Práctica Final:
Aplicación del Shuffled Frog-Leaping Algorithm para el Problema del Agrupamiento con Restricciones

Curso 2019-2020

Algoritmos considerados:

- *SFLAv1*
- *SFLAv2_1*
- *SFLAv2_2*
- *SFLAv2_2parallel*

Autor:

- *Nombre*: Alejandro Alonso Membrilla
- *DNI*: 75577394S
- *e-mail*: aalonso99@correo.ugr.es
- *Grupo*: Jueves de 17:30h a 19:30h. Aula 3.2. Profesor: Salvador García

Índice:

[1] Introducción al Problema de Agrupamiento con Restricciones:

El **agrupamiento** o **análisis de clusters** o ***clustering*** en inglés, persigue la clasificación de objetos de acuerdo a posibles similitudes entre ellos, agrupándolos en conjuntos llamados clusters. Este problema puede resolverse mediante múltiples algoritmos que pueden diferir significativamente en su percepción de qué es un cluster o en su método para calcularlos. Algunas nociones populares de cluster incluyen grupos cuyos miembros se encuentran a una distancia cercana unos de otros, o áreas densas del espacio de datos. El algoritmo de clustering y la configuración de sus parámetros deben ser escogidos dependiendo del dataset específico a estudiar y del uso que se le pretenda dar a los resultados. El **análisis de clusters** se incluye dentro del **aprendizaje no supervisado,** un tipo de aprendizaje automático que busca patrones previamente desconocidos en un conjunto de datos sin etiquetas ni supervisión humana. Esto permite el modelado de densidades de probabilidad mediante puntos u otro tipo de datos de entrada, lo que resulta de inmensa utilidad en el análisis de sistemas complejos pero de los que pueden tomarse un gran número de muestras de datos. El clustering tiene como aplicaciones el reconocimiento y clasificación de estímulos externos como imágenes o sonidos (Informática), clasificación de especies y subespecies de plantas y animales (Biología), clasificación de una enfermedad en diferentes subtipos dependiendo de los genes que se ven afectados por la misma (Medicina), etc.

El clustering con restricciones, por lo general, incorpora una serie de restricciones ***must-link*** (asociación necesaria) o ***cannot-link*** (asociación no permitida) al problema de clustering clásico (esto lo convierte en una técnica de **aprendizaje semisupervisado**). Ambas definen una relación entre dos instancias de datos: *must-link* indica que dos elementos deben estar en el mismo cluster, y *cannot-link* que deben estar en clusters diferentes. Algunos algoritmos abortan su ejecución cuando no es posible encontrar una solución que satisfaga todas y cada una de las restricciones (restricciones fuertes), mientras que otros tratan de minimizar el número de restricciones insatisfechas atribuyéndole un peso a este factor y penalizando las soluciones obtenidas que las incumplen.

[2] Modelización matemática del problema:

El objetivo de los algoritmos realizados será encontrar una partición C de los elementos del dataset X que minimice la **distancia euclídea** de forma general entre los elementos de cada cluster (**desviación media intra-cluster**) y la ***infeasibility*** o **infactibilidad** dada por el número de restricciones insatisfechas. Denotaremos N como el tamaño de la muestra (número de elementos distintos en el dataset), $c_1 \dots c_k$ denotarán los k clusters distintos (k será un parámetro de ambos algoritmos) cumpliendo que

$$c_1 \cup c_2 \cup \dots \cup c_k = X \quad , \quad |c_i| > 0 \; \forall i \in \{1, \dots, k\} \quad , \quad c_i \cap c_j = \emptyset \; \forall i, j : i \neq j$$

Finalmente, $C = \{c_1, \dots c_k\}$ será una posible solución del problema. Esta solución se representará mediante un vector S de tamaño N en el que el i-ésimo elemento $s_i \in \{1, \dots, k\}$. Es decir, cada elemento del vector solución S indica el cluster al que pertenecería el elemento correspondiente de X.

NOTA: se denominará dimX al número de coordenadas de los elementos de X.

Para cada cluster c_i se puede calcular su centroide asociado μ_i como el vector promedio de las instancias de X que lo componen:

$$\vec{\mu}_i = \frac{1}{|c_i|} \sum_{\vec{x}_j \in c_i} \vec{x}_j$$

Definimos la distancia media intra-cluster c_i como la media de las distancias de las instancias que lo conforman a su centroide:

$$\bar{c}_i = \frac{1}{|c_i|} \sum_{\vec{x}_j \in c_i} \|\vec{x}_j - \vec{\mu}_i\|_2$$

Definimos la desviación general de la partición C como la media de las desviaciones intra-cluster de los clusters de C:

$$\bar{C} = \frac{1}{k} \sum_{c_i \in C} \bar{c}_i$$

Antes de mostrar la fórmula para el cálculo de la infactibilidad, definimos la siguiente notación:

$$ML = \{(\vec{x}_i, \vec{x}_j, 1) / \textit{hay una restricción must-link entre } \vec{x}_i \textit{ y } \vec{x}_j\}$$

$$CL = \{(\vec{x}_i, \vec{x}_j, -1) / \textit{hay una restricción cannot-link entre } \vec{x}_i \textit{ y } \vec{x}_j\}$$

El 1 y el -1 de la definición anterior nos permite diferenciar entre un tipo de restricción y otra en el conjunto R total de restricciones.

$$R = CL \cup ML$$

Dada la i-ésima restricción de un conjunto de restricciones, por ejemplo ML (análogo para CL), denotaremos $\vec{ML}_{[i,1]}$ al primer dato que conforma dicha restricción y $\vec{ML}_{[i,2]}$ al segundo.

Finalmente, definimos $h_C(\cdot)$ como la función que devuelve, para cada vector del dataset, el índice del cluster al que pertenece de acuerdo con la partición C.

La *infeasibility*, tal como se ha definido anteriormente, puede calcularse de la siguiente forma:

$$infeasibility = \sum_{i=1}^{|ML|} \mathrm{I}[h_C(\vec{ML}_{[i,1]}) \neq h_C(\vec{ML}_{[i,2]})] + \sum_{i=1}^{|CL|} \mathrm{I}[h_C(\vec{CL}_{[i,1]}) = h_C(\vec{CL}_{[i,2]})]$$

Las restricciones pueden almacenarse o bien en forma de matriz (tal como vienen dadas en sus ficheros de texto correspondientes) o en forma de lista, eliminando las relaciones nulas y agilizando el cómputo del valor total.

La función objetivo que se desea minimizar viene dada por la siguiente expresión:

$$f = \bar{C} + \lambda * infeasibility$$

Donde λ es una constante dependiente de X dado por:

$$\lambda = \frac{\lceil D \rceil}{|R|}$$

Esto es, λ representa la relación entre la distancia máxima entre dos elementos pertenecientes a X, y el número total de restricciones.

El pseudocódigo del cálculo de los centroides, la desviación general, la infactibilidad y la función objetivo puede escribirse de forma general como viene a continuación:

```
C = {c_1...c_k} partición de X en k clusters
centroides = matriz de dimensión k*dim(X), cada fila guardará el centroide
            de un cluster

calcular_centroides(){

      for i=1 to i=k:
            centroides[i] = suma(x : x en c_i)/|c_i|
      return centroides
}
```

```
X = dataset; Para x en X, hc(x) = cluster al que pertenece x
μ_hc(x) = centroide del cluster hc(x)

desviacion_general(){
      desviacion_intra_cluster[k]={0,…,0}
      for x in X:
            desviacion_intra_cluster[h_c(x)] += ||x-μ_hc(x)||_2 / |c_hc(x)|
      desviacion_general = media(desviacion_intra_cluster)

      return desviacion_general
}
```

```
R = lista de restricciones; Si r en R entonces r[0], r[1] son elementos de
X, y r[2] = 1 ó -1 (indicando el tipo de restricción)
infeasibility(){
      infeasibility = 0
      for r in R:
            if h_c(r[0]) != h_c(r[1]) and r[2]==1 or
               h_c(r[0]) == h_c(r[1]) and r[2]==-1:
                  infeasibility += 1
      return infeasibility
}
```

```
funcion_objetivo():
      return desviacion_general() + λ*infeasibility()
```

[3] El Shuffled Frog-Leaping Algorithm:

El SFLA (Shuffled Frog-Leaping Algorithm) se trata de un algoritmo memético que agrupa una población de distintas soluciones, denominadas **ranas,** en subpoblaciones **(memeplexes)**, a las que aplica una búsqueda local basada en el PSO (Particle Swarm Algorithm) usando una función objetivo para medir la bondad de cada solución. Tras cierto número de iteraciones destinadas a esta búsqueda local, las distintas subpoblaciones se mezclan y se vuelven a repartir, en un proceso de baraje o **shuffling** que puede encontrarse en otros algoritmos anteriores basados en el SCEA (Shuffled Complex Evolutionary Algorithm).

Tal como podría esperarse debido a su nombre, el SFLA se inspira en el comportamiento de las ranas saltando entre las piedras de una charca. Cada piedra tiene una bondad asociada (por ejemplo, cantidad de comida disponible) y es el objetivo de cada rana mejorar su posición en la charca. Las ranas pueden comunicarse entre ellas, lo cual ayuda a las ranas en peores piedras a desplazarse hacia zonas de la charca mejores. La piedra en la que una rana busca comida tiene una serie de propiedades, llamadas **memes** (análogas a los genes en los algoritmos genéticos), que pueden compartirse y distribuirse al resto de la población, o a una parte. Cada grupo de ranas que comparten información es un **memeplex,** como ya se ha mencionado anteriormente.

Entremos en el algoritmo en sí. Como se ha indicado el SFLA consta de dos fases alternadas: una global y otra local.

Fase Global:

1. *Generar población inicial:* dado m un número de memeplexes, y de ranas n, previamente fijado, generamos $m \cdot n$ soluciones factibles que guardamos en una matriz.
2. *Ordenamos la población:* evaluamos cada rana y ordenamos el total en orden creciente de función objetivo (recordamos que el problema de clustering busca minimizar dicha función).
3. *Separar las ranas en memeplexes:* para asegurar que se hace un reparto equilibrado, a partir de la población ordenada en el paso anterior aplicaremos un baraje **modular.** Esto es, dado una rana con posición i en el ranking global, el memeplex correspondiente a dicha rana será aquel con índice $i \bmod m$.
4. *Evolución memética:* búsqueda local basada en PSO. Explicada más adelante.
5. *Shuffling:* la búsqueda local en cada memeplex ha podido modificar los memes que portan las ranas que lo componen, cambiando el valor objetivo de cada rana. Al remezclar las ranas, las volvemos a ordenar en orden creciente de función objetivo.
6. *Comprobar condición de convergencia:* si esta se satisface acaba el algoritmo devolviendo la primera rana (aquella con valor objetivo más bajo). En caso contrario se vuelve al paso 3.

Fase Local (paso 4 de la Fase Global):

1. *Inicializar:* $im=0$ (índice del memeplex al que se aplica la búsqueda local). Se fija un número máximo de iteraciones $N>0$.

2. *Iniciar iteraciones de la búsqueda local: iN=0.*
3. *Construcción de un submemeplex:* tomamos un subconjunto aleatorio de *q* ranas del memeplex. Como se toma de forma aleatoria, no siempre tomaremos los mejores memes, favoreciendo la exploración. Sí que se asignan, sin embargo, probabilidades más altas a las ranas en mejor posición. De cara a la implementación, esto puede obtenerse iterando sobre los elementos del memeplex y seleccionándo el elemento *k* con probabilidad $2(n+1-k)/[n(n+1)]$ (la iteración se repite hasta que se hayan seleccionado *q* ranas distintas). La implementación realizada emplea este método por eficiencia, aunque podría usarse algún tipo de algoritmo de muestreo sin repetición más directo o incluso tener en cuenta no solo la posición de cada rana, sino su valor objetivo exacto.
4. *La peor rana se acerca a la mejor:* para cada elemento/coordenada de la solución correspondiente a la peor de las ranas en el submemeplex, desplazamos su valor en dirección al mismo elemento de la mejor rana del mismo submemeplex. Esto puede conseguirse conservando la exploración, generando un valor aleatorio entre ambas posiciones. Una vez modificada la peor rana, la evaluamos. Si la peor rana ha mejorado su función objetivo avanzamos al paso 7. Si no lo ha hecho o si el resultado obtenido no es factible, avanzamos al paso 3.
5. *La peor rana se acerca a la mejor de la charca:* tomamos la peor rana sin la modificación del paso 4, y aplicamos el mismo procedimiento del paso 4 pero tomando como referencia a la mejor rana **global. S**i la peor rana ha mejorado su función objetivo avanzamos al paso 7. Si no lo ha hecho o si el resultado obtenido no es factible, avanzamos al paso 6.
6. *Censura:* cambiamos la peor rana por una solución factible generada aleatoriamente.
7. *Devolución al memeplex:* devolvemos las ranas del submemeplex al memeplex. Esto implica modificar el valor que se tenía anteriormente de la rana modificada.
8. *Incrementamos la iteración: iN=iN+1.* Si $iN<N$, volvemos al paso 3.
9. *Cambiamos de memeplex: im=im+1.* Si $im\leq m$, volvemos al paso 2.

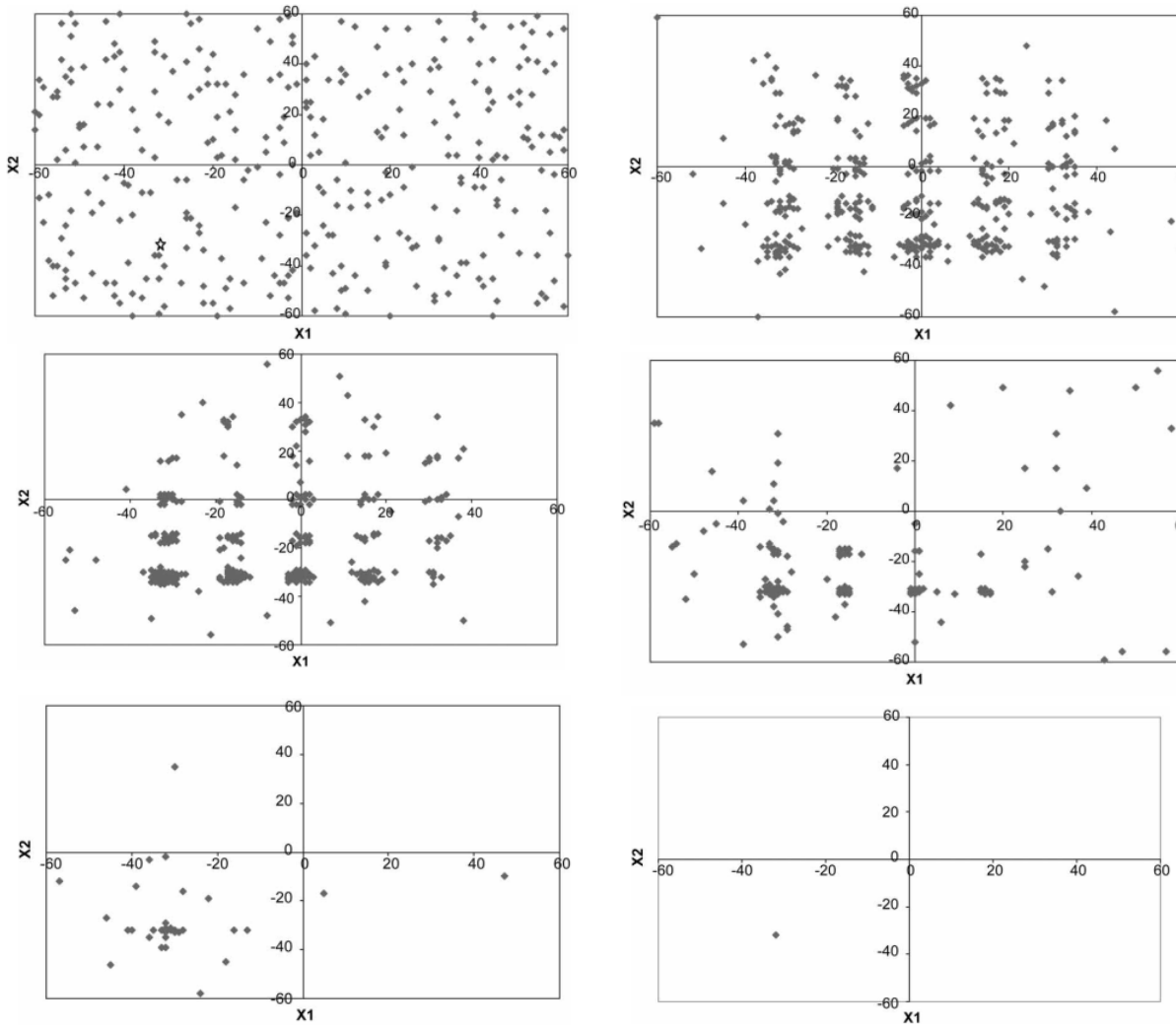
La adaptación del problema de agrupamiento de restricciones al SFLA es directa y sencilla. Cada solución factible al problema se almacenará como un vector (rana) con tantas coordenadas como elementos haya en el dataset, y a cada coordenada se le asignará el cluster al que adscribimos el punto correspondiente. La función objetivo usada para medir la bondad de cada rana será la descrita en el apartado **[2]**.

El algoritmo requerirá llevar un **ranking** de todas las ranas (tanto global como localmente) así que indexar las ranas permite saber a que individuo corresponde cada valor objetivo sin tener que ordenar la matriz total, lo cual sería computacionalmente ineficiente. Cada rana será evaluada, formando una pareja índice-objetivo que añadiremos a un conjunto que las ordena automáticamente por su valor objetivo.

Por otro lado, observamos que el algoritmo cuenta con una serie de **parámetros:** número de memeplexes, ranas por memeplex, tamaño de los submemeplexes y número de iteraciones en la exploración local. Si bien podría realizarse un estudio concienzudo sobre cuales son los valores más adecuados que asignar a cada parámetro, en nuestra implementación hemos usado *m=10, n=30, q=10* y *N=20,* puesto que han dado resultados relativamente buenos para el problema estudiado y el límite de evaluaciones de la función objetivo establecido.

En el SFLA cada memeplex realiza una búsqueda independiente del resto de memeplexes, lo que constituye una estructura muy favorable para paralelizar este algoritmo evolutivo. Hemos implementado una versión paralela del mismo, el *SFLAv2_2parallel*, capaz de aprovechar los 2 núcleos físicos de la máquina en la que se ha ejecutado, cuyos resultados analizaremos más adelante.

A continuación mostramos un ejemplo de la evolución de la población de ranas en una aplicación del SFLA (fotos ordenadas por filas), sacada del artículo *Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization (M Eusuff, K Lansey, F Pasha; 2006).*



[4] Mejoras sobre el algoritmo:

Además del SFLA estándar, hemos diseñado un par de mejoras para hacer el algoritmo más efectivo para el problema del agrupamiento con restricciones.

En primer lugar, hemos añadido al SFLA un esquema de reinicialización que regenera, cada cierto número de ejecuciones de la función objetivo, la mayor parte de la población de la charca. Para evitar deshacernos de las mejores soluciones encontradas hasta el momento, cada vez que reiniciamos la población guardamos cierto número de las mejores ranas. El número de ranas que guardamos empieza siendo "bajo" para favorecer la exploración, y crece con el número de reinicios. En nuestra implementación específica, el número de ranas reservadas es igual al doble de reinicios que se hayan realizado, aunque este criterio es arbitrario y podría convenir matizarlo si cambiásemos los parámetros establecidos.

En segundo lugar, dada la versión mejorada recién expuesta, hemos probado dos modelos de inicialización de soluciones aleatorias: el modelo básico, en el que se inicializa cada coordenada (meme o gen) de la solución de forma directa y aleatoria (asegurando la factibilidad), y una segunda forma, en la que colocamos aleatoriamente tantos puntos como centroides en el espacio factible (dentro de los límites de los extremos del dataset), una especie de pseudocentroides, asociando cada punto de los datos a su psuedocentroide más cercano. Esto limita la posible forma de las soluciones generadas, pero al mismo tiempo facilita que los clusters iniciales tengan puntos cercanos entre sí. La comparación entre ambos métodos de inicialización se estudiará en el análisis de resultados. El código SFLAv2-1 emplea el primer método mientras que el SFLAv2-2 usa el segundo.

Los pseudocódigos de generación de soluciones aleatorias y del SFLA con reinialización se muestran a continuación:

```
generar_solución_aleatoriaV1(){     // X es el dataset
      solución = vector[|X|]  // |X| es el número de puntos en X
      do{
            for s in solución {
                  s = random_integer(0, num_clusters)
            }
      while(algún_cluster_vacío(solución))
      return solución
}
```

```
generar_solución_aleatoriaV2(){
      solución = vector[|X|]
      centroides = vector[num_clusters][dim(X)]

      for i=0, i<|X| {
            Para k = 1...num_clusters:
            solución[i] = argmin(distancia(centroides[k],X[i]))
      }

      if(alúun_cluster_vacío en solución){
            solución = generar_solución_aleatoriaV1()
      }
      return solución
}
```

```
//Para los parámetros del SFLA, se usa la notación expuesta en el apartado 3
sfla_con_reinicio(){

   ranas[n*m][|X|]
   loss = set<entero, flotante>     //Conjunto de parejas índice-objetivo
   for rana in ranas{
      rana = generar_solucion_aleatoria() //Alguno de los dos métodos
      calcular_centroides()
      añadir {rana.indice, f_objetivo(rana)} a loss
   }
   for r=0 to r=total_de_reinicios{
      for im=0 to im=m{
         memeplex = set<entero, flotante>  (ranas[posicion mod im == 0])
         for i=0 to i=q{
            submemeplex = set<entero, flotante>
            for rana in memeplex{
               if(random_float(0,1) < 2*(n-rana.pos)/(rana.pos^2+rana.pos)){
                  submemeplex ← rana
               }
            }
            acercar_peor_hacia_mejor() //Paso 4 de la exploración local
            if(no es factible o no hay mejoría){
               acercar_peor_hacia_mejor_de_la_charca() //Paso 5
               if(no es factible o no hay mejoria){
                  peor_rana = generar_solucion_aleatoria()
               }
            }
         }
      }
      vaciar(loss)
      for each memeplex{
         loss ← memeplex
      }

      for(i=r to i=n*m){
         ranas[i] = generar_solución_aleatoria()
         calcular_centroides(ranas[i])
         loss ← {i, f_objetivo(ranas[i])}
      }
   }

   Return ranas[loss.mejor_rana.indice]

}
```

En lo que respecta a los parámetros de ejecución, para el SFLAv2 (todas las versiones con reinicialización) se han establecido los parámetros *m=10, n=30, q=10, N=20 y 5 inicializaciones* (es decir, cada iteración tiene 20000 evaluaciones disponibles de la función objetivo).

[5] Procedimiento llevado en el desarrollo de la práctica:

El código de los algoritmos estudiados en la presente práctica ha sido escrito por el autor de la misma usando C++.

Las ejecuciones del algoritmo han sido realizadas en un ordenador Acer Aspire E15 con las siguientes especificaciones:

Memoria: 8GB DDR3 L Memory
Procesador: Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz
Arquitectura: x86_64
CPU(s): 4
Hilo(s) de procesamiento por núcleo: 2
Núcleo(s) por «socket»: 2
«Socket(s)»: 1
CPU MHz: 1772.602
CPU MHz máx.: 2700,0000
CPU MHz mín.: 800,0000
Caché L1d: 32K
Caché L1i: 32K
Caché L2: 256K
Caché L3: 3072K
Gráficos: NVIDIA GeForce 940M/PCIe/SSE2
Disco: 969,9 GB
SO: Ubuntu 18.04.4 LTS

La ejecución de los algoritmos ha sido automatizada mediante mediante una función main() que los ejecuta y muestra sus resultados por terminal.

[5.1] Manual de usuario:

Para ejecutar el código entregado basta con compilar los fuentes (entrar en BIN desde la terminal y ejecutar el Makefile) y ejecutar el binario *main*. Si se desea ejecutar un algoritmo para un dataset/semilla/número de clusters específico es posible pasarlos como parámetro (la instrucciones necesarias se indican al ejecutar el programa sin argumentos). Si se escribe *main all*, se ejecutarán todos los algoritmos para los distintos datasets y semillas secuencialmente.

[6] Experimentos y análisis de resultados:

En el desarrollo de la práctica se ha trabajado con 8 instancias del PAR generadas a partir de los 4 conjuntos de datos siguientes:

1. Iris: Clásico en ciencia de datos. Contiene información sobre las características de tres tipos de flor de Iris. Tiene 3 clases (k = 3).

2. Ecoli: Contiene medidas sobre ciertas características de diferentes tipos de células que pueden ser empleadas para predecir la localización de ciertas proteínas. Tiene 8 clases (k = 8).

3. Newthyroid: Contiene medidas cuantitativas tomadas sobre la glándula tiroides de 215 pacientes. Presenta 3 clases distintas (k = 3).

4. Rand: Conjunto de datos artificial. Formado por tres agrupamientos bien diferenciados generados en base a distribuciones normales (k = 3).

A partir de cada conjunto de datos se obtienen 2 instancias distintas generando 2 conjuntos de restricciones, correspondientes al 10% y 20% del total de restricciones posibles. Los ficheros de datos contienen en cada fila una instancia del conjunto en cuestión con sus característica

separadas por comas (sin espacios). Los ficheros de restricciones almacenan una matriz de restricciones con el formato descrito en las transparencias del Seminario 2, con los valores separados por comas y sin espacios.

Las semillas empleadas en las 5 ejecuciones de ambos algoritmos son, en este orden: 31, 41, 71, 109 y 163. A modo de anotación, es necesario señalar que debido a la naturaleza asíncrona de la versión paralela (SFLAv2_2parallel) los resultados obtenidos por la misma **no son replicables.** Sin embargo, como se verá a continuación, sus resultados medios son prácticamente iguales a los de SFLAv2-2 (lo cual tiene sentido, puesto que lo único que difiere entre ellos es en los números pseudoaleatorios que generan).

También necesitamos tener en cuenta que los algoritmos genéticos y meméticos de la práctica 2 se programaron en Python, por lo que sus tiempos de ejecución no son comparables a los obtenidos en esta práctica.

Los resultados obtenidos son los siguientes (el tiempo T se mide en segundos):

[6.1] Tablas de resultados

Tabla 1: Resultados obtenidos por el algoritmo SFLAv1 en el PAR con 10% de restricciones

	Iris				**Ecoli**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0.6810	3.0000	0.7025	1.9370	27.6378	495.0000	40.9188	6.3920
Ejecución 2	0.8676	224.0000	2.4719	2.2200	22.9687	510.0000	36.6521	6.4630
Ejecución 3	0.6693	0.0000	0.6693	1.9680	26.5981	1006.0000	53.5892	6.1980
Ejecución 4	0.6846	39.0000	0.9639	1.9390	22.7831	384.0000	33.0858	6.4180
Ejecución 5	0.6693	0.0000	0.6693	1.9510	27.5334	1135.0000	57.9856	6.1840
Media	0.7144	53.2000	1.0954	2.0030	25.5042	706.0000	44.4463	6.3310

	Newthyroid				**Rand**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	9.4222	668.0000	34.1091	3.4530	0.7536	18.0000	0.8986	1.8940
Ejecución 2	15.7016	45.0000	17.3646	3.5470	1.3666	232.0000	3.2359	1.8440
Ejecución 3	16.4248	51.0000	18.3096	3.4720	0.7119	9.0000	0.7845	1.8560
Ejecución 4	14.3816	28.0000	15.4164	3.3550	0.7156	0.0000	0.7156	1.8550
Ejecución 5	11.9314	96.0000	15.4792	3.4410	0.8567	65.0000	1.3804	1.8850

Tabla 2: Resultados obtenidos por el algoritmo SFLAv2-1 en el PAR con 10% de restricciones

	Iris				**Ecoli**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0.6693	0.0000	0.6693	2.3170	25.5454	526.0000	39.6581	6.7640
Ejecución 2	0.8535	230.0000	2.5008	2.3680	28.8595	880.0000	52.4700	6.5870
Ejecución 3	0.6693	0.0000	0.6693	2.3650	26.7185	983.0000	53.0925	6.4320
Ejecución 4	0.6744	50.0000	1.0325	2.3190	23.0199	338.0000	32.0885	6.7910
Ejecuión 5	0.6693	0.0000	0.6693	2.3470	26.8886	1086.0000	56.0261	6.5450
Media	0.7072	56.0000	1.1083	2.3432	26.2064	762.6000	46.6670	6.6238

	Newthyroid				**Rand**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	13.5583	102.0000	17.3278	3.8720	0.7452	6.0000	0.7935	2.2240
Ejecución 2	15.5443	33.0000	16.7639	3.7710	1.4524	213.0000	3.1686	2.2440
Ejecución 3	15.2403	62.0000	17.5316	3.7370	0.7156	0.0000	0.7156	2.1920
Ejecución 4	15.0127	39.0000	16.4540	3.7190	0.7156	0.0000	0.7156	2.2880
Ejecuión 5	11.9314	96.0000	15.4792	3.7030	0.7755	33.0000	1.0414	2.2900

Tabla 3: Resultados obtenidos por el algoritmo SFLAv2-2 en el PAR con 10% de restricciones

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0.6693	0.0000	0.6693	2.3340	24.9765	535.0000	39.3306	6.7930
Ejecución 2	0.6693	0.0000	0.6693	2.3740	22.1345	994.0000	48.8036	6.6190
Ejecución 3	0.6693	0.0000	0.6693	2.3510	20.8452	640.0000	38.0165	6.8210
Ejecución 4	0.6696	6.0000	0.7126	2.3360	22.8403	311.0000	31.1845	6.8410
Ejecución 5	0.6693	0.0000	0.6693	2.2710	27.6420	551.0000	42.4254	6.6180
Media	0.6694	1.2000	0.6780	2.3332	23.6877	606.2000	39.9521	6.7384

	Newthyroid				Rand			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	10.6371	99.0000	14.2958	3.8400	0.7156	0.0000	0.7156	2.2480
Ejecución 2	14.4106	30.0000	15.5193	3.8070	0.7156	0.0000	0.7156	2.2100
Ejecución 3	14.6860	45.0000	16.3490	3.7970	0.7156	0.0000	0.7156	2.2490
Ejecución 4	14.5961	41.0000	16.1113	3.7520	0.7156	0.0000	0.7156	2.2650
Ejecución 5	11.1760	107.0000	15.1303	3.8190	0.7156	0.0000	0.7156	2.2710
Media	13.1012	64.4000	15.4811	3.8030	0.7156	0.0000	0.7156	2.2486

Tabla 4: Resultados obtenidos por el algoritmo SFLAv2-2parallel en el PAR con 10% de restricciones

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0.6693	0.0000	0.6693	1.8760	27.7487	488.0000	40.8418	3.6870
Ejecución 2	0.6602	17.0000	0.7820	1.7160	21.2105	609.0000	37.5500	3.7000
Ejecución 3	0.6693	0.0000	0.6693	1.8010	25.5817	428.0000	37.0650	3.9330
Ejecución 4	0.6693	0.0000	0.6693	1.7900	22.6110	468.0000	35.1675	4.0820
Ejecución 5	0.6850	15.0000	0.7924	1.7280	24.0092	646.0000	41.3414	4.2990
Media	0.6706	6.4000	0.7165	1.7822	24.2322	527.8000	38.3931	3.9402

	Newthyroid				Rand			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	10.6235	96.0000	14.1714	2.5820	0.7156	0.0000	0.7156	1.7030
Ejecución 2	14.8311	56.0000	16.9007	2.6320	0.7156	0.0000	0.7156	1.6770
Ejecución 3	17.1971	77.0000	20.0428	2.3030	0.7156	0.0000	0.7156	1.8970
Ejecución 4	14.8446	23.0000	15.6946	2.3340	0.7156	0.0000	0.7156	1.9770
Ejecución 5	10.7155	99.0000	14.3742	2.4440	0.7156	0.0000	0.7156	1.5790
Media	13.6424	70.2000	16.2367	2.4590	0.7156	0.0000	0.7156	1.7666

Tabla 5: media de los resultados obtenidos por cada algoritmo en el PAR con 10% de restricciones

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
COPKM	0.6640	12.8000	0.7480	0.3360	36.9980	222.2000	42.6600	13.6620
BL	0.7087	48.0000	1.0525	0.0468	31.9513	1059.6000	60.3805	4.0406
BMB	0.6693	0.0000	0.6693	0.4610	30.9789	1091.2000	60.2559	4.1538
ES	0.6693	0.0000	0.6693	0.1796	21.7699	78.6000	23.8788	2.6034
ILS_BL	0.6693	0.0000	0.6693	0.2120	36.8628	1128.0000	67.1272	3.8272
ILS_ES	0.6693	0.0000	0.6693	0.2088	39.2603	1187.6000	71.1238	1.0324
ILS_ESv2	0.7518	69.8000	1.2518	0.1198	27.1459	380.2000	37.3467	3.2306
AGE-CU	1.8289	433.2000	4.9645	395.4912	45.1246	1728.6000	91.8093	1266.7649
AGE-CSF	0.8961	179.4000	2.1947	395.7278	43.1012	1558.0000	85.1784	1254.1372
AGG-CU	1.3802	319.0000	3.6892	356.2720	42.9482	1454.6000	82.2329	1125.3195
AGG-CSF	0.9792	187.8000	2.3386	360.6781	42.1086	1429.0000	80.7019	1144.2363
AM-(10,1.0)	1.4801	322.4000	3.8137	376.1226	44.6770	1653.0000	89.3200	1189.8078
AM-(10,0.1)	0.9950	189.6000	2.3674	378.3358	43.3957	1490.8000	83.6581	1165.9391
AM-(10,0.1MEJ)	1.1810	229.4000	2.8415	380.5959	43.1913	1496.4000	83.6049	1177.8385
SFLAv1	0.7144	53.2000	1.0954	2.0030	25.5042	706.0000	44.4463	6.3310
SFLAv2-1	0.7072	56.0000	1.1083	2.3432	26.2064	762.6000	46.6670	6.6238
SFLAv2-2	0.6694	0.0000	0.6780	2.3332	23.6877	606.2000	39.9521	6.7384
SFLAv2-2parallel	0.6706	6.4000	0.7165	1.7822	24.2322	527.8000	38.3931	3.9402

	Newthyroid				Rand			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
COPKM	x	x	x	x	0.7500	0.0000	0.7500	0.3240
BL	12.5763	43.2000	14.1728	0.4404	0.8471	47.4000	1.2290	0.0526
BMB	11.9972	134.4000	16.3226	1.4482	0.7156	0.0000	0.7156	0.2892
ES	12.6504	42.8000	14.2321	0.3530	0.7156	0.0000	0.7156	0.1504
ILS_BL	12.6504	42.8000	14.2321	0.8454	0.7156	0.0000	0.7156	0.1324
ILS_ES	13.8421	86.4000	17.0351	0.4906	0.7156	0.0000	0.7156	0.1792
ILS_ESv2	12.4152	90.4000	15.7561	0.6350	0.7978	29.0000	1.0314	0.1184
AGE-CU	13.4154	1038.8000	51.8573	633.7665	2.6531	426.6000	6.1073	390.3711
AGE-CSF	15.4570	560.4000	36.1952	646.3932	1.3098	165.2000	2.6474	397.6981
AGG-CU	14.2847	683.0000	39.5599	576.6832	1.9655	281.0000	4.2407	356.2747
AGG-CSF	14.6835	544.8000	34.8444	586.1826	1.4330	192.8000	2.9941	362.8217
AM-(10,1.0)	13.9925	803.8000	43.7379	600.9357	2.0743	310.0000	4.5843	373.4074
AM-(10,0.1)	14.0753	704.2000	40.1350	607.7462	1.5731	207.4000	3.2524	378.5930
AM-(10,0.1MEJ)	14.9616	583.0000	36.5361	626.2960	1.5597	205.2000	3.2212	391.4706
SFLAv1	13.5723	177.6000	20.1358	3.4536	0.8809	64.8000	1.4030	1.8668
SFLAv2-1	14.2574	66.4000	16.7113	3.7604	0.8809	50.4000	1.2869	2.2476
SFLAv2-2	13.1012	64.4000	15.4811	3.8030	0.7156	0.0000	0.7156	2.2486
SFLAv2-2parallel	13.6424	70.2000	16.2367	2.4590	0.7156	0.0000	0.7156	1.7666

Tabla 6: Resultados obtenidos por el algoritmo SFLAv1 en el PAR con 20% de restricciones

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0.6693	0.0000	0.6693	2.9810	27.2518	943.0000	39.9022	10.1530
Ejecución 2	0.8400	495.0000	2.6118	3.2300	23.8072	1949.0000	49.9532	10.4350
Ejecución 3	0.6693	0.0000	0.6693	3.1780	26.7726	1901.0000	52.2747	10.1830
Ejecución 4	0.6741	116.0000	1.0893	3.0710	20.9566	1021.0000	34.6534	11.0140
Ejecución 5	0.6693	0.0000	0.6693	2.8890	26.3619	2196.0000	55.8214	10.3440
Media	0.7044	122.2000	1.1418	3.0698	25.0300	1602.0000	46.5210	10.4258

	Newthyroid				Rand			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	10.9369	1224.0000	33.5494	5.5560	0.7384	16.0000	0.8028	2.9980
Ejecución 2	15.8640	170.0000	19.0046	5.3770	1.3400	483.0000	3.2850	3.0930
Ejecución 3	14.3891	108.0000	16.3844	5.5920	0.7156	0.0000	0.7156	2.9070
Ejecución 4	14.4221	21.0000	14.8100	5.5340	0.7156	0.0000	0.7156	2.8890
Ejecución 5	11.9314	218.0000	15.9588	5.5930	0.8024	101.0000	1.2091	2.9510
Media	13.5087	348.2000	19.9414	5.5304	0.8624	120.0000	1.3456	2.9676

Tabla 7: Resultados obtenidos por el algoritmo SFLAv2-1 en el PAR con 20% de restricciones

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0.6693	0.0000	0.6693	3.3940	25.2553	1101.0000	40.0253	11.0020
Ejecución 2	0.8426	498.0000	2.6251	3.4840	30.0510	1974.0000	56.5323	10.4290
Ejecución 3	0.6693	0.0000	0.6693	2.9540	26.3830	1878.0000	51.5765	10.4640
Ejecución 4	0.6721	15.0000	0.7258	3.4310	23.4666	874.0000	35.1914	11.1690
Ejecuión 5	0.6761	13.0000	0.7226	3.5540	26.6985	2253.0000	56.9226	10.6100
Media	0.7059	105.2000	1.0824	3.3634	26.3709	1616.0000	48.0496	10.7348

	Newthyroid				Rand			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	13.6751	379.0000	20.6769	6.0870	0.7362	17.0000	0.8046	3.3220
Ejecución 2	15.0187	107.0000	16.9954	5.7330	1.3389	489.0000	3.3080	3.5030
Ejecución 3	14.7567	83.0000	16.2901	5.7200	0.7156	0.0000	0.7156	3.2700
Ejecución 4	14.1598	13.0000	14.4000	5.6370	0.7156	0.0000	0.7156	3.3620
Ejecuión 5	11.9314	218.0000	15.9588	5.8780	0.8551	110.0000	1.2980	3.3380
Media	13.9083	160.0000	16.8642	5.8110	0.8723	123.2000	1.3684	3.3590

Tabla 8: Resultados obtenidos por el algoritmo SFLAv2-2 en el PAR con 20% de restricciones

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0.6693	0.0000	0.6693	3.4450	25.4251	1182.0000	41.2817	11.3350
Ejecución 2	0.6649	17.0000	0.7258	3.4710	26.0660	980.0000	39.2127	10.8220
Ejecución 3	0.6693	0.0000	0.6693	3.3880	24.8050	1536.0000	45.4105	10.6470
Ejecución 4	0.6693	0.0000	0.6693	3.5560	23.7675	773.0000	34.1374	11.3630
Ejecución 5	0.6693	0.0000	0.6693	3.4350	23.9678	914.0000	36.2292	11.1700
Media	0.6684	3.4000	0.6806	3.4590	24.8063	1077.0000	39.2543	11.0674

	Newthyroid				Rand			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	14.1468	69.0000	15.4215	5.8790	0.7156	0.0000	0.7156	3.2930
Ejecución 2	14.8687	96.0000	16.6422	5.8820	0.7156	0.0000	0.7156	3.4040
Ejecución 3	15.4115	110.0000	17.4437	5.8840	0.7156	0.0000	0.7156	3.3720
Ejecución 4	14.5329	44.0000	15.3458	5.7250	0.7156	0.0000	0.7156	3.4470
Ejecución 5	10.7267	232.0000	15.0127	5.9570	0.7156	0.0000	0.7156	3.2920
Media	13.9373	110.2000	15.9732	5.8654	0.7156	0.0000	0.7156	3.3616

Tabla 9: Resultados obtenidos por el algoritmo SFLAv2-2parallel en el PAR con 20% de restricciones

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0.6693	0.0000	0.6693	2.4080	25.8713	1059.0000	40.0779	6.5060
Ejecución 2	0.6693	0.0000	0.6693	2.4030	18.4052	1215.0000	34.7045	6.3850
Ejecución 3	0.6693	0.0000	0.6693	2.4080	28.0064	1136.0000	43.2460	6.2220
Ejecución 4	0.6693	0.0000	0.6693	2.1170	23.9128	842.0000	35.2083	6.4850
Ejecución 5	0.6693	0.0000	0.6693	2.6560	25.2560	1367.0000	43.5944	6.0760
Media	0.6693	0.0000	0.6693	2.3984	24.2903	1123.8000	39.3662	6.3348

	Newthyroid				Rand			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	14.0642	81.0000	15.5606	3.5460	0.7156	0.0000	0.7156	2.2810
Ejecución 2	15.4920	137.0000	18.0230	4.1360	0.7156	0.0000	0.7156	2.2390
Ejecución 3	14.1419	96.0000	15.9154	3.4900	0.7156	0.0000	0.7156	2.2040
Ejecución 4	14.6703	43.0000	15.4647	3.9280	0.7156	0.0000	0.7156	2.2610
Ejecución 5	10.7267	232.0000	15.0127	3.5600	0.7156	0.0000	0.7156	2.2450
Media	13.8190	117.8000	15.9953	3.7320	0.7156	0.0000	0.7156	2.2460

Tabla 10: media de los resultados obtenidos por cada algoritmo en el PAR con 20% de restricciones

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
COPKM	0.6620	2.0000	0.6680	0.3080	36.1200	181.4000	38.4980	6.5308
BL	0.7048	100.8000	1.0656	0.0928	31.9257	2041.0000	59.3058	8.4040
BMB	0.6693	0.0000	0.6693	0.7616	30.7502	2183.0000	60.0353	8.9296
ES	0.6693	0.0000	0.6693	0.3212	21.7674	168.6000	24.0292	5.7378
ILS_BL	0.6693	0.0000	0.6693	0.3770	36.5980	2256.4000	66.8678	7.9742
ILS_ES	0.6693	0.0000	0.6693	0.3422	37.1773	2622.2000	72.3543	1.8396
ILS_ESv2	0.7345	0.0000	1.0810	0.2480	27.2157	677.8000	36.3084	6.6278
AGE-CU	1.7605	892.4000	4.9888	519.6885	45.1190	3499.0000	92.3681	1884.1881
AGE-CSF	0.9441	376.4000	2.3058	524.5151	42.2472	3169.0000	85.0401	1882.9124
AGG-CU	1.1444	542.6000	3.1072	438.9558	41.7953	3066.0000	83.1973	1583.8081
AGG-CSF	0.9326	368.4000	2.2653	431.5933	40.2405	2872.2000	79.0256	1545.9742
AM-(10,1.0)	1.4224	658.0000	3.8027	521.6618	44.6353	3382.0000	90.3046	1833.8258
AM-(10,0.1)	1.0489	426.0000	2.5900	493.9535	42.1419	3074.8000	83.6628	1762.7859
AM-(10,0.1MEJ)	1.0107	407.8000	2.4859	468.9191	41.4123	3090.8000	83.1493	1642.9583
SFLAv1	0.7044	122.2000	1.1418	3.0698	25.0300	1602.0000	46.5210	10.4258
SFLAv2-1	0.7059	105.2000	1.0824	3.3634	26.3709	1616.0000	48.0496	10.7348
SFLAv2-2	0.6684	3.4000	0.6806	3.4590	24.8063	1077.0000	39.2543	11.0674
SFLAv2-2parallel	0.6693	0.0000	0.6693	2.3984	24.2903	1123.8000	39.3662	6.3348

	Newthyroid				Rand			
	Tasa_C	*Tasa_inf*	*Agr.*	T	*Tasa_C*	*Tasa_inf*	*Agr.*	T
COPKM	x	x	x	x	0.7500	0.0000	0.7500	0.2780
BL	12.8428	91.8000	14.5387	0.8972	0.8414	98.8000	1.2392	0.0876
BMB	12.0509	249.2000	16.6547	3.3726	0.7156	0.0000	0.7156	0.5932
ES	13.5906	52.8000	14.5660	0.8560	0.7156	0.0000	0.7156	0.2566
ILS_BL	13.5906	52.8000	14.5660	1.9664	0.7156	0.0000	0.7156	0.2500
ILS_ES	13.5048	169.2000	16.6306	1.0406	0.7156	0.0000	0.7156	0.3268
ILS_ESv2	14.1890	56.6000	15.2346	1.3328	0.7741	52.6000	0.9859	0.2390
AGE-CU	13.9083	1981.0000	50.5548	899.9752	2.6088	890.4000	6.2119	516.9353
AGE-CSF	15.4180	1090.6000	35.5930	906.5044	1.2264	316.4000	2.5068	527.9900
AGG-CU	15.3512	1140.4000	36.4475	759.7317	1.8450	570.4000	4.1532	428.3510
AGG-CSF	15.3210	892.2000	31.8259	740.3376	1.3435	387.0000	2.9095	432.6337
AM-(10,1.0)	15.0514	1379.0000	40.5615	830.9825	1.8289	569.4000	4.1330	486.1355
AM-(10,0.1)	15.5643	918.8000	32.5612	860.2795	1.3035	362.0000	2.7684	497.1874
AM-(10,0.1MEJ)	15.1769	1047.2000	34.5491	820.9502	1.2961	338.2000	2.6646	479.2806
SFLAv1	13.5087	348.2000	19.9414	5.5304	0.8624	120.0000	1.3456	2.9676
SFLAv2-1	13.9083	160.0000	16.8642	5.8110	0.8723	123.2000	1.3684	3.3590
SFLAv2-2	13.9373	110.2000	15.9732	5.8654	0.7156	0.0000	0.7156	3.3616
SFLAv2-2parallel	13.8190	117.8000	15.9953	3.7320	0.7156	0.0000	0.7156	2.2460

[6.2] Análisis comparativo:

Vamos a proceder a analizar los resultados obtenidos por los algoritmos estudiados en esta práctica, y a compararlos entre ellos y con los resultados obtenidos en las prácticas anteriores.

En primer lugar, observamos que el reinicio de poblaciones añadido al SFLA ha supuesto una mejora objetiva respecto del algoritmo inicial, siempre y cuando se emplee el segundo método de generación aleatoria de soluciones. Es probable, viendo los resultados, que el primer método no sea adecuado por el énfasis en la exploración que supone regenerar la población. Una generación completamente aleatoria tal vez se hubiese visto mejor apoyada por una intensificación en la explotación, que se ve dificultada al incluir este tipo de medidas tan radicales en favor de la exploración. El método de generación de pseudocentroides, por contra, ha logrado reducir de forma general la agregación obtenida por SFLA y homogeneizar ligeramente sus resultados.

La implementación paralela del SFLAv2-2 no solamente ha demostrado obtener resultados en la línea de su versión secuencial (lo esperado, puesto que los cálculos y el esquema seguido son los mismos en ambos casos), sino que ha logrado unas ganancias temporales de entre el 30% y el 75% de media, aspecto que parece crecer con el tamaño del dataset.

En lo que respecta a los resultados del SFLA comparados con los del resto de algoritmos, observamos que los resultados obtenidos por el SFLA y sus derivados para el problema del PAR son, con mucha diferencia, mejores que los de los algoritmos genéticos y meméticos de la práctica 2, a pesar de que puedan parecer semejantes en un primer momento. Si comparamos, por ejemplo, el caso del Ecoli con 10% de restricciones, el mejor de los AG/AM estudiados fue el CSF con una media de agregación de 80’7019, muy por detrás del SFLA con un 44’4463 de media.

El caso de los algoritmos basado en búsquedas locales y búsquedas locales iteradas o multiarranque no es tan destacado, pero sigue siendo notable que, a excepción del ES y el ILS-ESv2 de la práctica 3, el algoritmo de este tipo con la media más baja resulta ser la BL, aún más de 13 puntos por encima del SFLA, y 18 puntos de media por encima del SFLAv2-2. Estos últimos parecen presentar una varianza de soluciones bastante elevada (no muy diferente a la BL) pero, si bien recordamos que algoritmos multiarranque como la BMB paliaban la heterogeneidad de las soluciones, el SFLA (y especialmente el SFLAv2-2) han obtenido, en el caso del Ecoli, resultados mejores a la media de la BMB en la totalidad de sus ejecuciones. Este no ha sido el caso en los conjuntos más pequeños, pudiendo interpretarse de esto que una algoritmo como el SFLAv2-2, con un fuerte énfasis en la exploración, resulta menos adecuado para conjuntos pequeños, en los que pueda bastar con una explotación más intensa.

El COPKM, como se ha visto hasta ahora, es un algoritmo robusto y con buenos resultados. El SFLAv2-2 logra competir con él en cuanto a funciones objetivo, aunque se observa una curiosa tendencia por la cual el COPKM tiende a encontrar soluciones con mayores desviaciones medias y menores tasas de infactibilidad, en contraposición con todo el resto de algoritmos estudiados (salvo el ES, que continúa siendo el mejor en ambos flancos). No hay que olvidar, sin embargo, que mientras que el COPKM es un algoritmo específico para el problema del agrupamiento, el SFLA es un algoritmo de propósito general para optimización combinatoria y los resultados obtenidos resultan muy prometedores. Su abundancia de parámetros permite versatilidad a la hora de ajustarlo a un problema específico y eso lo ha hecho útil a la hora de afrontar una implementación para resolver el PAR.

En lo que respecta a los tiempos de ejecución, el SFLA presenta marcas entre un 50% y un 60% superiores a las de la BL. Esto no es de extrañar, puesto que el algoritmo memético requiere almacenar los elementos de la población de forma ordenada y gestionar las estructuras de datos necesarias para ello, mientras que la búsqueda local y sus derivados no necesitan mantener un control de las soluciones distintas a la actual (excepto de la mejor encontrada hasta el momento en el caso de las búsquedas iteradas, lo cual no supone un costo computacional muy elevado). Cabe destacar que el SFLAv2-2parallel obtiene tiempos inferiores a los de la BL.

[7] Conclusión:

En la presente práctica se ha realizado la implementación de diversos algoritmos basados en la metaheurística memética SFLA, o Shuffled Frog-Leaping Algorithm. Los hemos usado para obtener soluciones al problema del PAR en las mismas condiciones y conjuntos de datos que en las prácticas anteriores, obteniendo resultados relativamente buenos en comparación con los algoritmos de las prácticas anteriores, especialmente para el dataset Ecoli (el conjunto de datos más grande y argumentablemente más complejo de los utilizados), y mucho mejores que los de los algoritmos genéticos estudiados hasta ahora.

La versión mejorada del SFLA implementada en esta práctica ha demostrado poder alcanzar mejores soluciones que la versión original, aunque esto haya requerido aumentar la cantidad de parámetros necesarios para el algoritmo y hecho más compleja su implementación.

El SFLA también ha demostrado poseer una estructura fácilmente paralelizable, lo que nos ha permitido realizar una implementación concurrente del SFLA paralelo con ganancias satisfactorias en los tiempos de ejecución.

En conclusión, el algoritmo memético SFLA para optimización combinatoria ha resultado ser adecuado para el PAR, y tanto su abundancia de parámetros como su estructura paralela lo hacen una metaheurística prometedora para afrontar todo tipo de problemas de optimización discreta de forma eficiente.

[8] Referencias:

- *Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization (M Eusuff, K Lansey, F Pasha; 2006)*

- https://en.wikipedia.org/wiki/Unsupervised_learning

- *The Elements of Statistical Learning,* (Trevor Hastie, Robert Tibshirni, Jerome Friedman)