Universidad de Granada

E.T.S Ingeniería Informática y de Telecomunicación
Grado en Ingeniería Informática
Departamento de Ciencias de la Computación e Inteligencia Artificial

Metaheurísticas - MH
Grupo A1 (Miércoles 17:30 - 19:30)
Curso 2019/2020

Práctica 3.b:
Búsquedas por Trayectorias para el Problema del Agrupamiento con Restricciones

Javier Rodríguez Rodríguez
78306251Z
e.doblerodriguez@go.ugr.es

Índice

1. Descripción del problema

El problema propuesto consiste en el agrupamiento (clustering) de instancias de datos no etiquetadas que están sujetos a un conjunto de restricciones de instancia de tipo Must-Link (ML) o Cannot-Link (CL). Estas restricciones se consideran débiles, es decir, no son limitantes a una solución, pero el cumplimiento de estas es un factor importante de calidad de la solución.

Este problema, perteneciente al campo de aprendizaje semi-supervisado, es NP-Completo, y por tanto, resulta imposible obtener un algoritmo que halle la solución óptima en un tiempo razonable. Entendiendo esta limitación, y razonando dentro del campo de la metaheurística, consideramos dos algoritmos basados en trayectorias: la Búsqueda Local (BL) con estrategia "el primer mejor", ya planteada en la Práctica 1, y el Enfriamiento Simulado (ES) que, inspirándose en la termodinámica, plantea una estrategia de búsqueda iterativa con aceptación de soluciones en función de su valor de temperatura, que cambia a lo largo de la ejecución. Estos algoritmos son capaces de llegar por sí mismos a soluciones adecuadas, pero también es posible utilizarlos como mecanismo de explotación en algoritmos más extensos, entre los cuales consideraremos la Búsqueda Multiarranque Básica (BMB) y la Búsqueda Local Iterada (ILS, por sus siglas en inglés, Iterated Local Search). Además, incorporamos nuevamente el Greedy K-medias (COPKM) para comparar los resultados obtenidos contra un algoritmo no fundamentado en metaheurísticas.

La comparación entre los algoritmos estudiados se establecerá en dos niveles distintos: la calidad de la solución y la eficiencia temporal. La calidad debe medir adecuadamente qué tan parecidos son realmente los datos que se agrupan en un mismo cluster y cuántas restricciones incumple, mientras que la eficiencia temporal debe medir el tiempo requerido para dar una solución que el criterio del algoritmo considere suficiente.

2. Aplicación de los algoritmos

Para aplicar los algoritmos descritos, primero que nada, es necesario describir las estructuras de datos que contienen toda la información. Los datos de entrada, las instancias y las restricciones, son provistas en forma de archivos en texto plano cuyos nombres identifican el conjunto de datos de donde se extrajeron, y en el caso de ser restricciones, el porcentaje de relaciones restringidas. Estos ficheros se leen y almacenan en una estructura de matriz: para las instancias, cada fila corresponde a una instancia, mientras que las columnas corresponden a las distintas dimensiones de cada instancia; para las restricciones, es una matriz cuadrada simétrica donde cada valor [i,j] $\in \{-1,0,1\}$ corresponde a la restricción entre la instancia i y la instancia j. Cada valor representa un tipo de restricción, siendo -1 una restricción CL, 1 una restricción ML y 0 ninguna restricción.

La solución del problema, para un conjunto de instancias y restricciones dado, vendrá dado en un vector de tantos elementos como instancias de datos haya, y que en cada posición contendrá el índice del cluster al que pertenece la instancia en dicha fila en la matriz de datos: es decir, Solución[i] contendrá el número de cluster al que pertenece Datos[i].

Para la medición de la calidad se emplea una función objetivo, cuyo valor ha de expresar la tasa de diferencia entre los datos de un mismo cluster y la cantidad de restricciones incumplidas, y por tanto, se ha de minimizar mientras de mayor calidad sea la solución. Utilizaremos la siguiente función

$$\textbf{Objetivo} = C_{General} + \textbf{Infeasibility} * \lambda$$

donde:

- $C_{General}$ corresponde a la desviación general de la solución, es decir, el valor medio de desviación de cada cluster. La desviación de un cluster corresponde al valor medio de las distancias entre su centroide asociado y cada una de las instancias contenidas en el cluster. Es decir:

$$(1)\ C_{General} = \sum_{i=1}^{k} C_i / k$$

$$(2)\ C_i = \sum_{j=1}^{ni} |\mu_i - x_j| / j$$

  donde k sea la cantidad de clústeres, μ_i el centroide del cluster i, x_j el dato j en el cluster y ni el número de datos en el cluster i.

- Infeasibility es un número entero que corresponde a la cantidad de restricciones incumplidas

- λ es un factor de escala, con el propósito de darle peso suficiente a la infeasibility y que se valore correctamente respecto al valor de $C_{General}$. Un valor general que se considera correcto es un valor mayor a la distancia máxima entre un par de instancias, seleccionando nosotros el techo de dicha distancia, entre la cantidad de restricciones. Es decir:

$$(3)\ \lambda = \text{techo}(D_{max}) / |R|$$

siendo |R| el número de restricciones y D_{max} la distancia máxima entre dos instancias. Importante destacar, de cara a la implementación, que en la contabilización de restricciones debemos evitar contabilizar cada restricción dos veces (debido a la simetría de la matriz)

Para la medición de la eficiencia temporal, consideraremos como métrica el tiempo de ejecución de ambos algoritmos en condiciones similares (mismo equipo bajo carga similar).

Por último, es necesario establecer, dado que ambos algoritmos contienen instrucciones estocásticas, es necesario especificarle a los algoritmos un generador de números aleatorios que, para una semilla dada, sea capaz de replicar los valores generados.

Entre estas instrucciones estocásticas, una presente en todas las técnicas analizadas es la generación de una solución inicial, totalmente aleatoria. Sin embargo, esta solución ha de cumplir con la regla de validez de toda solución: que ningún clúster quede vacío.

Para ello, la estrategia es la siguiente

Solución Inicial(k, tamaño):
# k es la cantidad de clústeres
# i_j es un valor aleatorio ∈ {0, 1, …, k-1}
Solución = {0, 1, …, k-1} ∪ {i_k, i_{k+1}, …, $i_{tamaño-1}$}
Solución = Barajar(Solución)
Retornar Solución

Este pequeño trozo de código asegura que todos los clústeres tengan por lo menos un elemento asignando, de partida, los primeros k elementos a cada clúster respectivamente, y por tanto asignando un elemento a cada clúster. El resto de tamaño - k elementos se asignan aleatoriamente entre los valores de cada clúster, sin ningún tipo de cuidado adicional, pues ya aseguramos la condición de validez. Finalmente, para evitar que las primeras k asignaciones de la solución sean constantes en cada llamada, aplicamos una operación de barajado sobre el vector, asegurando así verdadera aleatoriedad sin perder la certeza de validez.

3. Métodos de búsqueda

Hemos de contemplar cuatro métodos de búsqueda: dos “simples” y dos extensiones sobre éstos. Los métodos de búsqueda simples son Enfriamiento Simulado y Búsqueda Local.

Enfriamiento Simulado: esta técnica, inspirada en procesos termodinámicos, consiste en la exploración de una solución aleatoria inicial, pero cuyo criterio de aceptación está guiado por un parámetro adicional, la temperatura, que inicialmente permite, con el objetivo de evitar óptimos locales, explorar soluciones peores a la actual. Sin embargo, para permitir la convergencia, este parámetro de flexibilidad disminuye en cada iteración hasta no permitir ninguna solución peor.

Su mecanismo de búsqueda viene descrito por el siguiente pseudocódigo:

```
Enfriamiento Simulado(datos, restricciones, k, λ, μ, Φ, T_F, máx_evaluaciones):
Solución = Solución inicial(k, datos.longitud)
Mejor = Solución
# Cálculo de la temperatura inicial
T_0 = μ * Objetivo_Solución / -ln(Φ)
Mientras (T_0 ≤ T_F):
     T_F = T_F / 10 # Esto asegura que T_F será siempre menor que T_0

máx_vecinos = datos.longitud * 10
máx_éxitos = 0,1 * máx_vecinos
máx_iteraciones = máx_evaluaciones / máx_vecinos
β = (T_0 - T_F) / (máx_iteraciones * T_0 * T_F)
T = T_0
# Esquema de exploración
Mientras (T > T_F):
     Para i = 0 : máx_vecinos - 1:
          un_solo_elem = {clústeres que solo tienen un elemento en
Solución}
          cambiar = {Índice aleatorio de Solución que no esté en
un_solo_elem} # Pues de cambiar este, quedaría un clúster vacío
          cambia_por = {Valor aleatorio en 0..k-1 ≠
Solución[cambiar]}
# Para evitar que el vecino sea igual a la solución actual
          Vecino = Copia(Solución)
          Vecino[cambiar] = cambia_por

          Δf = Objetivo_Vecino - Objetivo_Solución
          U = Valor aleatorio en 0…1
          Si (Δf < 0 ∨ U ≤ exp(-Δf/T_k):
               Solución = Vecino
               Si (Objetivo_Solución < Objetivo_Mejor):
                    Mejor = Solución
```

```
     # Esquema de enfriamiento (Cauchy Modificado)
     T = T / (1 + β*T)
Devolver Solución
```

Este pseudocódigo consiste en tres fragmentos principales: cálculo de la solución inicial, temperatura inicial y establecimiento de constantes, exploración de vecindario y esquema de enfriamiento. Los valores de determinadas constantes se especifican como parámetros, y el esquema utilizado para el manejo de la temperatura es el Cauchy Modificado.

Búsqueda Local: a pesar de haber analizado esta técnica en la Práctica 1, su uso en otras técnicas de interés para esta práctica hace conveniente un repaso sobre su método de búsqueda: explorar el vecindario de una solución aleatoria y, bajo estrategia "el primer mejor", ir reemplazando la solución por un vecino y repetir el proceso hasta no encontrar ninguna solución mejor en el vecindario, momento en el que encontramos un óptimo local y paramos.

En pseudocódigo, el método se describe así

```
Búsqueda Local(datos, restricciones, k,λ, máx_iteraciones):
Solución = Solución Inicial(k, datos.longitud)
Para i = 0 : máx_iteraciones - 1:

     # Generación de vecindario virtual
     un_solo_elem = {clústeres que solo tienen un elemento en
Solución}
     elementos_posibles = {clústeres que no estén en un_solo_elem}
     Para j = 0 : k - 1:
          Posibles = {Elementos p de elementos_posibles tales que
Solución[p] ≠ j}
     # Esto evita que el vecino sea la solución
          Vecindario = Vecindario ∪ {Posibles x j}
     Vecindario = Barajar(Vecindario)

     Para índice, clúster en Vecindario:
          Vecino = Copia(Solución)
          Vecino[índice] = clúster
          Si Objetivo_Vecino < Objetivo_Solución:
               Solución = Vecino
               Mejor = False
               Salir del ciclo
     Si !Mejor:
          Salir del ciclo
Devolver Solución
```

Este pseudocódigo describe el método de búsqueda. Importante destacar el uso de un vecindario virtual (pares índice/cambio) en vez de soluciones enteras,

con importantes beneficios en cuanto a espacio. La creación de este vecindario es lo más complejo del algoritmo, pues requiere el cuidado de seleccionar solamente aquellos pares tales que no modifiquen un elemento que sea el único de su clúster (pues al cambiarlo dejaría el clúster vacío) y que el cambio no sea al clúster en donde se encuentra actualmente (pues en ese caso el vecino sería la misma solución).

Como el vecindario se construye siguiendo el orden de los clústeres, pero la exploración de éste ha de ser aleatoria, lo barajamos antes. La exploración en sí es sencillamente la creación del vecino a partir del vecino virtual, su evaluación y comparación con la solución actual. Si es mejor, la reemplaza, y si no, se desecha y se sigue explorando. Si no se consigue ningún vecino mejor en todo el vecindario, detenemos la ejecución, pues alcanzamos ya un óptimo local.

Búsqueda Multiarranque Básica: esta técnica no es más que una extensión sencilla de la Búsqueda Local, donde se conserva la mejor de un número determinado de ejecuciones de la BL.

En pseudocódigo resulta

```
Búsqueda Multiarranque Básica(datos, restricciones, k, λ, máx_iteraciones, nro_inicios):
Objetivo_Mejor = ∞
Para i = 0 : nro_inicios - 1:
     Actual = Búsqueda Local(datos, restricciones, k, λ, máx_iteraciones)
     Si (Objetivo_Actual < Objetivo_Mejor):
          Mejor = Actual
Retornar Mejor
```

Como se ve, el mecanismo de búsqueda es sencillamente la ejecución de múltiples BL, de forma que experimentemos con diversos puntos de partida, y al final devolvemos la mejor de todas esas búsquedas.

Iterated Local Search: la última de las técnicas estudiadas, ILS es una extensión sobre las técnicas de búsqueda explicadas antes, que al igual que BMB ejecuta un algoritmo de búsqueda varias veces, pero con la diferencia de que en vez de ser soluciones iniciales aleatorias en cada iteración, a partir de la segunda iteración parte de una mutación fuerte de la mejor solución explorada hasta ese momento. En esta técnica se nos pide usar tanto BL como ES, así que en el pseudocódigo debemos contemplar ambos escenarios:

```
Iterated Local Search(datos, restricciones, k, λ, máx_iteraciones, algoritmo_búsqueda):
Tamaño_segmento = Redondear(datos.longitud * 0,1)
Solución = Solución Inicial(k, datos.longitud)
```

```
Solución = algoritmo_búsqueda(datos, restricciones, k, λ, máx_iteraciones, Solución)
# Puede ser Búsqueda Local o Enfriamiento Simulado

# Operaciones de mutación
Para i = 0 : máx_iteraciones - 1:
     Mutación = Copia(Solución)
     Inicio_segmento = Número aleatorio en 0…datos.longitud
     Segmento = {Inicio_segmento, Inicio_segmento+1 % datos.longitud,..., Inicio_segmento+Tamaño_segmento % datos.longitud}
     # % representa la operación módulo

     Resto = {Índices de Mutación que no están en Segmento}
     Clústeres_faltantes = {Clústeres que no están en Mutación[Resto]}
     # Nos aseguramos de restablecer los clústeres que dejamos
     #vacíos
     Segmento_mutado = Clústeres_faltantes ∪ {Tamaño_segmento - Clústeres_faltantes.longitud valores aleatorios en 0… k-1}
     Segmento_mutado = Barajar(Segmento_mutado)
     # Reestablecemos aleatoriedad
     Mutación[Segmento] = Segmento_mutado

     Mutación = algoritmo_búsqueda(datos, restricciones, k, λ, máx_iteraciones, Mutación)
     Si (Objetivo_Mutación < Objetivo_Solución):
          Solución = Mutación
Devolver Solución
```

Si bien en concepto puede resultar similar a la BMB, en cuanto a que es una ejecución reiterada de un algoritmo de búsqueda por trayectorias y conservando el mejor resultado, el ILS incorpora mutaciones para determinar la solución sobre la que se ejecuta el algoritmo de búsqueda. Esta mutación ha de ser fuerte para explorar adecuadamente; en nuestro caso, del 10% de la solución. Para realizar tal mutación se selecciona aleatoriamente el punto de inicio del segmento a mutar y se determinan los índices que conforman tal segmento. Luego, para asegurar que la mutación de una solución válida, debemos evitar vaciar algún clúster, es decir, que al mutar el segmento se pierdan los elementos que conformaban algún clúster. Para ello, observamos qué clústeres no tienen ningún elemento en el resto de la solución, y por ende, quedan vacíos sin el segmento, sobre los cuales utilizaremos una estrategia similar a la de creación de soluciones iniciales: los incorporamos al segmento mutado forzosamente, asegurando así que todos los clústeres tienen al menos un elemento, y el resto del segmento lo construimos aleatoriamente. Finalmente, para reestablecer la aleatoriedad, barajamos el segmento mutado y lo reemplazamos en la solución.

4. Algoritmos de comparación

No se incorpora ningún algoritmo de comparación nuevo. El análisis de los datos se hace entre las distintas estrategias de búsqueda por trayectorias y el algoritmo Greedy K-Medias (COPKM) analizado en la práctica 1: un algoritmo Greedy donde la solución es construida seleccionando, para cada elemento de la solución, el clúster más cercano a ésta de entre aquellos que violen la menor cantidad de restricciones al asignarse.

Un pseudocódigo para este algoritmo es

```
COPKM(datos, restricciones, k, λ):
Mínimos = {Mínimo(datos) para cada dimensión}
Máximos = {Máximo(datos) para cada dimensión}
Centroides = {k vectores de valores aleatorios en Mínimos,…, Máximos
para cada dimensión}
Recorrido = Barajar(0… datos.longitud - 1)
Anteriores = []
Solución = []
Mientras (Centroides ≠ Anteriores):
     Para i ∈ Recorrido:
          Considerados = {Valores j en 0… k-1 tales que
Infeasibility sea mínima si Solución[i] = j}
          Clúster = {Valor j en Considerados tal que
Distancia(Centroides[j], i) sea mínima}
          Solución[i] = Clúster
     Para i = 0… k-1:
          Centroides[i] = Media(Valores j tal que Solución[j] = i)
     Si (Centroides ≠ Anteriores):
          Anteriores = Centroides
Devolver Solución
```

Si bien describimos el procedimiento de forma sencilla, este algoritmo no está implementado en la entrega, y solo lo mencionamos para usar los resultados de los experimentos con éste de la Práctica 1 para el análisis de resultados.

5. Procedimiento para desarrollar la práctica:

Todo el proceso de procesamiento de datos, implementación de algoritmos y obtención, cálculo y almacenamiento de los resultados finales fue hecho en el lenguaje de programación Python 3.7.6, a través de su distribución especializada para ciencia de datos, Anaconda. La razón de la elección de este lenguaje es por ser uno de los más utilizados y destacados en problemas de aprendizaje, por lo que se consideró una buena oportunidad para practicar su uso y sintaxis, y en particular, el uso relacionado al paquete de cómputo científico NumPy, una de las principales ventajas del lenguaje contra sus competidores y la principal herramienta detrás de todas las implementaciones.

Las ejecuciones fueron realizadas sobre una distribución Ubuntu 18.04 contenida en Windows Subsystem for Linux (WSL), en un equipo con sistema operativo Windows 10 1909.

El código proporcionado se decidió dejar de lado, pues no se consideró necesario implementar el código en Python cuando NumPy provee las herramientas de randomización necesarias.

El proceso de desarrollo de la práctica se hizo de forma lineal, aprovechando los mecanismos ya implementados en las prácticas anteriores: los fragmentos de código para la lectura y ejecución de los datos, así como de evaluación de la función objetivo los tomamos directamente de las prácticas anteriores, luego implementamos la búsqueda local optimizando ligeramente el código de la práctica 1, y finalmente implementamos los nuevos algoritmos en orden de aparición en la documentación, depurando cada uno según se terminaba.

- **Manual de ejecución:**

Es requisito necesario para la ejecución de la práctica que el equipo donde se ejecuten contenga Python 3.4 en adelante, así como el paquete NumPy 1.17. Además, el fichero P01b.py debe estar en el mismo directorio que un directorio llamado “Instancias y Tablas PAR 2019-20”, que contenga los ficheros de datos y restricciones. Estos ficheros deben ser los contenidos en el fichero comprimido “nuevos conjuntos de datos PAR 2019-20.zip” disponible en la plataforma PRADO.

Una vez se tengan todos los requisitos, tan solo es necesario indicar por terminal la ejecución del script, mediante la sintaxis

> <path a Python> P03b.py

Durante la ejecución no imprimirá información por pantalla, pues todos los resultados estarán contenidos en el fichero “solutions_P03b.txt” que se creará en el mismo directorio que contiene el script.

6. Experimentos y análisis de resultados:

Una ejecución del programa siguiendo las instrucciones anteriores genera la siguientes tablas de resultados. En todos los casos las semillas correspondientes a cada ejecución son 1, 112, 241, 27, 472, y el tiempo viene expresado en segundos.

Resultados obtenidos por el algoritmo Enfriamiento Simulado en el PAR con 10% de restricciones

	Iris				**Ecoli**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0.67	0.00	0.67	2.29	22.24	74.00	24.23	19.18
Ejecución 2	0.67	0.00	0.67	2.74	22.17	74.00	24.15	12.57
Ejecución 3	0.67	0.00	0.67	3.32	21.81	96.00	24.39	9.52
Ejecución 4	0.67	0.00	0.67	2.04	21.24	127.00	24.65	14.65
Ejecuión 5	0.67	0.00	0.67	2.01	22.18	67.00	23.98	10.89
Media	0.67	0.00	0.67	2.48	21.93	87.60	24.28	13.36

	Rand				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0.72	0.00	0.72	2.59	10.89	96.00	14.42	5.83
Ejecución 2	0.72	0.00	0.72	1.09	10.90	97.00	14.48	5.55
Ejecución 3	0.72	0.00	0.72	1.70	13.83	6.00	14.06	5.52
Ejecución 4	0.72	0.00	0.72	1.76	13.83	6.00	14.06	5.40
Ejecuión 5	0.72	0.00	0.72	1.30	13.83	6.00	14.06	5.76
Media	0.72	0.00	0.72	1.69	12.66	42.20	14.21	5.61

Resultados obtenidos por el algoritmo Enfriamiento Simulado en el PAR con 20% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.67	0.00	0.67	3.75	21.91	171.00	24.20	45.95
Ejecución 2	0.67	0.00	0.67	4.26	21.91	172.00	24.21	38.79
Ejecución 3	0.67	0.00	0.67	5.76	21.89	181.00	24.31	36.21
Ejecución 4	0.67	0.00	0.67	4.43	21.88	160.00	24.03	45.92
Ejecuión 5	0.67	0.00	0.67	4.59	21.89	150.00	23.90	48.17
Media	0.67	0.00	0.67	4.56	21.89	166.80	24.13	43.01

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.72	0.00	0.72	1.69	10.90	235.00	15.24	6.02
Ejecución 2	0.72	0.00	0.72	2.83	14.29	0.00	14.29	18.91
Ejecución 3	0.72	0.00	0.72	1.77	10.82	268.00	15.77	5.61
Ejecución 4	0.72	0.00	0.72	2.69	14.29	0.00	14.29	21.75
Ejecuión 5	0.72	0.00	0.72	2.36	14.29	0.00	14.29	25.74
Media	0.72	0.00	0.72	2.27	12.92	100.60	14.77	15.60

Resultados obtenidos por el algoritmo Búsqueda Multiarranque Básica en el PAR con 10% de restricciones

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0.67	0.00	0.67	3.71	21.54	176.00	26.26	54.79
Ejecución 2	0.67	0.00	0.67	3.62	21.66	151.00	25.71	54.48
Ejecución 3	0.67	0.00	0.67	3.87	21.87	146.00	25.78	55.18
Ejecución 4	0.67	0.00	0.67	3.89	21.29	232.00	27.52	57.18
Ejecuión 5	0.67	0.00	0.67	3.76	21.77	107.00	24.64	59.79
Media	0.67	0.00	0.67	3.77	21.63	162.40	25.98	56.28

	Rand				Newthyroid			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0.72	0.00	0.72	3.25	13.83	6.00	14.06	6.51
Ejecución 2	0.72	0.00	0.72	3.37	13.83	6.00	14.06	6.22
Ejecución 3	0.72	0.00	0.72	3.27	13.83	6.00	14.06	7.15
Ejecución 4	0.72	0.00	0.72	3.22	13.83	6.00	14.06	7.95
Ejecuión 5	0.72	0.00	0.72	3.35	13.83	6.00	14.06	8.84
Media	0.72	0.00	0.72	3.29	13.83	6.00	14.06	7.33

Resultados obtenidos por el algoritmo Búsqueda Multiarranque Básica en el PAR con 20% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.67	0.00	0.67	3.32	22.21	263.00	25.73	56.58
Ejecución 2	0.67	0.00	0.67	3.37	22.66	231.00	25.76	58.78
Ejecución 3	0.67	0.00	0.67	3.28	22.00	148.00	23.99	58.64
Ejecución 4	0.67	0.00	0.67	3.33	22.28	153.00	24.34	54.92
Ejecuión 5	0.67	0.00	0.67	3.15	22.11	246.00	25.41	57.30
Media	0.67	0.00	0.67	3.29	22.25	208.20	25.05	57.24

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.72	0.00	0.72	3.33	14.29	0.00	14.29	6.32
Ejecución 2	0.72	0.00	0.72	3.21	14.29	0.00	14.29	6.48
Ejecución 3	0.72	0.00	0.72	3.16	14.29	0.00	14.29	6.74
Ejecución 4	0.72	0.00	0.72	3.29	14.29	0.00	14.29	7.85
Ejecuión 5	0.72	0.00	0.72	3.03	14.29	0.00	14.29	6.64
Media	0.72	0.00	0.72	3.20	14.29	0.00	14.29	6.81

Resultados obtenidos por el algoritmo Iterative Local Search en el PAR con 10% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.67	0.00	0.67	1.77	21.69	76.00	23.73	44.50
Ejecución 2	0.67	0.00	0.67	1.82	21.23	77.00	23.30	45.56
Ejecución 3	0.67	0.00	0.67	1.82	21.23	78.00	23.32	45.45
Ejecución 4	0.67	0.00	0.67	2.13	21.35	99.00	24.00	44.02
Ejecuión 5	0.67	0.00	0.67	1.60	19.14	143.00	22.97	44.83
Media	0.67	0.00	0.67	1.83	20.93	94.60	23.47	44.87

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.72	0.00	0.72	1.81	10.89	95.00	14.40	4.50
Ejecución 2	0.72	0.00	0.72	1.68	13.83	6.00	14.06	3.44
Ejecución 3	0.72	0.00	0.72	1.78	10.89	95.00	14.40	5.06
Ejecución 4	0.72	0.00	0.72	1.91	13.83	6.00	14.06	4.24
Ejecuión 5	0.72	0.00	0.72	1.71	10.89	95.00	14.40	5.33
Media	0.72	0.00	0.72	1.77	12.07	59.40	14.27	4.52

Resultados obtenidos por el algoritmo Iterative Local Search en el PAR con 20% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.67	0.00	0.67	1.71	21.75	145.00	23.69	50.66
Ejecución 2	0.67	0.00	0.67	2.41	21.84	115.00	23.39	47.94
Ejecución 3	0.67	0.00	0.67	2.48	21.82	149.00	23.82	48.68
Ejecución 4	0.67	0.00	0.67	2.20	21.97	132.00	23.74	49.19
Ejecuión 5	0.67	0.00	0.67	2.13	21.55	211.00	24.38	50.39
Media	0.67	0.00	0.67	2.19	21.79	150.40	23.80	49.37

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.72	0.00	0.72	1.84	10.87	231.00	15.14	4.37
Ejecución 2	0.72	0.00	0.72	1.68	14.29	0.00	14.29	3.94
Ejecución 3	0.72	0.00	0.72	1.62	10.89	233.00	15.19	3.39
Ejecución 4	0.72	0.00	0.72	1.73	14.29	0.00	14.29	3.48
Ejecuión 5	0.72	0.00	0.72	1.56	14.29	0.00	14.29	3.30
Media	0.72	0.00	0.72	1.68	12.92	92.80	14.64	3.70

Resultados obtenidos por el algoritmo ILS-ES en el PAR con 10% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.67	0.00	0.67	17.16	44.92	1746.00	91.76	12.16
Ejecución 2	0.67	0.00	0.67	17.58	45.50	1709.00	91.36	13.15
Ejecución 3	0.67	0.00	0.67	17.21	45.35	1718.00	91.45	12.42
Ejecución 4	0.67	0.00	0.67	17.13	45.45	1740.00	92.13	11.63
Ejecuión 5	0.67	0.00	0.67	19.72	45.57	1722.00	91.77	11.03
Media	0.67	0.00	0.67	17.76	45.36	1727.00	91.69	12.08

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.72	0.00	0.72	16.73	13.09	1100.00	53.74	2.95
Ejecución 2	0.72	0.00	0.72	15.95	13.11	1086.00	53.25	2.84
Ejecución 3	0.72	0.00	0.72	15.90	13.10	1088.00	53.31	3.15
Ejecución 4	0.72	0.00	0.72	15.80	13.16	1078.00	53.00	3.17
Ejecuión 5	0.72	0.00	0.72	15.61	13.13	1074.00	52.82	3.40
Media	0.72	0.00	0.72	16.00	13.12	1085.20	53.22	3.10

Resultados obtenidos por el algoritmo ILS-ES en el PAR con 20% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.67	0.00	0.67	17.71	45.47	3514.00	92.61	12.36
Ejecución 2	0.67	0.00	0.67	16.92	45.58	3508.00	92.64	13.14
Ejecución 3	0.67	0.00	0.67	17.54	45.64	3514.00	92.78	13.28
Ejecución 4	0.67	0.00	0.67	17.09	45.08	3536.00	92.51	12.45
Ejecuión 5	0.67	0.00	0.67	18.37	45.17	3488.00	91.96	13.03
Media	0.67	0.00	0.67	17.53	45.39	3512.00	92.50	12.85

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.72	0.00	0.72	16.96	12.77	2213.00	53.65	2.95
Ejecución 2	0.72	0.00	0.72	16.31	12.89	2209.00	53.70	2.84
Ejecución 3	0.72	0.00	0.72	17.07	12.87	2221.00	53.90	3.15
Ejecución 4	0.72	0.00	0.72	16.95	13.31	2201.00	53.97	3.17
Ejecuión 5	0.72	0.00	0.72	16.56	13.08	2230.00	54.28	3.40
Media	0.72	0.00	0.72	16.77	12.98	2214.80	53.90	3.10

Resultados obtenidos por el algoritmo Búsqueda Local[1] en el PAR con 10% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.67	0.00	0.67	0.33	22.03	52.00	23.43	15.87
Ejecución 2	0.67	0.00	0.67	0.41	23.91	104.00	26.70	15.68
Ejecución 3	0.67	0.00	0.67	0.53	21.94	71.00	23.84	20.42
Ejecución 4	0.67	0.00	0.67	0.64	21.33	118.00	24.49	15.97
Ejecuión 5	0.67	0.00	0.67	0.39	21.28	110.00	24.23	16.43
Media	0.67	0.00	0.67	0.46	22.10	91.00	24.54	16.87

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.72	0.00	0.72	0.36	10.90	99.00	14.55	0.95
Ejecución 2	0.72	0.00	0.72	0.40	10.90	99.00	14.55	1.09
Ejecución 3	0.72	0.00	0.72	0.36	10.88	97.00	14.46	1.12
Ejecución 4	0.72	0.00	0.72	0.36	10.88	97.00	14.46	1.50
Ejecuión 5	0.72	0.00	0.72	0.46	13.83	6.00	14.06	0.78
Media	0.72	0.00	0.72	0.39	11.48	79.60	14.42	1.09

Resultados obtenidos por el algoritmo Búsqueda Local[1] en el PAR con 20% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.67	0.00	0.67	0.39	21.91	161.00	24.07	12.57
Ejecución 2	0.67	0.00	0.67	0.37	21.94	163.00	24.12	14.35
Ejecución 3	0.67	0.00	0.67	0.34	21.88	178.00	24.27	15.30
Ejecución 4	0.67	0.00	0.67	0.40	21.93	166.00	24.16	18.08
Ejecuión 5	0.67	0.00	0.67	0.35	21.94	127.00	23.64	15.80
Media	0.67	0.00	0.67	0.37	21.92	159.00	24.05	15.22

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.72	0.00	0.72	0.30	14.29	0.00	14.29	0.91
Ejecución 2	0.72	0.00	0.72	0.44	10.81	260.00	15.61	0.90
Ejecución 3	0.72	0.00	0.72	0.45	14.29	0.00	14.29	1.37
Ejecución 4	0.72	0.00	0.72	0.37	14.29	0.00	14.29	1.17
Ejecuión 5	0.72	0.00	0.72	0.36	14.29	0.00	14.29	0.81
Media	0.72	0.00	0.72	0.38	13.59	52.00	14.55	1.03

[1] Los resultados son distintos a los de la Práctica 1 porque en aquella implementación no se contempló la cota de máximo de iteraciones. Para esta ejecución sí se consideró, modificando ligeramente la implementación. El resto del algoritmo permanece igual entre ambas prácticas

Resultados obtenidos por el algoritmo Greedy COPKM en el PAR con 10% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.67	11	0.75	0.02	39.87	373	49.87	1.86
Ejecución 2	0.67	4	0.69	0.02	36.31	374	46.34	0.16
Ejecución 3	0.67	0	0.67	0.02	37.54	291	45.35	3.39
Ejecución 4	0.67	7	0.72	0.02	39.26	341	48.41	0.14
Ejecuión 5	0.67	0	0.67	0.02	35.16	77	37.22	0.12
Media	0.67	4.4	0.7	0.02	37.63	291.2	45.44	1.13

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.72	0	0.72	0.02	15.35	53	17.31	0.03
Ejecución 2	0.71	9	0.78	0.02	16.48	101	20.21	0.04
Ejecución 3	0.72	0	0.72	0.03	14.15	54	16.14	0.04
Ejecución 4	0.72	0	0.72	0.02	13.83	70	16.41	0.04
Ejecuión 5	0.72	0	0.72	0.01	15.12	114	19.33	0.03
Media	0.71	1.8	0.73	0.02	14.98	78.4	17.88	0.04

Resultados obtenidos por el algoritmo Greedy COPKM en el PAR con 20% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.67	10.00	0.71	0.03	39.73	268.00	43.32	0.14
Ejecución 2	0.67	0.00	0.67	0.02	34.75	86.00	35.90	0.18
Ejecución 3	0.67	21.00	0.74	0.02	36.36	315.00	40.58	0.12
Ejecución 4	0.67	0.00	0.67	0.02	38.99	262.00	42.50	0.99
Ejecuión 5	0.67	0.00	0.67	0.03	36.75	183.00	39.20	0.08
Media	0.67	6.20	0.69	0.02	37.31	222.80	40.30	0.30

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.72	0.00	0.72	0.02	14.29	0.00	14.29	0.03
Ejecución 2	0.72	0.00	0.72	0.02	14.10	76.00	15.51	0.03
Ejecución 3	0.72	0.00	0.72	0.02	15.20	567.00	25.68	0.03
Ejecución 4	0.72	0.00	0.72	0.01	14.78	106.00	16.74	0.03
Ejecuión 5	0.72	0.00	0.72	0.01	14.18	102.00	16.07	0.04
Media	0.72	0.00	0.72	0.02	14.51	170.20	17.66	0.03

Resultados obtenidos por el algoritmo ILS-ES adaptado en el PAR con 10% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.67	0.00	0.67	4.41	22.47	81.00	24.64	38.72
Ejecución 2	0.67	0.00	0.67	4.76	21.25	157.00	25.47	38.39
Ejecución 3	0.67	0.00	0.67	6.19	21.85	105.00	24.66	38.69
Ejecución 4	0.67	0.00	0.67	7.42	21.26	117.00	24.40	36.59
Ejecuión 5	0.67	0.00	0.67	4.16	21.35	116.00	24.46	34.82
Media	0.67	0.00	0.67	5.39	21.64	115.20	24.73	37.44

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.72	0.00	0.72	3.24	10.90	97.00	14.48	5.55
Ejecución 2	0.72	0.00	0.72	3.31	10.90	97.00	14.48	5.54
Ejecución 3	0.72	0.00	0.72	3.48	10.88	97.00	14.46	6.09
Ejecución 4	0.72	0.00	0.72	3.22	13.83	6.00	14.06	4.44
Ejecuión 5	0.72	0.00	0.72	3.32	10.90	99.00	14.55	4.88
Media	0.72	0.00	0.72	3.31	11.48	79.20	14.41	5.30

Resultados obtenidos por el algoritmo ILS-ES adaptado en el PAR con 20% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.67	0.00	0.67	4.06	22.50	177.00	24.87	38.72
Ejecución 2	0.67	0.00	0.67	4.95	22.32	139.00	24.18	38.39
Ejecución 3	0.67	0.00	0.67	4.91	21.89	184.00	24.36	38.69
Ejecución 4	0.67	0.00	0.67	3.45	22.30	137.00	24.14	36.59
Ejecuión 5	0.67	0.00	0.67	4.08	21.96	195.00	24.58	34.82
Media	0.67	0.00	0.67	4.29	22.19	166.40	24.43	37.44

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
Ejecución 1	0.72	0.00	0.72	3.29	14.29	0.00	14.29	5.73
Ejecución 2	0.72	0.00	0.72	2.93	14.29	0.00	14.29	4.74
Ejecución 3	0.72	0.00	0.72	3.26	14.29	0.00	14.29	4.98
Ejecución 4	0.72	0.00	0.72	4.47	14.29	0.00	14.29	5.49
Ejecuión 5	0.72	0.00	0.72	4.26	14.29	0.00	14.29	5.22
Media	0.72	0.00	0.72	3.64	14.29	0.00	14.29	5.23

Resultados globales en el PAR con 10% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
BL	0.67	0.00	0.67	0.46	22.10	91.00	24.54	16.87
ES	0.67	0.00	0.67	2.48	21.93	87.60	24.28	13.36
BMB	0.67	0.00	0.67	3.77	21.63	162.40	25.98	56.28
ILS	0.67	0.00	0.67	1.83	20.93	94.60	23.47	44.87
ILS-ES	0.67	0.00	0.67	17.76	45.36	1727.00	91.69	12.08
ILS-ES Adapt.	0.67	0.00	0.67	5.39	21.64	115.20	24.73	37.44
COPKM	0.67	4.40	0.70	0.02	37.63	291.20	45.44	1.13

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
BL	0.72	0.00	0.72	0.39	11.48	79.60	14.42	1.09
ES	0.72	0.00	0.72	1.69	12.66	42.20	14.21	5.61
BMB	0.72	0.00	0.72	3.29	13.83	6.00	14.06	7.33
ILS	0.72	0.00	0.72	1.77	12.07	59.40	14.27	4.52
ILS-ES	0.72	0.00	0.72	16.00	13.12	1085.20	53.22	3.10
ILS-ES Adapt.	0.72	0.00	0.72	3.31	11.48	79.20	14.41	5.30
COPKM	0.71	1.80	0.73	0.02	14.98	78.40	17.88	0.04

Resultados globales en el PAR con 20% de restricciones

	Iris				Ecoli			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
BL	0.67	0.00	0.67	0.37	21.92	159.00	24.05	15.22
ES	0.67	0.00	0.67	4.56	21.89	166.80	24.13	43.01
BMB	0.67	0.00	0.67	3.29	22.25	208.20	25.05	57.24
ILS	0.67	0.00	0.67	2.19	21.79	150.40	23.80	49.37
ILS-ES	0.67	0.00	0.67	17.53	45.39	3512.00	92.50	12.85
ILS-ES Adapt.	0.67	0.00	0.67	4.29	22.19	166.40	24.43	37.44
COPKM	0.67	6.20	0.69	0.02	37.31	222.80	40.30	0.30

	Rand				Newthyroid			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
BL	0.72	0.00	0.72	0.38	13.59	52.00	14.55	1.03
ES	0.72	0.00	0.72	2.27	12.92	100.60	14.77	15.60
BMB	0.72	0.00	0.72	3.20	14.29	0.00	14.29	6.81
ILS	0.72	0.00	0.72	1.68	12.92	92.80	14.64	3.70
ILS-ES	0.72	0.00	0.72	16.77	12.98	2214.80	53.90	3.10
ILS-ES Adapt.	0.72	0.00	0.72	3.64	14.29	0.00	14.29	5.23
COPKM	0.72	0.00	0.72	0.02	14.51	170.20	17.66	0.03

Los resultados obtenidos han de ser analizados desde dos puntos de vista: la calidad de la solución y el tiempo empleado. Determinar si un algoritmo es mejor que otro vendrá de cómo estén balanceados sus resultados en ambas dimensiones.

Desde el punto de vista de la calidad de la solución, al analizar los datos notamos que los resultados son homogéneos en los sets Iris y Rand, producto de su simplicidad. Por tanto, no nos detendremos a comentar sobre ellos, tan solo son una prueba de suficiencia: si el algoritmo da el resultado óptimo en estos sets, es suficientemente bueno. Como vemos, para todos los algoritmos, a excepción de COPKM, que no está basado en metaheurísticas, esto ocurre, por lo que no tenemos información para comparar sus rendimientos en cuanto a calidad.

De Newthyroid y, sobre todo, Ecoli, se puede concluir mucho más respecto a la calidad relativa de los algoritmos. Todos los algoritmos metaheurísticos dan resultados de la función objetivo en el mismo rango numérico (~24 en Ecoli y ~14 en Newthyroid) con una clara excepción: ILS-ES, cuyos resultados son notablemente peores a los del resto, siendo incluso mucho peores que los de COPKM. Analizando las posibles causas de tal situación encontramos con que el algoritmo enfría demasiado rápido, por lo que las soluciones (malas) que contempla al inicio permanecen y son devueltas al enfriar del todo antes de contemplar mejores soluciones. Por esta razón, realizamos una segunda ejecución del algoritmo pero con el parámetro de máxima cantidad de vecino adaptado: lo dividimos entre 10. Los resultados con esta adaptación son considerablemente mejores, similares al resto de algoritmos. Sin embargo, es bastante posible que con una adaptación más completa de los parámetros pudiésemos mejorar aún más los resultados.

Otro detalle de interés es que los resultados de la BL son mejores que los de BMB en Ecoli. La razón de esto es que, en la ejecución individual de BL, la cota de evaluaciones es de 100000, mientras que en BMB cada BL tiene un máximo de 10000 evaluaciones. En Ecoli, por la complejidad de los datos, se aprovecha más recorrer a mayor profundidad el vecindario que probar con distintos puntos de partida. Sin embargo, el planteameniento de BMB no es que resulte peor que una BL con más evaluaciones, pues en Newthyroid, que son unos datos más sencillos, sí que llega a mejores resultados. De hecho, si consideramos los resultados de ILS también, podemos razonar sobre la forma de los datos. ILS da los mejores resultados para Ecoli, pero en Newthyroid son el ILS-ES adaptado y BMB los mejores. Podemos predecir que esto es porque Newthyroid es un conjunto de datos con gran variedad de óptimos locales, y se saca mucho beneficio de la exploración de diversos puntos de partida (la estrategia de BMB) o incorporar estrategias que le hagan contemplar soluciones peores para no estancarse en dichos óptimos (el mecanismo de ES y de mutación de ILS-ES adaptado). Mientras tanto, Ecoli es una función con menos óptimos locales pero que ha de explorar un espacio de soluciones muy extenso, por lo que aquellos mecanismos que exploren con más detalle el vecindario (la BL al tener más iteraciones o ILS al ejecutar reiteradamente sobre mutaciones de la solución) llegan a mejores soluciones.

En cuanto a la eficiencia temporal, claramente COPKM es mucho mejor que sus contrapartes metaheurísticas, pero su pérdida de calidad aún en los conjuntos

sencillos hacen que pierda interés como algoritmo. De los metaheurísticos básicos, BL es mucho más eficiente en tiempo que ES, cosa que no es sorprendente considerando que este último recorre un espacio más amplio al considerar soluciones peores. BMB, al ser una ejecución reiterada de BL (aunque limitada a menos iteraciones), supera el tiempo de BL en todos los casos. Sin embargo, comparada con ES vemos que, si bien ES suele ser más rápida, hay casos donde no ocurre, probablemente porque las búsquedas locales convergen rápidamente y la exploración del ES, aún siendo una sola contra 10 iteraciones, demande más tiempo.

El mecanismo de mutación de ILS, si bien son operaciones adicionales, parece que permite a la ejecución global converger más rápido, siendo mejor que BMB en todas las ejecuciones e incluso superando a ES en cuanto a eficiencia en los conjuntos sencillos, debido a que la convergencia en éstos es rápida y partir de mutaciones lo restringe a un espacio de búsqueda menor que otro punto de partida aleatorio.

Por último, llama la atención el comportamiento temporal de ILS-ES: en los conjuntos sencillos es con diferencia el algoritmo que más tarda, mientras que en los conjuntos complejos llega a tardar menos que una BL (aunque dando terribles resultados). Esto, nuevamente, producto de la mala adaptación de los parámetros del ES al mecanismo de ILS que ocasiona el enfriamiento tan rápido que lleva al retorno apresurado de soluciones. Consistentemente con este razonamiento, la ejecución adicional con los parámetros adaptados toma tiempos mucho más acordes al resto de algoritmos, y sorprendentemente, superando en eficiencia al ILS clásico para el conjunto Ecoli, lo que nos lleva a pensar que el operador de mutación de ILS favorece más la exploración en el ES que en la BL para conjuntos complejos como el de Ecoli.

En conclusión, todos los algoritmos estudiados son relativamente equivalentes para los conjuntos propuestos, e incluso algunos están terriblemente adaptados con sus parámetros iniciales. Los conjuntos de datos cada vez resultan peores para comparar, pues dos de los cuatro ya solo tienen interés a nivel de eficiencia temporal y como validadores de resultados. Por tanto, queda probado que a nivel individual, las búsquedas basadas en trayectorias (BL y ES) son técnicas poderosas y que optimizaciones sobre éstas, si bien dan lugar a soluciones marginalmente mejores en los conjuntos complejos, requieren el trabajo adicional de adaptar adecuadamente los parámetros, así como un potencial costo añadido en cuanto a eficiencia temporal dependiendo de qué tan bien se adapte la estrategia al problema.

7. Referencias bibliográficas:

- https://numpy.org/doc/1.17/
- Seminario 2. Problemas de optimización con técnicas basadas en búsqueda local. Óscar Cordón García.
- https://docs.python.org/3/library/pathlib.html
- Constrained K-means Clustering with Background Knowledge. Proceedings of the Eighteenth International Conference on Machine Learning, 2001, p. 577–584. Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schroedl.
- https://realpython.com/python-timer/
- http://zetcode.com/python/fstring/
- https://en.wikipedia.org/wiki/Iterated_local_search
- https://en.wikipedia.org/wiki/Simulated_annealing