.72	0	0.72	89.62s
Ejecución 5	0.72	0	0.72	95.71s
Media	0.72	0	0.72	90.53s

20% BMB	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.72	0	0.72	70.37s
Ejecución 2	0.72	0	0.72	64.30s
Ejecución 3	0.72	0	0.72	60.00s
Ejecución 4	0.72	0	0.72	59.96s
Ejecución 5	0.72	0	0.72	58.96s
Media	0.72	0	0.72	62.72s

3.1.5. Análisis Tabla de Búsqueda Local Reiterada.

10% ILS	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.72	0	0.72	50.05s
Ejecución 2	0.72	0	0.72	49.43s
Ejecución 3	0.72	0	0.72	47.39s
Ejecución 4	0.72	0	0.72	46.70s
Ejecución 5	0.72	0	0.72	49.48s
Media	0.72	0	0.72	48.61s

20% ILS	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.72	0	0.72	35.86s
Ejecución 2	0.72	0	0.72	35.02s
Ejecución 3	0.72	0	0.72	37.77s
Ejecución 4	0.72	0	0.72	32.54s
Ejecución 5	0.72	0	0.72	33.08s
Media	0.72	0	0.72	34.85s

3.1.6. Análisis Tabla de Búsqueda Local Reiterada Simulado.

10% ILS-ES	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.73	14	0.83	111.7s
Ejecución 2	0.72	0	0.72	114.48s
Ejecución 3	0.75	32	0.98	107.52s
Ejecución 4	0.76	40	1.05	96.94s
Ejecución 5	0.72	0	0.72	98.24s
Media	0.73	17.2	0.86	105.78s

20% ILS-ES	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.72	0	0.72	81.59s
Ejecución 2	0.72	0	0.72	73.57s
Ejecución 3	0.72	0	0.72	82.40s
Ejecución 4	0.72	0	0.72	75.13s
Ejecución 5	0.72	0	0.72	83.68s
Media	0.72	0	0.72	79.27s

3.1.7. Análisis Tabla de Resultados Medios.

10%	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
COPKM	1.02	220.40	2.61	1.30s
BL	0.72	0	0.72	3.00s (1529 its)
ES	0.72	0	0.72	124.06s
BMB	0.72	0	0.72	90.53s
ILS	0.72	0	0.72	48.61s
ILS-ES	0.73	17.2	0.86	105.78s

20%	**RAND (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
COPKM	0.85	238.00	1.71	1.40s
BL	0.72	0	0.72	3.08s (1555 its)
ES	0.72	0	0.72	91.55s
BMB	0.72	0	0.72	62.72s
ILS	0.72	0	0.72	34.85s
ILS-ES	0.72	0	0.72	79.27s

Como ya hemos visto anteriormente, se trata de un “dataset” simple, cuya solución es obtenida por la mayoría de algoritmos, posteriormente comentaremos el problema de ILS-ES, por qué en los resultados, obtiene resultados “malos” con respecto de los demás.

Además, comentar que los tiempos son en general más bajos que en la práctica anterior, ya que requiere mucho menos cómputo aleatorio y general.

3.2 Dataset “Iris”.

El factor obtenido es $\lambda = 0.0063$, también definido para 3 clusters.

3.2.1. Análisis Tabla de Resultados Greedy.

10% COPKM	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	1.16	522.00	4.47	2.26s
Ejecución 2	1.10	690.00	5.47	1.87s
Ejecución 3	0.78	454.00	3.66	1.13s
Ejecución 4	0.91	414.00	3.54	1.50s
Ejecución 5	0.94	384.00	3.38	1.89s
Media	0.98	492.8	4.10	1.73s

20% COPKM	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	1.13	1044.00	4.44	1.59s
Ejecución 2	0.93	866.00	3.68	2.35s
Ejecución 3	0.70	632.00	2.70	1.20s
Ejecución 4	0.85	598.00	2.74	2.41s
Ejecución 5	0.88	498.00	2.46	2.14s
Media	0.90	727,6	3.20	1.94s

3.2.2. Análisis Tabla de Búsqueda Local.

10% BL	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.67	0.00	0.67	2.69s (1344 its)
Ejecución 2	0.67	0.00	0.67	2.86s (1440 its)
Ejecución 3	0.67	0.00	0.67	2.93s (1450 its)
Ejecución 4	0.67	0.00	0.67	3.29s (1742 its)
Ejecución 5	0.67	0.00	0.67	3.42s (1804 its)
Media	0.67	0.00	0.67	3.04s (1556 its)

20% BL	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.67	0.00	0.67	3.63s (1781 its)
Ejecución 2	0.67	0.00	0.67	3.51s (1734 its)
Ejecución 3	0.67	0.00	0.67	3.00s (1486 its)
Ejecución 4	0.67	0.00	0.67	3.46s (1704 its)
Ejecución 5	0.67	0.00	0.67	3.55s (1766 its)
Media	0.67	0.00	0.67	3.63s (1781 its)

3.2.3. Análisis Tabla de Enfriamiento Simulado.

10% ES	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.67	0.00	0.67	108.64s
Ejecución 2	0.67	0.00	0.67	114.72s
Ejecución 3	0.67	0.00	0.67	109.93s
Ejecución 4	0.67	0.00	0.67	109.21s
Ejecución 5	0.67	0.00	0.67	113.18s
Media	0.67	0.00	0.67	111.14s

20% ES	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.67	0.00	0.67	88.68s
Ejecución 2	0.67	0.00	0.67	87.44s
Ejecución 3	0.67	0.00	0.67	90.18s
Ejecución 4	0.67	0.00	0.67	89.23s
Ejecución 5	0.67	0.00	0.67	88.41s
Media	0.67	0.00	0.67	88.79s

3.2.4. Análisis Tabla de Búsqueda Multiarranque Básica.

10% BMB	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.67	0.00	0.67	83.71s
Ejecución 2	0.67	0.00	0.67	89.77s
Ejecución 3	0.67	0.00	0.67	86.79s
Ejecución 4	0.67	0.00	0.67	84.37s
Ejecución 5	0.67	0.00	0.67	89.84s
Media	0.67	0.00	0.67	86.90s

20% BMB	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.67	0.00	0.67	70.47s
Ejecución 2	0.67	0.00	0.67	66.01s
Ejecución 3	0.67	0.00	0.67	59.44s
Ejecución 4	0.67	0.00	0.67	66.73s
Ejecución 5	0.67	0.00	0.67	69.05s
Media	0.67	0.00	0.67	66.34s

3.2.5. Análisis Tabla de Búsqueda Local Reiterada.

10% ILS	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.67	0.00	0.67	45.69s
Ejecución 2	0.67	0.00	0.67	38.89s
Ejecución 3	0.67	0.00	0.67	34.15s
Ejecución 4	0.67	0.00	0.67	39.22s
Ejecución 5	0.67	0.00	0.67	45.47s
Media	0.67	0.00	0.67	40.69s

20% ILS	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.67	0.00	0.67	36.18s
Ejecución 2	0.67	0.00	0.67	35.13s
Ejecución 3	0.67	0.00	0.67	34.87s
Ejecución 4	0.67	0.00	0.67	36.00s
Ejecución 5	0.67	0.00	0.67	37.10s
Media	0.67	0.00	0.67	35.86s

3.2.6. Análisis Tabla de Búsqueda Local Reiterada Simulado.

10% ILS-ES	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.67	18.00	0.78	45.69s
Ejecución 2	0.67	0.00	0.67	38.89s
Ejecución 3	0.67	10.00	0.73	34.15s
Ejecución 4	0.67	0.00	0.67	39.22s
Ejecución 5	0.67	0.00	0.67	45.47s
Media	0.67	5.60	0.70	40.68s

20% ILS-ES	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	0.67	0.00	0.67	76.24s
Ejecución 2	0.67	0.00	0.67	77.11s
Ejecución 3	0.67	0.00	0.67	71.96s
Ejecución 4	0.67	0.00	0.67	76.88s
Ejecución 5	0.67	0.00	0.67	64.83s
Media	0.67	0.00	0.67	73.40s

3.2.7. Análisis Tabla de Resultados Medios.

10%	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
COPKM	0.90	492.8	3.20	1.94s
BL	0.67	0.00	0.67	3.04s (1556 its)
ES	0.67	0.00	0.67	111.14s
BMB	0.67	0.00	0.67	86.90s
ILS	0.67	0.00	0.67	40.69s
ILS-ES	0.67	5.60	0.70	40.68s

20%	**IRIS (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
COPKM	0.98	727,6	4.10	1.73s
BL	0.67	0.00	0.67	3.63s (1781 its)
ES	0.67	0.00	0.67	88.79s
BMB	0.67	0.00	0.67	66.34s
ILS	0.67	0.00	0.67	35.86s
ILS-ES	0.67	0.00	0.67	73.40s

De manera similar al problema anterior, obtenemos soluciones buenas debido a que se trata de un dataset simple, los tiempos de la misma forma, son altos pero muy bajos comparados con la práctica anterior, algo que se agradece en esta práctica ya que Python es muy lento para la generación de números aleatorios.

3.3 Dataset “Ecoli”.

El factor obtenido es $\lambda = 0.0268$, un problema con 8 clusters.

3.3.1. Análisis Tabla de Greedy.

10% COPKM	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	23.54	1326.00	59.12	208.91s
Ejecución 2	28.54	1526.00	69.49	201.19s
Ejecución 3	26.73	1622.00	70.25	447.20s
Ejecución 4	32.74	1316.00	68.05	252.68s
Ejecución 5	34.22	1470.00	73.66	145.05s
Media	29.16	1452.00	68.11	251.01s

20% COPKM	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	26.07	2574.00	60.60	139.98s
Ejecución 2	29.19	2760.00	66.22	198.87s
Ejecución 3	32.94	3356.00	77.96	332.54s
Ejecución 4	30.97	2294.00	61.75	258.97s
Ejecución 5	33.18	2704.00	69.45	252.50s
Media	30.47	2737.60	67.20	236.57s

3.3.2. Análisis Tabla de Búsqueda Local.

10% BL	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	22.57	32.00	23.43	248.91s (22842 its)
Ejecución 2	22.72	36.00	23.69	210.88s (19171 its)
Ejecución 3	22.56	42.00	23.68	289.42s (27032 its)
Ejecución 4	22.58	34.00	23.50	292.90s (26797 its)
Ejecución 5	26.29	172.00	30.91	210.31s (19129 its)
Media	23.35	63.20	25.04	250.48s (22994 its)

20% BL	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	22.49	84.00	23.58	256.63s (22880 its)
Ejecución 2	21.98	134.00	23.78	253.50s (23528 its)
Ejecución 3	22.39	90.00	23.60	185.05s (16341 its)
Ejecución 4	22.36	116.00	23.92	222.64s (20570 its)
Ejecución 5	22.27	86.00	23.43	186.59s (16803 its)
Media	22.29	102.0	23.66	220.88s (20024 its)

3.3.3. Análisis Tabla de Enfriamiento Simulado.

10% ES	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	22.70	26.00	23.40	840.95s
Ejecución 2	22.23	44.00	23.41	902.29s
Ejecución 3	22.59	38.00	23.61	889.53s
Ejecución 4	22.72	30.00	23.53	896.95s
Ejecución 5	22.55	26.00	23.24	912.35s
Media	22.56	32.80	23.44	888.41s

20% ES	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	22.40	80.00	23.48	907.69s
Ejecución 2	22.63	68.00	23.54	939.58s
Ejecución 3	22.24	104.00	23.63	951.88s
Ejecución 4	22.64	92.00	23.87	962.33s
Ejecución 5	22.51	94.00	23.77	931.47s
Media	22.48	87.60	23.66	938.59s

3.3.3. Análisis Tabla de Búsqueda Multiarranque Básica.

10% BMB	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	25.02	152.00	29.10	1823.72s
Ejecución 2	22.62	248.00	29.27	1834.99s
Ejecución 3	24.32	174.00	28.99	1832.56s
Ejecución 4	23.22	168.00	27.73	1842.87s
Ejecución 5	23.21	182.00	28.10	1835.62s
Media	23.68	184.80	28.64	1833.95s

20% BMB	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	23.11	376.00	28.15	1147.51s
Ejecución 2	23.23	280.00	26.99	1204.14s
Ejecución 3	23.11	206.00	25.87	1269.32s
Ejecución 4	23.28	332.00	27.73	1314.76s
Ejecución 5	23.32	252.00	26.70	1332.61
Media	23.21	282.20	27.09	1251.67

3.3.4. Análisis Tabla de Búsqueda Local Reiterada.

10% ILS	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	23.05	44.00	24.23	1403.70s
Ejecución 2	22.84	64.00	24.56	1418.45s
Ejecución 3	23.07	54.00	24.51	1473.21s
Ejecución 4	22.70	74.00	24.69	1430.80s
Ejecución 5	23.12	56.00	24.62	1481.59s
Media	22.96	58.40	24.52	1441.55s

20% ILS	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	22.70	112.00	24.20	917.81s
Ejecución 2	22.47	188.00	24.99	939.70s
Ejecución 3	22.67	162.00	24.85	815.21s
Ejecución 4	23.23	136.00	25.06	828.59s
Ejecución 5	22.94	142.00	24.85	840.82s
Media	22.80	148.00	24.79	868.43s

3.3.5. Análisis Tabla de Búsqueda Local Reiterada Simulado.

10% ILS-ES	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	40.62	1720.00	86.77	355.97s
Ejecución 2	37.82	1728.00	84.18	343.44s
Ejecución 3	38.71	1740.00	85.40	348.37s
Ejecución 4	38.46	1834.00	87.67	338.86s
Ejecución 5	38.18	1712.00	84.11	358.99s
Media	38.76	1746.80	85.63	349.13s

20% ILS-ES	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	37.10	3688.00	86.57	188.34s
Ejecución 2	37.28	3572.00	85.20	186.10s
Ejecución 3	38.18	3986.00	91.65	181.38s
Ejecución 4	37.84	3484.00	84.57	192.41s
Ejecución 5	38.55	3766.00	89.07	190.51s
Media	37.79	3699.20	87.41	187.55s

3.3.6. Análisis Tabla de Resultados Medios.

10%	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
COPKM	29.16	1452.00	68.11	251.01s
BL	23.35	63.20	25.04	250.48s (22994 its)
ES	22.56	32.80	23.44	888.41s
BMB	23.68	184.80	28.64	1833.95s
ILS	22.96	58.40	24.52	1441.55s
ILS-ES	38.76	1746.80	85.63	349.13s

20%	**ECOLI (k = 8)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
COPKM	30.47	2737.60	67.20	236.57s
BL	22.29	102.0	23.66	220.88s (20024 its)
ES	22.48	87.60	23.66	938.59s
BMB	23.21	282.20	27.09	1251.67
ILS	22.80	148.00	24.79	868.43s
ILS-ES	37.79	3699.20	87.41	187.55s

En este dataset podemos observar claramente las diferencias, observamos que en general el agregado es bajo inferior a 30.0, la solución greedy es alta al igual que ILS-ES, se comentará más adelante el motivo.

En general vemos que todos obtienen una solución de calidad, además, podemos observar que ES obtiene mejor resultado, esto es debido a que ES explora mucho y dada unas iteraciones, intensifica la explotación, dando resultados de calidad. Aún así, no podemos afirmar cuál es mejor debido a que estos son dependientes a una semilla, por lo que los valores pueden oscilar.

Finalmente comentar que para este problema, los tiempos de ejecución son altos, pero para ILS-ES este enfría demasiado rápido por la definición del problema.

3.4 Dataset “Newthyroid”.

El factor obtenido es $\lambda = 0.0365$, Con una agrupación de 3 clusters.

3.4.1. Análisis Tabla de Greedy.

10% COPKM	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	13.21	1034.00	51.03	8.67s
Ejecución 2	12.84	750.00	40.27	5.48s
Ejecución 3	13.37	782.00	41.97	5.43s
Ejecución 4	12.91	720.00	39.24	4.33s
Ejecución 5	10.32	1398.00	61.45	4.51s
Media	12.53	936.80	46.79	5.68s

20% COPKM	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	12.95	1802.00	45.89	9.33s
Ejecución 2	12.99	1366.00	37.96	4.58s
Ejecución 3	13.07	1448.00	39.55	5.61s
Ejecución 4	12.84	1510.00	40.45	4.60s
Ejecución 5	13.36	2506.00	59.17	4.73s
Media	13.04	1726.40	44.60	5.77s

3.4.2. Análisis Tabla de Búsqueda Local.

10% BL	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	14.09	4.00	14.23	9.42s (2740 its)
Ejecución 2	14.09	4.00	14.23	9.87s (2883 its)
Ejecución 3	10.09	194.00	17.99	11.14s (3466 its)
Ejecución 4	14.09	4.00	14.23	8.84s (2703 its)
Ejecución 5	10.87	192.00	17.90	15.37s (4929 its)
Media	12.81	79.6	15.72	10.93s (3344 its)

20% BL	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	14.29	0.00	14.29	8.43s (2499 its)
Ejecución 2	10.81	462	19.26	7.95s (2413 its)
Ejecución 3	10.86	462	19.30	11.16s (3374 its)
Ejecución 4	14.29	0.00	14.29	8.13s (2481 its)
Ejecución 5	14.29	0.00	14.29	7.30s (2190 its)
Media	12.91	184.8	16.28	8.59s (2591 its)

3.4.3. Análisis Tabla de Enfriamiento Simulado.

10% ES	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	14.09	4.00	14.23	205.31s
Ejecución 2	14.09	4.00	14.23	215.73s
Ejecución 3	10.62	192.00	17.65	215.65s
Ejecución 4	14.09	4.00	14.23	211.70s
Ejecución 5	14.09	4.00	14.23	209.99s
Media	13.40	41.60	14.92	211.68s

20% ES	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	14.29	0.00	14.29	225.12s
Ejecución 2	14.29	0.00	14.29	228.52s
Ejecución 3	14.29	0.00	14.29	211.42s
Ejecución 4	14.29	0.00	14.29	212.43s
Ejecución 5	10.77	460.00	19.18	211.44s
Media	13.58	92.0	15.27	217.79s

3.4.4. Análisis Tabla de Búsqueda Multiarranque Básica.

10% BMB	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	14.09	4.00	14.23	278.23s
Ejecución 2	14.09	4.00	14.23	211.10s
Ejecución 3	14.09	4.00	14.23	219.55s
Ejecución 4	14.09	4.00	14.23	252.96s
Ejecución 5	14.09	4.00	14.23	224.55s
Media	14.09	4.0	14.23	237.28s

20% BMB	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	14.29	0.00	14.29	96.55s
Ejecución 2	14.29	0.00	14.29	98.53s
Ejecución 3	14.29	0.00	14.29	105.61s
Ejecución 4	14.29	0.00	14.29	99.88s
Ejecución 5	14.29	0.00	14.29	89.22s
Media	14.29	0.0	14.29	97.87s

3.4.5. Análisis Tabla de Búsqueda Local Reiterada.

10% ILS	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	14.09	4.00	14.23	118.37s
Ejecución 2	14.09	4.00	14.23	124.26s
Ejecución 3	14.09	4.00	14.23	147.20s
Ejecución 4	14.09	4.00	14.23	112.31s
Ejecución 5	14.09	4.00	14.23	184.66s
Media	14.09	4.0	14.23	137.36s

20% ILS	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	14.29	0.0	14.29	54.47s
Ejecución 2	14.29	0.0	14.29	47.92s
Ejecución 3	14.29	0.0	14.29	65.87s
Ejecución 4	14.29	0.0	14.29	49.97s
Ejecución 5	14.29	0.0	14.29	52.17s
Media	14.29	0.0	14.29	54.08s

3.4.6. Análisis Tabla de Búsqueda Local Reiterada Simulado.

10% ILS-ES	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	13.94	96.00	17.45	158.71s
Ejecución 2	14.26	46.00	15.95	158.62s
Ejecución 3	14.14	50.00	15.97	155.11s
Ejecución 4	14.30	112.00	18.39	158.95s
Ejecución 5	14.28	92.00	17.64	156.35s
Media	14.18	79.20	17.08	157.55s

20% ILS-ES	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
Ejecución 1	14.17	164.00	17.17	69.26s
Ejecución 2	14.25	224.00	18.35	71.06s
Ejecución 3	14.40	202.00	18.09	71.21s
Ejecución 4	14.24	182.00	17.57	74.19s
Ejecución 5	14.22	152.00	17.00	72.56s
Media	14.25	184.80	17.63	71.66s

3.4.7. Análisis Tabla de Resultados Medios.

10%	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
COPKM	12.53	936.80	46.79	5.68s
BL	12.81	79.6	15.72	10.93s (3344 its)
ES	13.40	41.60	14.92	211.68s
BMB	14.09	4.0	14.23	237.28s
ILS	14.09	4.0	14.23	137.36s
ILS-ES	14.18	79.20	17.08	157.55s

20%	**NEWTHYROID (k = 3)**			
	Tasa_C	***Tasa_inf***	***Agr.***	***T***
COPKM	13.04	1726.40	44.60	5.77s
BL	12.91	184.8	16.28	8.59s (2591 its)
ES	13.58	92.0	15.27	217.79s
BMB	14.29	0.0	14.29	97.87s
ILS	14.29	0.0	14.29	54.08s
ILS-ES	14.25	184.80	17.63	71.66s

Este dataset mantiene dos soluciones mínimas locales, es de ahí que aunque se trate de un problema sencillo, es el algoritmo explorador quien encuentre una mejor solución. En nuestro caso, podemos encontrar que todos son buenos, a excepción de ILS-ES aunque ES es peor ya que le dedica poco tiempo a explorar y mucho a explotar.

Los tiempos son equivalentes y el agregado es bajo en general, aquí no se ve representado ILS-ES por el tipo de conjunto, como hemos comentado antes, un conjunto siemple.

3.5 Análisis Global de los resultados.

Vemos que para todos los problemas, estos algoritmos son buenos en ofrecer una solución de calidad, aunque podemos ver un error de ± 2.00 que puede ser debido a las semillas utilizadas ya que cada algoritmo hace uso de valores aleatorios, haciendo que cada algoritmo esté en situaciones diferentes.

Estos algoritmos en comparación con la práctica anterior, han requerido un 100% menos tiempo del que habían utilizado los algoritmos genéticos ya que en general, requieren menos memoria y menos números aleatorios, siendo un factor importante en los tiempos de cómputo.

Observamos que Enfriamiento Simulado ofrece buenos resultados en general, esta técnica es buena si se utiliza una buena función de enfriamiento ya que prevalece la exploración en primeras iteraciones ya que la probabilidad en entrar en el bucle es mayor, como se comentó anteriormente. Luego este utilizará la explotación para ofrecer una solución de calidad.

Por otro lado, Búsqueda Multiarranque Básica ofrece unas soluciones similares a la Búsqueda local, esto es debido a que utiliza la búsqueda local como exploración y utiliza 10 iteraciones para la exploración, generando una solución inicial. En comparativa con el anterior, vemos que es un algoritmo que, en general, explora menos, es por eso que no podemos dar una afirmación clara ya que al utilizar valores aleatorios, hemos podido tener la "suerte" de obtener mejor solución que Enfriamiento Simulado, aunque este debería ser mejor debido debido a su probabilidad de salir de mínimos locales.

La Búsqueda Local Reiterada es una técnica similar a la Búsqueda Multiarranque Básica, pero su clara diferencia es que en vez de generar una nueva solución inicial, utiliza la mejor solución hasta el momento y aplica una mutación fuere, conservando así la localidad de la solución y posteriormente una nueva búsqueda de una solución. Ambos enunciados anteriores utilizan como técnica de búsqueda, la Búsqueda Local Primer Mejor, como vemos podemos utilizar cualquier búsqueda híbrida, es por ello que en el siguiente apartado, haremos uso del enfriamiento simulado para converger a una solución.

Como se ha comentado antes, utilizaremos una hibridación de enfriamiento simulado para la explotación y una técnica de mutación para la exploración. Observando las tablas, vemos unos resultados peores a los anteriores, la

explicación es simple, existe una restricción de 10 veces menos iteraciones para la convergencia, esto quiere decir que explora menos y converge antes. Es por ello que con 10.000 iteraciones, este algoritmo no es suficiente y por ello que obtenga peores resultados, una modificación para garantizar una mejor solución sería utilizar otro esquema de enfriamiento, menos penalizador para que pueda explorar más o por otro lado, incrementar el número de iteraciones hasta la convergencia, lo que introduciría un incremento en tiempo de cómputo.

Podemos sacar en conclusión que todas estas técnicas son buenas, además son generalmente rápidas, aunque encontramos soluciones mejores a la búsqueda local, este en realidad es muy rápido y para sistemas muy complejos que requieran Python y bajos tiempos de cómputo, sería mi opción.

Finalmente comentar que la hibridación de técnicas que exploran y explotan son técnicas muy útiles y sencillas de implementar que generalmente funcionan bien. Una de las desventajas es que pueden ser complejas y dependientes de factores, lo que implica un estudio previo de los parámetros a considerar, que a diferencia de Búsqueda Local, no. Todas las técnicas se ven beneficiadas de algún u otro modo en cómo y cuánto exploran ya que si contiene muchos puntos locales (mínimo o máximos), puede estancarse rápido, es por ello que estos algoritmos utilizan factores aleatorios para escapar de estos y buscar en otras regiones.

4. Bibliografía

1. *K. Alsabti, S. Ranka, V. Singh*. (**1998**). **An Efficient K-Means Clustering Algorithm**. "*Inthispaper, we present a novel algorithm for performing k-means clustering. It organizes all the patterns in a k-d tree structure such that one can find all the patterns which are closest to a given prototype efficiently. The main intuition behind our approach is as follows. All the prototypes are potential candidates for the closest prototype at the root level.However, for the children of the root node, we may be able to prune the candidate set by using si...*". Obtenido de https://www.cs.utexas.edu/~kuipers/readings/Alsabti-hpdm-98.pdf
2. *P. Krömer, H. Zhang, Y. Liang, J-S. P. S*. (**2018**) .**Proceedings of the Fifth Euro-China Conference on Intelligent Data Analysis**. *"This volume of Advances in Intelligent Systems and Computing highlights papers presented at the Fifth Euro-China Conference on Intelligent Data Analysis and Applications (ECC2018), held in Xi'an, China from October 12 to 14 2018. The conference was co-sponsored "* Obtenido de https://books.google.es/books?id=GzKBDwAAQBAJ&pg=PA513&lpg=PA513&dq=newthyroid+dataset+kmean&source=bl&ots=04b14UkELW&sig=ACfU3U1C4JwYT8QMCkFrC79eNn-ShPEDVA&hl=es&sa=X&ved=2ahUKEwiQjomnm67oAhWcA2MBHVgVCAwQ6AEwCnoECAkQAQ#v=onepage&q=newthyroid%20dataset%20kmean&f=false
3. *C. Blum, A. Roli* (**2003**). **Metaheuristics in Combinatorial Optimization: Overviewand Conceptual Comparison**. "*The field of metaheuristics for the application to combinatorial optimization problems isa rapidly growing field of research. This is due to the importance of combinatorialoptimization problems for the scientific as well as the industrial world...*" Obtenido de https://www.iiia.csic.es/~christian.blum/downloads/blum_roli_2003.pdf