UNIVERSIDAD
DE GRANADA

Grupo 1

Metaheurísticas — Técnicas de Búsqueda basadas en Poblaciones para el Problema del Agrupamiento con Restricciones

---

Juan Mota Martínez
juanmotam@correo.ugr.es

27 de abril de 2020

Índice general

§0.1: Práctica 2: Técnicas de Búsqueda basadas en Poblaciones para el Problema del Agrupamiento con Restricciones

En esta práctica se nos pide resolver el problema del agrupamiento con restricciones, el cuál cosiste en la asignación de un conjunto de datos a diferentes clusters cumpliendo una serie de requisitios y asegurándonos que la distancia media entre los datos del cluster y el centro del mismo es mínima. Las restricciones o requisitos que se ha cumplir pueden ser de tipo ML(para dos datos ambos han de pertencer al mismo cluster) o CL(los datos han de estar en clusters diferentes), además de evitar a toda costa que uno de los clusters se encuentre vacío. Para realizar las comprobación se van a realizar las agrupaciones con los siguientes conjuntos de datos:

- **iris** Información sobre tres tipos de flor de Iris, contiene tres clusters o clases.
- **ecoli** Medidas sobre ciertos tipos de células, 8 clases.
- **newthyroid** Medidas sobre la glándula tiroides, 3 clases.
- **rand** Conjunto de datos artificial, 3 clases.

§0.1.1: Representación de los datos

Para representar datos y el cluster al que pertenecen se ha creado un vector de tamaño igual al número de datos a asignar, en cada componente del vector se asignará un número que indicará a que cluster pertenece, luego para representar que el dato 57 está en el cluster 2, en la posición 57 se almacenará el número 2, es por este motivo que los datos comienzan en el 0, para indicar que un dato aún no ha sido asignado se alamcena un -1. Estos datos se guardan en la clase Poblacion, la cuál además del vector mencionado anteriormente, guarda también los centroides de cada cluster, representados como una matriz de floats de tamaño variable para acomodar distintos números de clusters representados como vectores n-dimensionales.

Para inicializar un objeto Población es necesario indicar: el número máximo de datos que va a tener que almacenar, el número de dimensiones que han de tener los centroides, el número de clusters, y los valores mínimos y máximos que alcanzan las coordenadas de los datos que se van a almacenar, estas dos últimas variables son necesarias para la generación aleatoria de centroides. Se devuelve un objeto con un conjunto de datos sin clusters asignados (todos los valores del vector son -1) y un número de centroides con valores aleatorios comprendidos entre el mínimo y máximo suministrado.

Para representar las restricciones se está usando una lista de tuplas, las tuplas cosisten en dos enteros y un double, los enteros indican el índice de los datos y el double la relación entre ellos, 1 ML y -1 CL, cuando se leen los datos no se tiene encuenta las relacions de 0, ya que no nos interesan para el calculo del la infactivilidad, y las restricciones del tipo (45, 45 ,1), al tratarse del mismo dato, siempre va a pertenecer a su mismo cluster, de forma que para reducier el tiempo que se tarda en recorrer la lista, se han ignorado estos valores presentes en los ficheros de restricciones. Se ha implementado como una lsita de tuplas ya que se trata del contenedor mas rápido y no se existe un problema al no poder acceder a una componenete específica ya que siempre vamos a recorrer todas las tuplas para calcular la infactivilidad de la solución.

Leemos las tuplas del fichero de restricciones de la siguiente forma:

```
Inicio(1)
   #Inicializamos el fichero de restricciones
   inicializar(fich)
   val1 = 0
   val2 = 0

   Mientras no estemos en el final del fichero:
   Inicio(2)
      #Leemos la linea por linea el fichero
      inicializar(linea)
      Mientras no estemos en el final de la linea:
      Inicio(3)
         #Leemos uno por uno los datos de la linea
         inicializar(dato)
         si el dato != 0:
            #insertamos la tupla
            restricciones.insertar(tupla(val1, val2, dato))

         val2++
      Final(3)
      val1++
      val2=0
   Final(2)
Final(2)
```

Los datos se representan mediante una matriz de floats, cada fila de la matriz se corresponde con un dato y sus columnas son los valores para cada una de sus dimensiones, no se ha implementado mediante una lista como las restricciones, puesto que varios de los algoritmos nos obligan a recorrer los datos de forma aleatoria, es por ello que vamos a necesitar poder acceder a un dato concreto. Al igual que las restricciones hemos de leer los datos desde un fichero, y almacenarlos en una variable denominada datos del programa, el código de lectura es el siguiente:

```
Inicio(1)
   #Inicializamos el fichero de datos
   inicializar(fichero)

   Mientras no estemos en el final de fichero:
   Inicio(2)
      #Creamos un vector de floats para almacenar las coordenadas
      inicializar(vector)
      #Leemos linea por linea
      Mientras no estemos en el final de la linea:
      Inicio(3)
         #Leemos el dato y los introducimos en el vector
         inicializar(dato)
         vector.insertar(dato)
      Final(3)
      #Una vez hemos terminado insertamos el vector en la matriz
```

```
      datos.insertar(vector)
   Final(2)
Fin(1)
```

Para almacenar el conjunto de Poblaciones se ha creado la clase PAR, la cuál consiste en un vector de poblaciones, los cuáles se inicializan todos con los datos sin cluster asignado, para crear un objeto de clase PAR es necesario suministrar el número de poblaciones además de los datos necearios para crear un objeto de la clase Población. Mientras que la clase Población se centra en métodos para modificar y calcular datos, la clase PAR se centra en la obtención de datos, además de la generación de poblaciones aleatorias, este método asigna a cada dato un valor aleatorio comprendido entre 0 y el númedo de clusters suministrados. Para evitar que puedan generarse poblaciones no aptas (uno de los clusters está vacío) se le aplica después un operador de reparación, el cuál comprueba si uno de los tamaños de los clusters es igual a 0, y para este caso selecciona un valor aleatorio del vector y le asigna el número del cluster vacío, el operador de reparación es similar al siguiente pseudocódigo:

```
Inicio(1)
   Para cada pob en el conjunto de poblaciones:
   Inicio(2)
      Si el tamaño de un cluster c de pob es igual a 0:
         #Generamos un dato aleatorio entre 0 y el número de datos máximo
         indice = aleatorio $\%$ datos$\_$maximos
         #Asignamos el dato al cluster vacío
         cromosoma[indice] = c
   Final(2)
Final(1)
```

El calculo de la infactivilidad se realiza mediante 2 funciones, una pensada para calcular error generado al introducir un dato en un cluster específico y otra que calcula error completo de la población. Se trata de un método de Poblacion al que se le suministran la lista de restricciones y se recorren desde principio a fin. Para calcular el error del cluster específico se indican también el dato y en que cluster se desea introducir, entonces se recorre la lista ignorando las tuplas en las que el primer valor es menor que el segundo (esto es para tratar de evitar que se repitan infactivilidades, ya que en la lista se almacenan todas las restricciones del fichero), cuando primer valor de la tupla se corresponde con el de suministrado se comprueba el a que cluster pertenece el segundo valor de la tupla, en caso de no tener un cluster asignado, se ignora, luego se comprueba si se trata de un restricción ML y CL y se actua en consecuencia. Para calcular la infactivilidad total se sigue un procedimiento similar, se recorren todas las tuplas de restricciones, ignorando como en el caso anterior aquellas en las que el segundo valor sea mayor que el primero, y se comprueba a que cluster pertenece el primer valor de la tupla y el segundo, después se revisa el tipo de restriccion y si se cumple, si no se cumple el error a devolver se aumenta en uno. Ambas funciones devuelven un entero que indica la infactivilidad parcial o total de la población.

```
Inicio(1)
   errores = 0
   Para cada tupla en el conjunto de restricciones:
   #Asegurándonos que tupla(0) > tupla(1) para no contar dos veces la misma restricción.
   Inicio(2)
      cluster1 = calculaCluster(tupla(0))
      cluster2 = calculaCLuster(tupla(1))

      Si cluster1 == cluster2 y tupla(2) == -1:
         errores++
      Si cluster1 != cluster2 y tupla(2) == 1:
         errores++

   Final(2)
   return errores
Final(1)
```

La función que calcula el error parcial es idéntica, pero se le suministra el dato que se quiere insertar y el cluster a insertar, de forma que se procede así:

```
Inicio(1)
```

```
   errores = 0
   Para cada tupla en el conjunto de restricciones:
   #Haciendo la misma comprobación que en la función anterior
   Inicio(2)
      si clusterAintroducir == tupla(0):
         cluster2 = calcularCluster(tupla(1))
         #Se comprueba también que cluster2 no es igual a -1
         Si cluster2 == clusterAintroducir y tupla(2) == -1:
            errores++
         Si cluster2 != clusterAintroducir y tupla(2) == 1:
            errores++

   Final(2)
   return errores
Final(1)
```

Se calcula la distancia intracluster recorriendo el vector de datos y comprobando si el valor almacenado en un índice específico es igual al número del cluster que nos interesa, entonces, se calcula la distancia entre el centroide del cluster y la posición del dato, guardamos la sumatoria de estos valores para después dividirla por el número de datos que hay almacenados en el cluster.

La desviación general es la distancia intracluster media de todos los clusters del problema.

§0.2: Algoritmos Genéticos

Un algoritmo genético es un tipo de algoritmo evolutivo que realiza operaciones sobre un conjunto de soluciones factibles, denominadas cromosomas, sobre las cuales se aplican las siguientes operaciones: torneo binario, cruce y mutación, para combinar la exploración del conjunto de soluciones junto con la explotación de las mismas. El torneo binario favorece la explotación ya que consiste en sleccionar dos cromosomas cualesquiera y sustituir el peor de ello por el otro, de esta forma eliminamos soluciones mal valoradas, obteniendo una población con una valoración media superior, sin embargo al eliminar estas soluciónes reducimos las exploración del algoritmo causando que se puedan alcanzar máximos locales. Las operaciones de cruce y mutación favorecen la exploración del conjunto del soluciones, ya que el primero toma dos cromosomas y genera un hijo a partir de ellos que comparte genes con ambos, los operadores de cruce usados en exta práctica son segmento fijo y uniforme, lo cuáles se explicarán en más tarde. El algoritmo de mutación se asemeja a las propiedades de los seres vivos de generar hijos con características no heredadas de los padres sino producidas por un accidente. Sin embargo esto puede causar que sustituyamos soluciones buenas por nuevos con peor valoración.

El torneo binario se ha implementado de la siguiente forma, se realizan un número de comparaciones igual al número de poblaciones existentes, entonces para cada comparación se toman dos cromosomas aleatorios y se comparan sus valoraciones, la mejor valorada sustituye a la otra y se vuelve a realizar una nueva comparación con la población resultante, esto puede causar que un mismo cromosoma se enfrente varias veces contra otros, pero también garantiza que vamos a eliminar poblaciones poco favorables.

```
Inicio(1)
   Mientras i sea menor que el número de poblaciones:
   Inicio(2)
      #Seleccionamos aleatoriamente dos poblaciones asegurándonos que no son la misma
      pob1 = obtenerPoblacionAleatoria()
      pob2 = obtenerPoblacionAleatoria()

      #Calcula valoración devuelve el agregado de la función.
      val1 = calculaValoracion(pob1)
      val2 = calculaValoracion(pob2)

      Si val1 >val2:
         pob2 = pob1
      en otro caso:
```

```
            pob1 = pob2
         i++
      Final(2)
   Final(1)
```

El operador de mutación actúa de la siguiente manera, se recorren todos los cromosomas, y cada uno de sus genes tiene un 0.001 % posibilidades de mutar, en caso de mutar, el valor del gen pasa a pertencer a un cluster, aunque esto cause que la valoración del cromosoma disminuya. Una vez se han aplicado estos 3 operadores obtenemos una nueva generación, con la inteción de conservar el elitismo antes de aplicar estos cambios sobre la población se selecciona la mejor solución de la generación anterior, y en caso de que esta fuese eliminada por el cruce o la mutación se sustituye por la peor solución de la nueva generación. También es posible que se generen cromosomas infactibles (uno de los clusters tiene tamaño 0), por lo que es necesario aplicar el algortimos de reparación mecionado en la sección anterior para arreglar los cromosomas infactibles.

EL pseudocódigo del operador de mutación:

```
Inicio(1)
   Para cada pob in Poblaciones:
   Inicio(2)
      Para cada gen en pob:
      Inicio(3)
         #Donde mutar gen puede asignar un cluster al gen con una probabiliad de 0.001
         Mutar(gen)
      Final(3)
   Final(2)
Final(1)
```

El algoritmo parará cuando se hayan realizado 100000 evaluaciones, se ha considerado una evaluación: el torneo binario, la generación de un hijo, una mutación y la reintroducción de la mejor solución porque ha sido eliminada en algún momento. En función de cómo se generen los hijos y la cantidad de mutaciones que se produzcan, el número de evaluaciones por generación puede variar, en las siguientes secciones se abordan la implementación de los dos algoritmos genéticos que se nos ha pedido implementar.

La forma de calcular la valoración de cada cromosoma viene dada por la siguiente función objetivo, dónde lambda es igual a al cociente entre la distancia máxima del conjunto de datos y el número de restricciones presentes en el problema

$$f(x) = \overline{C} + (infactibilidad * \lambda)$$

§0.2.1: Algoritmo Genético Generacional

Ambos algoritmos se diferencian en la forma en que generan los hijos, el generacional para una probabilidad de cruce toma dos padres, los cuáles generan dos hijos que los sutituyen. La probabilidad de cruce que se nos da es de un 70 %, lo que implica que para una población de 50 cromosomas, de las cuáles son posibles 25 parejas deberíam producirse de media 17.5 cruces, para eliminar la probabiliad se ha decidio que siempre se produzcan 17 cruces de la siguiente forma: se seleccionan aleatoriamente dos cromosomas, los cuáles han de ser distintintos, tras esto aplicamos el operador de cruce deseado y obtendremos 2 nuevos cromosomas creados a partir de los genes de ambos padres, los cuáles sustituyen a los progenitores y tras esto se repite el proceso hasta alcanzar los 17 cruces por generación.

Este algoritmo favorece la exploración respeto a la explotación ya que los hijos no siempre serán mejores que los padres, pero al generar soluciones más diferentes evitaremos que el algoritmo se estanque en máximos locales. El pseudocódigo del AGG es el siguiente:

```
Inicio(1):
   Mientras i sea menor que 17:
   Inicio(2)
      padre1 = seleccioandorAleatorioPadre()
      padre2 = seleccionadorAleatorioPadre()
```

```

    #Nos aseguramos que los padres no son el mismo, en caso contrario buscamos otro
    padre2

    hijo1 = OperadordeCruce(padre1,padre2)
    hijo2 = OperadordeCruce(padre2,padre1)

    #Borramos los padres de la población e introducimos los hijos
  Final(2)
Final(1)
```

§0.2.2: Algoritmo Genético Estacionario

Al diferencia que el anterior, este algoritmo selecciona los dos mejores padres de la población y genera dos nuevos hijos los cuáles entonces compiten con los peores cromosomas de la población actual, y en caso de poseer una valoración mejor que estos los sustituyen, la forma en la que se ha implementado es la siguiente: el primero de los hijos compite contra la peor solución de la población y en caso de ganar la sustituye, el segundo hijo entonces se enfrenta al segundo peor cromosoma y se repite el mismo procesos que con el hijo anterior, por la forma en la que está implementado existe la posibilidad de que el segundo hijo aún siendo mejor que el primero no pase a la nueva generación puesto que su valoración es pero que la del cromosomas contra el que se enfrenta. Las posibilidad que esto ocurra son muy pequeñas y para que se diese la diferencia de mejora entre el hijo1 y el hijo dos debería ser bastante pequeña por lo que no se ha introducido ningún método pasa solucionarlo.

Al contrario que el anterior este algoritmo favorece la explotación con respecto a la exploración, ya que siempre vamos a obtener una población con mejor o igual que la anterior. Como el número de evaluaciones se cuenta en parcialmente en función del número de cruces que se producen, el AGE va crear muchas más generaciónes en 100000 evaluaciones que AGG, el código de la función es el siguiente:

```
Inicio(1)
  #Mejores cromosomas devuelve a los n cromosomas mejors valorados de la población
  padre1, padre2 = MejoresCromosomas(2)

  hijo1 = OperadordeCruce(padre1, padre2)
  hijo2 = OperadordeCruce(padre2, padre1)

  #Al contrario que el anterior devuelve los n peores cromosomas
  peor1, peor2 = PeoresCromosomas(2)

  #Calculamos sus valoraciones y sustituimos en consecuencia
Final(1)
```

§0.2.3: Algoritmo de cruce SF y UN

Se han implementado dos tipos del algoritmos de cruce para la generación de hijos, segmento fijo y uniforme:

Uniforme, para dos padres p_1 y p_2 con n genes cada uno, selecciona $n/2$ genes aleatorios del primer padre y los introduce en el hijo, después termina de completar el cromosoma usando los $n/2$ restantes del segundo padre. La forma en la que se ha implemenado es la siguiente, se ha generado un vector de índices para representar cada gen, se ha desordenado aleatoriamente y se han recorrido sus $n/2$ primeras posiciones, comprobando a qué valor corresponde el índice obtenido en p_1 y copiandolo en el hijo, acto seguido se ha continuado recorriendo el vector de índices pero copiando loas valores de p_2 esta vez. Como se ha explicado en las secciones anteriores se llama al operador de cruce 2 veces con los mismos padres, intercambiando los progenitores para que en el segundo, cruce quien ocupase la posición de p_1 en el primero sea ahora p_2.

Segmento fijo genera dos valores aleatorios inicio del segmento y tamaño del segmento, inicio del segmento indica a parte de qué gen del primer padre se va a copiar el el hijo, y tamaño el número de genes a partir del primero que se van a copiar, es decir para inicio $= 13$ y tamaño igual a 20, se van a copiar de p_1 todos los genes comprendidos entre 13 y 33. Los genes restrantes se toman del segundo padre. La implementación de este operador es similar a su explicación, se han generado dos valores inicio y tamaño, a tamaño le sumamos inicio y así obtenemos la posición final del segmento, ahora recorriendo los ínidices comprobamos si estos se encuentran en este intervalo, en caso positivos tomamos los valores de los genes de p_1 y en el caso contrario de p_2.

§0.3: Algoritmo Memético

Los algoritmos evolutivos son malos explotadores, como se ha podido comprobar anteriormente, los algoritmos genéticos pueden provocar que la nueva generación tenga una valoración media inferior a la de anterior, a pesar de los procedimientos empleados para solucionar este defecto los algoritmos suelen tardar varias generaciones en converger, es por ello que para paliar esta característica se ha hibridado un algoritmo similar a la búsqueda local de la práctica anterior con uno algoritmos genéticos diseñados en esta. El procedimiento es el siguiente, cada x generaciones aplicaremos nuestra búsqueda local suave sobre la población actual obteniendo siempre una nueva generación con una valoración media mejor que su predecesora. Usamos una búsqueda local suave ya que no queremos eliminar completamente la exploración que aportan los algoritmos evolutimos, sólo queremos que los cromosomas generados cada cierto número de generaciones sean mejores para asegurarnos de que las soluciones se acerquen a converger a un máximo golbal.

La búsqueda local suave se ha implementado de la siguiente forma, se recorren todos los genes del cromosoma una única vez probado a asignar el gen a los clusters disponibles comprobando si la solución mejora, siempre nos quedaremos con la asignación que nos proporcione una solución mejor valorada, y partiendo de este nuevo cromosoma continuaremos recorriendo sus genes y aplicando mejoras dónde sea posible, como la búsqueda local suave puede causar que se queden clusters vacíos le aplicaremos el operador de reparación al terminar.

```
Inicio(1)
   Para todo gen en Poblacion:
   Inicio(2)
      Para todo c en clusteres:
      Inicio(3)
         #Calcular Valoracion devuelve el agregado de un cromosoma
         val$_$vieja =CalcularValoracion()
         datos[gen] = clus
         val$\_$nueva = CalcularValoracion()
         #Si val$\_$vieja es mejor que la nueva revertimos el cambio, en caso contrario
         lo conservamos
      Final(3)
   Final(2)
Final(1)
```

Aplicaremos esta búsqueda local cada 10 generaciones pero los cromosomas sobre los cúales aplicaremos esta mejora variarán en función de la hibridación que queramos obtener, a continuación las 3 opciones que se nos pedía en esta práctica, se han hibirdado estos algoritmos de optimización con el AGG-UN ya que ha sido el que mejores soluciones proporcionaba en relación al tiempo:

- **AM(10,1.0)** siempre aplicamos la BL sobre todos los cromosomas de la población.
- **AM(10,0.1)** como la anterior, cada diez generaciones se aplica la búsqueda local, pero en este caso sólo sobre el 10 % de la población, como siempre estamos trabajando sobre una población de 50 individuos deberíamos obtener siempre 5 individuos mejor valorados.
- **AM(10,0.1mej)** En este caso aplicaremos una búsqueda local sobre los 5 mejores cromosomas de la población actual.

§0.3.1: Análisis de los resultados

AM(10,0.1mej) 10 %

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882,00	0,04	0,00	0,04	36,89	1,51	235,00	28,08	197,78
572718921,00	0,03	0,00	0,03	36,58	1,92	175,00	21,71	197,29
34737829,00	0,04	0,00	0,04	36,48	1,48	415,00	48,41	196,95
28388992,00	0,03	0,00	0,03	36,56	1,32	208,00	24,84	196,97
9987479,00	0,03	0,00	0,03	36,63	1,59	335,00	39,48	196,85
Media	0,03	0,00	0,03	36,63	1,56	273,60	32,51	197,17

	Rand				Newthyroid			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882,00	0,63	0,00	0,63	35,99	1,10	0,00	1,10	70,84
572718921,00	0,63	0,00	0,63	35,74	1,09	0,00	1,09	72,15
34737829,00	1,09	0,00	1,09	38,50	1,04	0,00	1,04	70,98
28388992,00	0,92	0,00	0,92	35,96	0,93	0,00	0,93	71,09
9987479,00	0,03	0,00	0,03	36,63	1,70	0,00	1,70	70,32
Media	0,66	0,00	0,66	36,56	1,17	0,00	1,17	71,08

AGE-UN 10 %

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,0191203	0	0,0191203	81,9677	1,76195	146	18,2739	397,229
572718921	0,0181324	0,00	0,0181324	81,8512	1,22644	723,00	82,9943	394
34737829	0,0220699	0,00	0,0220699	81,9002	1,55436	351	41,2508	394,665
28388992	0,02605	0,00	0,02605	81,6618	1,31887	858	98,1687	393,747
9987479	0,0198619	0,00	0,0198619	81,945	1,44524	232,00	27,683	391,945
Media	0,02	0,00	0,02	81,87	1,46	462,00	53,67	394,32

	Rand				Newthyroid			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,50084	0	0,50084	80,8721	0,93427	0,00	0,93427	155,737
572718921	0,50084	0,00	0,50084	80,4402	0,545748	0,00	0,545748	155,893
34737829	0,50084	0,00	0,50084	87,2822	0,901431	0,00	0,901431	156,787
28388992	0,50084	0,00	0,50084	81,1147	0,783093	0,00	0,783093	156,425
9987479	0,50084	0,00	0,50084	83,3214	1,69897	0,00	1,69897	70,3177
Media	0,50	0,00	0,50	82,61	0,97	0,00	0,97	139,03

AGE-SF 10 %

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,0201047	0	0,0201047	81,9023	1,75291	354	41,7886	396,79
572718921	0,0228752	0,00	0,0228752	83,2344	1,28659	344	40,1914	398,124
34737829	0,0228134	0,00	0,0228134	82,2064	1,31926	342,00	39,9978	395,221
28388992	0,0216523	0,00	0,0216523	81,9379	1.43507	263	31.1791	396,101
9987479	0.017579	0,00	0.017579	82.0853	1.11956	602,00	69.2029	394,496
Media	0,02	0,00	0,02	82,32	1,45	381,00	40,66	396,15

	Rand				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,50084	0	0,50084	80,3442	0,720677	0,00	0,720677	155,927
572718921	0,50084	0,00	0,50084	80,8954	0,870489	0,00	0,870489	156,052
34737829	0.50084	0,00	0.50084	87.3754	0.724691	0,00	0.724691	156,812
28388992	0.50084	0,00	0.50084	81.1147	0.951239	4,00	1.34194	157,027
9987479	0.50084	0,00	0.50084	81.7697	0.811966	0,00	0.811966	155,675
Media	0,50	0,00	0,50	80,62	0,80	0,80	0,80	156,30

AGG-UN 10 %

	Iris				**Ecoli**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,0333167	0	0,0333167	31,2705	1,43823	206	24,7358	162,591
572718921	0,0441348	0,00	0,0441348	30,974	1,63514	295	34,9982	163,784
34737829	0,0437396	0,00	0,0437396	30,8651	1,17193	540,00	62,2434	163,032
28388992	0.0312388	0.00	0.0312388	30.9025	1.22203	405,00	47,03	163,942
9987479	0.044728	0,00	0.044728	31.0924	1.26062	361	42,088	163,414
Media	0,04	0,00	0,04	31,04	1,42	361,40	42,22	163,35

	Rand				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,88264	0	0,88264	30,2583	1,07315	0,00	1,07315	58,0157
572718921	1,40999	0,00	1,40999	30,3468	1,39248	0,00	1,39248	59,2617
34737829	1.07748	0.00	1.07748	32.5831	1.10415	0,00	1.10415	60.0069
28388992	0.763987	0.00	0.763987	30.3482	1.15193	0,00	1.15193	58,76
9987479	1.02009	0.00	1.02009	30.5981	1.12745	0,00	1.12745	59.7537
Media	1,15	0,00	1,15	30,30	1,23	0,00	1,23	58,68

AGG-SF 10 %

	Iris				**Ecoli**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,0325763	0	0,0325763	31,015	1,21067	370	43,0559	164,787
572718921	0,0240125	0,00	0,0240125	30,8001	1,62145	383	44,9369	161,528
34737829	0,0346339	0	0,0346339	30,7519	1,44211	467	54,2576	160,92
28388992	0,0385425	0	0,0385425	30,6933	1,4585	273	32,3335	162,529
9987479	0,0375547	0	0,0375547	30,8553	1,39178	287	33,8501	162,699
Media	0,03	0,00	0,03	30,82	1,42	356,00	41,69	162,49

	Rand				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	1,37992	0	1,37992	30,2243	0,978537	0,00	0,978537	58,7411
572718921	0,917627	0,00	0,917627	30,1178	1,30765	0,00	1,30765	58,7366
34737829	0,78344	0	0,78344	32,4841	0,0413767	0	0,0413767	53,8518
28388992	0,917627	0	0,917627	30,4112	1,09896	0	1,09896	58,9346
9987479	1,27457	0	1,27457	30,3912	1,07182	94	10,2532	59,0179
Media	1,05	0,00	1,05	30,73	0,90	18,80	2,74	57,86

AM(10,0.1mej) 20 %

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,0366544	0	0,0366544	63,5132	3,43749	185	24,3601	345,349
572718921	0,0291283	0,00	0,0291283	63,9925	2,82175	177	22,8396	343,95
34737829	0,0365172	0,00	0,0365172	63,4811	2,10563	240	29,2485	344,538
28388992	0,0380916	0,00	0,0380916	62,7203	1,32807	736	84,5662	372,31
9987479	0,0391612	0,00	0,0391612	62,8155	1,58885	502	58,3627	345,995
Media	0,04	0,00	0,04	63,30	2,26	368,00	43,88	350,43

	Rand				Newthyroid			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,788595	0	0,788595	62,0055	1,03281	0	1,03281	127,526
572718921	0,75712	0,00	0,75712	62,0806	1,29868	0,00	1,29868	127,07
34737829	0,75712	0,00	0,75712	75,712	0,865465	0,00	0,865465	127,126
28388992	0,50084	0	0,50084	61,8896	0,264819	0,00	0,264819	127,477
9987479	0,924422	0,00	0,924422	61,9732	0,95835	0,00	0,95835	124,524
Media	0,73	0,00	0,73	65,41	0,85	0,00	0,85	126,55

AGE-UN 20 %

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,0236006	0	0,0236006	142,607	2,20417	162	20,5256	699,129
572718921	0,0237598	0,00	0,0237598	143,188	1,71257	461	53,8495	692,569
34737829	0,0220699	0	0,0220699	142,9002	2,05842	218	26,7132	696,348
28388992	0,0145968	0,00	0,0145968	141,022	1,5622	431	50,3063	726,856
9987479	0,0178306	0,00	0,0178306	143,019	1,75206	380	44,7283	700,945
Media	0,02	0,00	0,02	142,53	1,77	372,50	43,90	704,18

	Rand				Newthyroid			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,50084	0	0,50084	140,56	0,754922	0,00	0,754922	279,629
572718921	0,50084	0,00	0,50084	140,668	0,928058	0,00	0,928058	280,975
34737829	0,50084	0,00	0,50084	140,78	0,896762	0,00	0,896762	280,558
28388992	0,50084	0,00	0,50084	140,631	0,683426	0,00	0,683426	279,279
9987479	0,50084	0,00	0,50084	141,79	0,952964	0,00	0,952964	279,307
Media	0,50	0,00	0,50	140,97	0,87	0,00	0,87	280,03

AGE-SF 20 %

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,0237409	0	0,0237409	142,957	1,87738	387	45,6452	697,588
572718921	0,0272129	0,00	0,0272129	142,929	1,47728	613	70,8047	692,444
34737829	0,0228134	0,00	0,0228134	144,2064	1,91	239	28,9398	694,147
28388992	0,0216523	0,00	0,0216523	141,618	1,99605	333	39,6568	725,196
9987479	0,0247699	0,00	0,0247699	143,006	2,46588	258	31,6445	701,691
Media	0,02	0,00	0,02	142,94	1,95	366,00	43,34	702,21

	Rand				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,50084	0	0,50084	140,535	0,827315	0,00	0,827315	280,179
572718921	0,50084	0,00	0,50084	142,128	0,868369	0,00	0,868369	280,549
34737829	0,50084	0,00	0,50084	142,78	0,704749	0,00	0,704749	279,722
28388992	0,50084	0,00	0,50084	141,25	0,802923	0,00	0,802923	280,182
9987479	0,50084	0,00	0,50084	141,098	0,849353	0,00	0,849353	279,665
Media	0,50	0,00	0,50	141,56	0,81	0,00	0,81	280,06

AGG-UN 20 %

	Iris				**Ecoli**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,0395478	0	0,0395478	53,5787	1,10848	579	66,5906	286,133
572718921	0,0455242	0	0,0455242	53,4796	1,65268	432	50,5098	286,232
34737829	0,0437396	0,00	0,0437396	63,8651	1,46895	389	45,463	287,142
28388992	0,0456766	0,00	0,0456766	63,5741	1,79753	460	53,8213	304,614
9987479	0,0299215	0,00	0,0299215	52,5913	2,07046	186	23,1062	285,146
Media	0,04	0,00	0,04	57,42	1,62	409,20	47,90	289,85

	Rand				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,78344	0	0,78344	52,2995	1,19102	0,00	1,19102	105,898
572718921	0,69916	0,00	0,69916	53,1276	1,24221	0,00	1,24221	105,041
34737829	0,69916	0,00	0,69916	53,1276	1,42305	0,00	1,42305	106,213
28388992	0,65448	0,00	0,65448	52,6455	1,34398	0,00	1,34398	105,454
9987479	0,75712	0,00	0,75712	52,5011	0,990932	0,00	0,990932	104,176
Media	0,72	0,00	0,72	52,74	1,24	0,00	1,24	105,36

AGG-SF 20 %

	Iris				**Ecoli**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,0392823	0	0,0392823	53,4486	1,36409	1760	200,412	284,967
572718921	0,0310814	0,00	0,0310814	53,5596	1,8475	690	79,8832	281,903
34737829	0,0346339	0,00	0,0346339	60,7519	1,71979	448	52,3865	281,735
28388992	0,0240749	0,00	0,0240749	52,794	2,01572	315	37,6407	303,327
9987479	0,0369899	0,00	0,0369899	52,7656	1,46792	437	50,8905	285,013
Media	0,03	0,00	0,03	54,66	1,68	730,00	84,24	287,39

	Rand				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	1,07748	0	1,07748	52,2651	1,07556	0,00	1,07556	104,238
572718921	0,50084	0,00	0,50084	52,4356	1,09007	0,00	1,09007	105,773
34737829	0,50084	0,00	0,50084	52,4356	1,14045	0,00	1,14045	104,696
28388992	0,88428	0,00	0,88428	52,3877	1,00315	0,00	1,00315	104,178
9987479	0,82648	0,00	0,82648	52,0231	1,19531	0,00	1,19531	103,902
Media	0,76	0,00	0,76	52,31	1,10	0,00	1,10	104,56

Se puede observar que exceptuando para los resultados del conjunto ecoli, la infactivilidad generada para la mejor solución de cada una de la poblaciones es prácticamente 0, además de que la distancia intracluster es ínfima para los conjuntos de datos, aunque esto puede deberse a que se ha eliminado las operaciones de raíces cuadradas y elevaciones al cuadrado con la intención de disminuir el tiempo de cálculo, con respecto a los tiempos hay una gran diferencia entre los conjuntos con un pequeño número de datos como rand e iris y aquellos que son significativamente más amplios como ecoli, de esto podemos deducir que el algoritmo de cálculo de restricciones no está correctamente optimizado causando que para grandes cantidades de datos y restricciones el tiempo necesario para realizar los cálculos crezca exponencialmente.

Para las tablas con restricciones de 20 de Rand, se puede observar que muchos de los resultados con distintos algoritmos y semillas devuelven los mismos valores como mejor solución, indicando que es muy probable que se trate del óptimo, es posbile que existan más similitudes pero esta es la primera que se ha visto y se ha querido mencionar.

§0.4: Correción de la práctica 1

Se han podido arreglar varios de los fallos que presentaba la primera práctica en esta sección se recogerán los principales cambios, la representación de los datos es la misma que para la práctica 2, pero en este caso no se hace uso del la calse PAR, tan sólo de las funciones de Población y otras funciones mencionadas en con anterioridad en el documento, necesariaspara una correcta ejecución.

§0.4.1: Greedy

Greedy parte de un población con clusteres vacío y recorre cada dato asignándolo primero al cluster que menos restricciones vulnere y en caso de que exista un empata al que se enceuntre más recano con respecto al dato, tras completar la asignación se realculan los centroides se vacían los datos y se repite el proceso hasta que las diferencias entre las distnacias intracluster de la población inicial y la nueva obtenido tras volver a recorrer los datos aleatoriamente sea menor que 0.05. El código de la función es muy simple:

```
Inicio(1)
   Para cada dato en Datos
   Inicio(2)
   #Datos se ha desordenado aleatoriamente
      Para cada c en clusters:
         #Comprobamos cuál provoca el menor número de infactivilidades

      Para cada c en clusters:
         #Probamos cada cluster buscando el más cercano al dato y asegurándonos que
         #incluple el meno número de infactivilidades obtenido anteriormente

      #Asignamos el dato y continuamos
   Final(2)
Final(1)

#Vaciamos los datos
#Recalculamos clusteres
#Comprobamos que la diferencia entra las distancias intracluster no es menor que 0.05
y repetimos desorndenando otra vez los Datos
```

§0.4.2: Búsqueda local

Usando el generador de poblaciones aletorias y aplicando después el algoritmo reparador sobre él, se parte de una solución inicial y se recorren todos los datos, en orden aleatorio, probando a asignar a cada uno de ellos un cluster distinto, en caso de que se produzca una mejora, se toma la nueva población generada y se vuelve a tratar de mejorar, cada intento de mejora se considera una evaluación, búsqueda local continúa tratando de mejorar la población hasta que se cumplan 1000000 iteraciones, como en los algoritmos genéticos o se evalue todo el vecindario de soluciones y no se produzca ningún cambio en la población, es decir, se alcance un máximo local. Búsqueda local tiene un funcionamiento similar a:

```
Inicio(1)
Mientras no se cumpla la condición de parada continuamos:
   Inicio(2)
      Para cada d en datos:
      Inicio(3)
         Para cada c en cluster:
         #Vemos si existe mejor, en ese caso, lo inciamos y repetimos desde Inicio(2)
      Final(3)
   Final(2)
   #Se ha recorrido todo el vecindario sin mejor o se han alcanzado las evaluaciones máximas
   #luego paramos
Final(1)
```

§0.5: Análisis de los resultados

Greedy 10 %

	Iris				**Ecoli**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,0422314	0	0,0422314	0,226019	4,23911	26	7,17959	9,69278
572718921	0,042503	0	0,042503	0,125442	2,40	46,00	7,60	5,86
47389829	0,06	0,00	0,06	0,13	2,53	70,00	10,44	5,11
28388992	0,04	6,00	0,16	0,12	1,58	259,00	30867,00	6,82
348427479	0,05	0,00	0,05	0,16	1,82	199,00	24,32	8,20
Media	0,05	1,20	0,07	0,15	2,51	120,00	6183,31	7,14

Rand				**Newthyroid**			
Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
0,915755	0	0,915755	0,124875	0,0592128	0	0,0592128	0,241551
2,35	0,00	2,35	0,12	1,50	66,00	7,94	0,41
2,02	0,00	2,02	0,12	1,73	2,00	1,92	0,43
1,07	0,00	1,07	0,12	1,35	0,00	1,35	0,85
0,81	0,00	0,81	0,16	2,18	84,00	10,38	0,40
1,43	0,00	1,43	0,13	1,36	30,40	4,33	0,47

Greedy 20 %

	Iris				**Ecoli**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,0592128	0	0,0592128	0,241551	3,64198	32	7,26103	8,89965
572718921	0,04	0,00	0,04	0,23	2,58	18,00	4,61	12,04
47389829	0,08	0,00	0,08	0,24	5942,00	0,00	5942,00	6,39
28388992	0,05	0,00	0,05	0,24	4,37	16,00	6,18	22,31
348427479	0,04	0,00	0,04	0,23	8,18	3,00	8,52	11947,00
Media	0,05	0,00	0,05	0,24	1192,16	13,80	1193,72	2399,33

	Rand				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,915755	0	0,915755	0,245616	1,58951	0	1,58951	0,827824
572718921	2,35	0,00	2,35	0,23	1,53	0,00	1,53	0,79
47389829	2,02	0,00	2,02	0,22	1,71	0,00	1,71	0,60
28388992	4,37	16,00	6,18	22,31	1,49	0,00	1,49	0,60
348427479	0,81	0,00	0,81	0,23	2,13	0,00	2,13	0,76
Media	2,10	3,20	2,46	4,65	1,69	0,00	1,69	0,72

BL 10 %

	Iris				**Ecoli**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,03	0,00	0,03	2,96	0,374872	2982	337,63	53,07
572718921	0,03	0,00	0,03	3,90	0,39	3030,00	343,07	52,19
34737829	0,03	0,00	0,03	3,87	0,35	2894,00	327,65	52,70
28388992	0,03	0,00	0,03	3,94	0,37	3088,00	349,61	53,03
9987479	0,03	0,00	0,03	4,25	0,41	3038,00	344,00	52255,00
Media	0,03	0,00	0,03	3,78	0,38	3006,40	340,39	10493,20

	Rand				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	1,71	0,00	1,71	3,60	0,26	0,00	0,26	12,39
572718921	1,71	0,00	1,71	3,63	0,26	0,00	0,26	13,72
34737829	1,71	0,00	1,71	4,36	0,26	0,00	0,26	15,26
28388992	1,71	0,00	1,71	4,27	0,26	0,00	0,26	14,83
9987479	1,71	0,00	1,71	4,03	0,26	0,00	0,26	11,46
Media	1,71	0,00	1,71	3,98	0,26	0,00	0,26	13,53

BL 20 %

	Iris				**Ecoli**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	0,027954	0	0,027954	4731	0,357933	5704	645,45	90,51
572718921	0,03	0,00	0,03	5616,00	0,37	6224,00	704,28	90,71
34737829	0,03	0,00	0,03	6,30	0,37	6076,00	687,54	91149,00
28388992	0,027954	0	0,027954	6,28292	0,37	6024,00	681,66	90,91
9987479	0,03	0,00	0,03	5,49	0,386978	6194	700,90	91,37
Media	0,03	0,00	0,03	2073,01	0,37	6044,40	683,96	18302,50

	Rand				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
1345378882	1,71	0,00	1,71	6,54	0,26	0,00	0,26	22,46
572718921	1,71	0,00	1,71	5,80	0,26	0,00	0,26	18,53
34737829	1,71	0,00	1,71	6,39	0,26	0,00	0,26	20,96
28388992	1,71	0,00	1,71	7,31	0,26	0,00	0,26	22,45
9987479	1,71	0,00	1,71	7,00	0,26	0,00	0,26	21,96
Media	1,71	0,00	1,71	6,61	0,26	0,00	0,26	21,27

Habiendo arreglado los fallos se obtienen los resultados esperados, greedy porporciona unas soluciones con infactivilidades casi nulas en un tiempo bastante pequeño, mientras que BL proporciona unos agredados generalmente inferiores, no se ha podido coregir las altísimas infactivilidades de ecoli para la BL.

§0.6: Resultados Globales

Resultados Globales 10 %

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
COPKM	0,05	1,20	0,07	0,15	2,51	120,00	16,08	7,14
BL	0,02	0	0,02	82,27	1,38	381	44,47	396,15
AGG-SF	0,03	0,00	0,03	30,82	1,42	356,00	41,69	162,49
AGG-UN	0,04	0,00	0,04	31,02	1,35	361,40	42,22	163,35
AGE-SF	0,02	0,00	0,02	82,27	1,38	381,00	44,47	396,15
AGE-UN	0,02	0,00	0,02	81,87	1,46	462,00	53,67	394,32
AM(10,1,0)	0,03	0,00	0,03	50,07	1,25	231,00	23,55	321,68
AM(10,0,1)	0,04	0,00	0,04	38,36	1,45	302,60	35,67	182,61
AM(10,0,1mej)	0,03	0,00	0,03	36,63	1,56	273,60	32,51	197,17

	Rand				Newthyroid			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
COPKM	1,43	0,00	1,43	0,13	1,36	30,40	4,33	0,47
BL	0,5	0	0,5	82,3	0,82	0,8	0,89	156,3
AGG-SF	1,05	0,00	1,05	30,73	0,90	18,80	2,74	57,86
AGG-UN	1,03	0,00	1,03	30,83	1,17	0,00	1,17	59,16
AGE-SF	0,5	0,00	0,50	82,3	0,82	0,80	0,89	156,3
AGE-UN	0,5	0,00	0,50	82,61	0,97	0,00	0,97	139,03
AM(10,1,0)	1,02	0,00	1,02	49,31	1,35	16,80	2,99	108,51
AM(10,0,1)	0,93	0,00	0,93	33,06	1,13	0,00	1,13	64,27
AM(10,0,1mej)	0,66	0,00	0,66	36,56	1,17	0,00	1,17	71,08

Resultados Globales 20 %

	Iris				Ecoli			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
COPKM	0,05	0,00	0,05	0,24	1192,16	13,80	1193,72	2399,33
BL	0,03	0,00	0,03	2073,01	0,37	6044,40	683,96	18302,50
AGG-SF	0,03	0,00	0,03	54,66	1,68	730,00	84,24	287,39
AGG-UN	0,04	0,00	0,04	57,42	1,62	409,20	47,90	289,85
AGE-SF	0,04	0,00	0,04	546,33	298,96	1799,35	502,46	5319,77
AGE-UN	0,02	0,00	0,02	142,53	1,77	372,50	43,90	704,18
AM(10,1,0)	0,04	0,00	0,04	83,66	2,15	232,40	28,37	111603,4
AM(10,0,1)	0,03	0,00	0,03	56,95	2,17	285,60	34,47	320,43
AM(10,0,1mej)	0,04	0,00	0,04	63,30	2,26	368,00	43,88	350,43

	Rand				Newthyroid			
	Tasa_C	***Tasa_inf***	***Agr,***	**T**	***Tasa_C***	***Tasa_inf***	***Agr,***	**T**
COPKM	2,10	3,20	2,46	4,65	1,69	0,00	1,69	0,72
BL	1,71	0,00	1,71	6,61	0,26	0,00	0,26	21,27
AGG-SF	0,76	0,00	0,76	52,31	1,1	0,00	1,10	104,56
AGG-UN	0,72	0,00	0,72	52,74	1,24	0,00	1,24	105,36
AGE-SF	1,32	0,80	1,41	29,08	1,07	0,00	1,07	57,98
AGE-UN	0,5	0,00	0,50	140,97	0,87	0,00	0,87	280,03
AM(10,1,0)	0,79	0,00	0,79	85,36	1,13	0,00	1,13	185,84
AM(10,0,1)	0,81	0,00	0,81	56,97	225,31	0,00	225,31	129,12
AM(10,0,1mej)	0,73	0,00	0,73	65,41	0,85	0,00	0,85	126,55

Es posible que algunos de los resutlados sean extraños esto ocurre por en en ocasiones al copiar de la terminal en el documento de excell se ignoran los puntos convirtiendo un valor de 24.677 en 24677, se han intentado corregir todos los que se han podido encontrar.

Como se puede observar los resultados globaes son similares exceptando greedy, sus tiempos son considerablemente menores y su agregado en muchas ocasiones muy alto. También hand e mencionarse los resultados de ecoli de la BL otra vez ya que porocionan soluciones nefastas por un motivo que no he conseguido entender.