UNIVERSIDAD DE GRANADA

Estudio de Metaheurística

Multi-Verse Optimizer

Alejandro Manzanares Lemus - 77393031D
alexmnzlms@correo.ugr.es

Junio 28, 2020

Índice general

Apartado 1:
Descripción del problema

El problema elegido es el **Problema del Agrupamiento con Restricciones**, a partir de ahora **PAR**, es una variante del problema de agrupamiento clásico. El problema de agrupamiento clásico consiste en que dado un conjunto X de datos con n características, hay que encontrar una partición C de tal manera que se minimice la desviación general de cada $c_i \in C$.
En la variante **PAR**, se introduce el concepto de restricción. Nosotros utilizamos las restricciones de instancia que pueden ser dos tipos:

- Restricciones **ML**(*Must-link*): Dos elementos $x_i \in X$ que posean una restricción ML, deben pertenecer al mismo $c_i \in C$.
- Restricciones **CL**(*Cannot-link*): Dos elementos $x_i \in X$ que posean una restricción CL, deben pertenecer a $c_i \in C$ distintos.

Además, estas restricciones serán débiles, es decir, el objetivo es minimizar tanto el número de restricciones incumplidas — una solución factible puede incumplir restricciones — como la desviación general de cada $c_i \in C$.

§1.1: Formalización de los datos

- Los **datos** se representan en una matriz $i \times n$ siendo i el número de datos que tenemos y n el número de características que tiene cada $x_i \in X$
  $\vec{x_i} = \{x_{i0}\ ...\ x_{in}\}$ donde cada $x_{ij} \in \mathbb{R}$
- Una **partición** C consiste en un conjunto de k clusters. $C = \{c_0\ ...\ c_k\}$. Cada c_i contiene un conjunto de elementos x_i. El número de elementos de c_i es $|c_i|$ y normalmente un cluster c_i tiene asociada una etiqueta l_i — Esto no lo utilizaremos en la implementación del problema —.
- Para cada cluster c_i se puede calcular su **centroide** $\vec{\mu_i}$ como el promedio de los elementos $x_i \in c_i$.
  $\vec{\mu_i} = \frac{1}{|c_i|} \sum_{\vec{x_j} \in c_i} \vec{x_j}$.
- La **distancia media intra-cluster** $\bar{c_i}$ se define como la media de las distancias de cada $x_i \in c_i$ a su centroide μ_i.
  $\bar{c_i} = \frac{1}{|c_i|} \sum_{\vec{x_j} \in c_i} \|\vec{x_j} - \vec{\mu_i}\|_2$
- La **desviación general** de la partición C se calcula como la media de las distancias medias intra-cluster $\bar{c_i}$.
  $\bar{C} = \frac{1}{k} \sum_{c_i \in C} \bar{c_i}$.
- El **conjunto de restricciones totales** R se calcula como la unión entre el conjunto de restricciones ML y el conjunto de restricciones CL.
  $|R|$ es el número de restricciones total $|R| = |ML| + |Cl|$.
- La **infactibilidad** — *infeasibility* — se calcula como el número de restricciones que incumple una partición C del conjunto X dado un conjunto de restricciones R. Se define $V(\vec{x_i}, \vec{x_j})$ como una función que devuelve 1 si la pareja $(\vec{x_i}, \vec{x_j})$ incumple alguna restricción.
  $infeasibility = \sum_{i=0}^{n} \sum_{j=i+1}^{n} V(\vec{x_i}, \vec{x_j})$

Apartado 2:
Descripción de los algoritmos empleados

§2.1: Función objetivo

Todos los algoritmos implementados comparten la misma función objetivo: $F_{objetivo} = \bar{C} + (infeasibility \cdot \lambda)$

El valor esta función se calcula como la suma de la **desviación general** ($\bar{C}$) y la infactibilidad de la partición (*infeasibility*) escalada por un parámetro (λ). Este parámetro tiene la función de dar relevancia al término *infeasibility*. Si se establece correctamente, primero da prioridad a reducir las restricciones incumplidas por encima de la desviación general.
Este parámetro se calcula como: $\lambda = \frac{\lceil D \rceil}{|R|}$. Siendo D la distancia entre los dos puntos más lejanos y R el número total de restricciones.
El objetivo de los algoritmos descritos a continuación es minimizar el valor de esta función objetivo.

- **K-medias Restringido Débil**: Algoritmo de heurística greedy que se basa en asignar un elemento al cluster de cuyo centroide se encuentre más cercano de manera iterativa, hasta obtener el mejor ajuste posible.
- **Búsqueda Local**: Algoritmo de búsqueda basado en trayectorias, se vale de la exploración de vecindarios para obtener la mejor solución posible.
- **Algoritmos Genéticos**: Algoritmos de búsqueda basado en poblaciones. En esta práctica implementamos un esquema generacional y uno estacionario, y los combinamos con dos operadores de cruce, uno uniforme y otro de segmento fijo, obteniendo así 4 algoritmos distintos.
- **Algoritmos Meméticos**: Algoritmos de búsqueda basado en poblaciones. Se basa en la implementación de un algoritmo genético con un esquema generacional, con la diferencia de que cada determinado número de generaciones, se aplica una búsqueda local a un determinado número de cromosomas. En esta práctica se implementan 3 versiones.
- **Búsqueda Multiarranque Básica**: Algoritmo de búsqueda basado en trayectorias. Se basa en realizar una búsqueda local partiendo de soluciones aleatorias distintas (10 BL con 10000 iteraciones) y guardar la solución cuyo valor de función objetivo es mínima. La solución que aporta la BMB es la solución guardada, es decir, la minima.
- **Enfriamiento Simulado**: Algoritmo de búsqueda basado en trayectorias. Este algoritmo es una modificación de la BL y trata de replicar el proceso de enfriamiento de un sistema físico. El proceso de búsqueda lleva asociado una temperatura. Cuando la temperatura es alta, se aceptan soluciones con peor valor de la función objetivo y según va disminuyendo la temperatura, solo se aceptan soluciones mejores.
- **Iterated Local Search**: Algoritmo de búsqueda basado en trayectorias. Similar a la Búsqueda Multiarranque Básica, con la diferencia de que a la solución de una BL se le aplica un operador de mutación y se ejecuta la siguiente BL con esta solución mutada como punto de partida.
- **Iterated Local Search - Enfriamiento Simulado**: Algoritmo de búsqueda basado en trayectorias. Mismo algoritmo que la ILS pero se aplica el enfriamiento simulado en lugar de la búsqueda local.

§2.2: Representación de los datos

Los datos comunes a ambos algoritmos se representan de la siguiente manera:

- Para los **datos** utilizo una matriz *posiciones* de números reales de dimensión $i \times n$.

```
double: matriz[i][n] posiciones
```

- Para los **centroides** de cada cluster utilizo una matriz *centroides* de dimensión $k \times n$.

```
double: matriz[k][n] centroides
```

- Para las **restricciones**, he elegido no utilizar las representación en forma de matriz, porque es muy costoso recorrerla secuencialmente, y la representación en forma de lista no te permite acceder a un elemento en concreto, por eso he decidido utilizar un map *restricciones*. Este tipo de estructura se puede recorrer secuencialmete de forma eficiente y ademas, existe un operador de búsqueda para poder acceder a un elemento concreto. El map se compone de dos elementos: la clave y el valor. La clave es la pareja de elementos x_i, x_j y el valor es 1 si la restricción es de tipo **ML** o -1 si es de tipo **CL**.

```
Pareja(int,int),int: map restricciones
```

- Los elementos x_i que pertenecen a los distintos **clusters** los almaceno en una matriz *clusters* de dimensión $k \times i$.

```
double: matriz[k][i] clusters
```

- Finalmente la **partición** C la represento en un vector de enteros *solucion*, en los que posición del vector i indica el elemento x_i y el contenido de la posición i, *solucion[i]* indica el cluster c_i al que pertenece.

```
int: vector[i] solucion
```

§2.3: Operadores comunes

Hay una serie de operadores que son comunes a los dos algoritmos, los describo a continuación:

- Operador **calcular_centroide(cluster i)** Calcula el centroide de un cluster i:

```
Para cada característica u del centroide i:
   u = 0

Para cada elemento j del cluster i:
   Para cada característica c, u del elemento j, centroide i:
      u += 1/k * c
```

- Operador **distancia_intracluster(cluster i)**: Calcula la distancia intracluster de un cluster i.

```
Para todos los clusters:
   d_intracluster = 0

Para cada elemento j del cluster i:
   Para cada característica c, u del elemento j, centroide i :
      d_intracluster += 1/k * abs(c - u) * abs(c - u)
```

- Operador **desviación_general()**: Calcula la desviación general del problema.

```
desv_gen = 0

Para cada distancia_intracluster i del cluster i:
   desv_gen += 1/k * i
```

- Operador **calcular_lambda()**: Calcula el parámetro λ

```
lambda = 0
double: d, d_max = 0
int: cluster

Para cada elemento e en posiciones:
   cluster = Cluster al que pertenece e
```

```
        Para cada número n de 0...k:
           Si n != cluster:
              Para cada elemento c de cluster n:
                 d = Distancia entre n y e
                 Si d es mayor que d_max:
                    d_max = d

    lambda = d_max / restricciones
```

- Operador **calcular_infact_sol(solucion)**: Calcula la infacibilidad de un vector solución dado

```
    int: infactibilidad = 0
    Para cada elemento i de la solucion:
       Desde j = i + 1 hasta el fin de solución:
          Si la pareja (i,j) pertence al set de restricciones:
             Si j igual -1 Y elemento i igual elemento j de sol:
                infactibilidad++
             Si j igual 1 Y elemento i distinto elemento j de sol:
                infactibilidad++

    Devolver infactibilidad
```

Apartado 3:
Estudio de una metaheurística existente

§3.1: Multi-Verse Optimizer

§3.1.1: Propuesta

Como propuesta para el estudio de una metaheurística ya existente he optado por la llamada Multi-Verse Optimizer (MVO).

Este es un algoritmo basado en poblaciones, como por ejemplo Ant Colony Optimization y Particle Swarm Optimization, que encuentra su inspiración en la teoría del multiverso. Sin entrar en detalle, la teoría del multiverso establece la existencia de diversos universos distintos al universo en el que vivimos. Estos universos interactúan entre ellos y es este concepto de interacción el que intenta imitar MVO.

En MVO existe un multiverso, formado por varios universos, siendo cada uno de estos una solución candidata a resolver el problema. Estos universos interactúan entre ellos y cambian. Hablaremos de los conceptos de agujero blanco, agujero negro y agujero de gusano — black hole, white hole y wormhole respectivamente—. A cada universo se le asocia un ratio de inflación en función del valor de su función objetivo. Este ratio está normalizado, por lo que tomará valores entre 0 y 1. Los valores dentro de cada universo se conocen como objetos.

Durante el proceso de optimización se aplican las siguientes reglas:

- Cuanto mayor sea el ratio de inflación, mayor es la probabilidad de tener un white hole.
- Cuanto mayor sea el ratio de inflación, menor es la probabilidad de tener un black hole.
- Universos con un alto ratio de inflación tienden a mandar objetos a través de white holes.
- Universos con un bajo ratio de inflación tienden a recibir objetos a través de black holes.
- Los objetos de todos los universos deben afrontar movimientos aleatorios hacia el mejor universo a través de un wormhole sin importar el ratio de inflación.

§3.1.2: Descripción del algoritmo

Podemos dividir el algoritmo en dos partes. En la primera parte los universos intercambian objetos entre ellos a través de los black y white holes, mientras que en la segunda los objetos de los universos sufren cambios y son enviados a través de los wormholes.

Comenzamos inicializando el multiverso con universos aleatorio. Después calculamos el valor de la función objetivo de los universos y su ratio de inflación normalizado de la siguiente manera:

$$ratio\_inflacion\_normalizado = \frac{f\_objetivo}{f\_objetivo\_max}$$

Ordenaremos los universos en función de su ratio de inflación normalizado. Recorreremos ahora todos los objetos de todos los universos. El objeto j del universo i será un black hole. Para seleccionar el universo cuyo objeto j se convertirá en un white hole, establecemos un método de selección de ruleta, en el que cuanto menor sea el ratio de inflación normalizado de un universo, mayor es la posibilidad de seleccionarlo. Establecemos entonces que la probabilidad de que el white hole envíe su valor hacía el black hole es el ratio de inflación normalizado del universo i (el universo en el que existe el black hole).

De esta forma los universos intercambian objetos constantemente, lo que provoca que el ratio de inflación y el valor de la función objetivo medios disminuya en el conjunto de universos.

Para que el algoritmo no se estanque y se introduzca variedad en las soluciones, permitiendo la exploración del espacio establecemos un wormhole siempre entre el mejor universo del multiverso actual y el universo i.

Calculamos los siguientes coeficientes:

- $WEP = min + l \times (\frac{max-min}{L})$
- $TDR = 1 - \frac{l^{\frac{1}{p}}}{L^{\frac{1}{p}}}$

WEP (wormhole existing probability) es la probabilidad de que el objeto j se envíe a través del wormhole. TDR (Travel Distance Rate) establece el grado de mutación de la solución, es decir, cuanto mayor sea TDR, más grande es la mutación que se le aplica al objeto j del universo i.

La mutación del objeto j se establece de la siguiente manera:

$$x_{ij} = X_j \pm TDR \times k - 1 \times random$$

donde x_{ij} es el objeto j del universo i, X_j es el objeto j del mejor universo del multiverso y k es el numero de clusters. Hay un 50 % de posibilidades de que el valor aumente o de que decremente.

Este cambio esta pensado para alterar un universo radicalmente de manera que se potencie la exploración del espacio. TDR y WEP son valores adaptativos, puesto que han resultado ser mas eficaces que valores prefijados a la hora de encontrar buenas soluciones. A medida que TDR disminuye WEP aumenta, esto implica que en las primeras iteraciones las mutaciones son más improbables pero cuando suceden son mutaciones muy grandes, mientras que para mayores valores de WEP las mutaciones son mas comunes pero no introducen cambios tan grandes.

Repetiremos este proceso hasta llegar a las 100000 iteraciones o hasta que $TDR \times k - 1 = 0$ y el ratio de inflación medio sea 1, lo que indica que el multiverso se ha estancado en una solución y ya no hay posibilidad de introducir nuevos cambios.

§3.1.3: Adaptación

A continuación se detalla la adapación de MVO al problema PAR asi como las mejoras propuestas.

Estructuras de datos

Para representar los universos, el valor de la función objetivo y los demás elementos necesarios se utiliza la siguiente estructura de datos:

```
int tam_multiverse;
std::vector<std::vector<int>> universe;
std::vector<double> f_universe;
std::vector<double> f_universe_normalized;
std::vector<std::pair<double,int>> sorted_universe;
```

He utilizado vectores ya que es muy comodo trabajar con ellos. El vector sorted_universe es una pareja de ratio de inflación normalizado de un universo y el universo al que pertenece, de esta manera la ordenación del vector es más rápida y se puede acceder a cada universo sabiendo su posición.

Parámetros utilizados

- tam_multiverse: Es el número de universos presentes en el multiverso, es decir, el tamaño de la población. En nuestro caso utilizaremos una población de tamaño 30.

- l y L: l es la iteración actual y L es el número máximo de iteraciones. Para la resolución del problema estableceremos un numero de iteraciones de 100000. Aunque el número máximo de iteraciones sea 100000, nunca las alcanzaremos debido a la condición de parada del algoritmo.
- min y max: Son los valores minimo y máximo respectivamente de WEP. min se establece a 0,2 y max a 1. Esto significa que al principio la probabilidad de intercambio de valores a través de un wormhole es del 20 % y llega al 100 % con el paso de la iteraciones.
- p: Este parametro afecta al calculo de TDR. Por las pruebas realizadas, es importante notar que TDR es un valor que influye significativamente en el comportamiento de MVO. Cuanto mayor es el valor de p, mas rapido disminuye el valor de TDR. p se establece a k, siendo que para *Iris*, *Rand* y *Newthyroid* es 3 y para *Ecoli* es 8. De esta manera a los cojuntos más pequeños les da tiempo a explorar el espacio antes de converger a una solución.

§3.1.4: Mejoras y modificaciones

Se han propuesto dos mejoras para mejorar el desempeño de MVO:

La primera de ellas ha sido no perder la mejor solución encontrada hasta el momento. Después de probar por primera vez MVO, me di cuenta de que si se encontraba una solución muy buena, existía una posibilidad de empeorarla si otras soluciones intercambiaban objetos con ella. Teóricamente esto es bueno, ya que el objetivo es conseguir que la solución no se estanque en un optimo local y explorar lo máximo posible el espacio de soluciones, pero en la practica, al ser las soluciones tan parecidas entre ellas, las posibilidades de empeorar la mejor solución encontrada y de no converger eran bastante altas. De hecho, hasta que el valor TDR no alcanzaba un valor bastante reducido, los universos exploraban el espacio de soluciones de forma caótica — si bien es cierto que poco a poco la inflación media disminuía en el multiverso—.
Por esto decidi implementar la siguiente mejora: en cada iteración se guarda el mejor universo encontrado hasta el momento. Una vez ha terminado el ciclo de intercambio y mutación entre los universos, se comprueba si la mejor solución del nuevo multiverso es mejor que la mejor solución almacenada. Si no es asi, la mejor solución almacenada se reintroduce en el multiverso siendo sustituida por la peor solución del cojunto. Con esta mejora conseguimos evitar que las buenas soluciones encontradas se pierdan permitiendo aún asi que se encuentren soluciones mejores.

Las segunda mejora implementada ha sido realizar una búsqueda local al 10 % de las soluciones. En nuestro caso a las 3 mejores soluciones del multiverso. La búsqueda local es un híbrido entre la implementada en las prácticas y una búsqueda local suave. Como máximo tiene 1000 iteraciones y 500 fallos — siendo un fallo cuando al generar un vecino este no mejore a la solución actual —.
Esta búsqueda local se realiza cada 10 iteraciones del algoritmo. Con esto conseguimos mejorar al máximo las buenas soluciones encontradas, aumentando asi sus posibilidades de que compartan información con otros universos y que mejore la inflación media del conjunto.

Por último se ha probado una versión en la que se implementan ambas mejoras que es de hecho, la que mejores soluciones encuentra.

§3.1.5: Pseudocódigo del algoritmo

```
int: tam_multiverse = 30
int: max_iter = 100000
int: iter = 0
double: min = 0.2
double: max = 1.0
double: p = n_cluster
double: wep, tdr = 0
int: best_universe
double: comp
bool: exit = false
int: black_hole_index, white_hole_index

iniciar_universe()
evaluate_fitness()
sort_universes()

Mientras iter < max_iter Y no exit:

   Para i = 0 hasta tam_multiverse:
      calcular wep y tdr
      black_hole_index = i

      Para j = 0 hasta tamaño de universo i:
         r1 = numero aleatorio entre 0 y 1

         Si r1 < ratio_infacion_normalizado_i:
            white_hole_index = roulette_wheel_selection()
            universe[black_hole_index][j] = universe[sorted_universe[white_hole_index].second][j]

         r2 = numero aleatorio entre 0 y 1
         Si r2 < wep:
            r3 = numero aleatorio entre 0 y 1
            r4 = numero aleatorio entre 0 y 1
            best_universe = sorted_universe[0].second

            Si r3 < 0.5:
               universe[i][j] = universe[best_universe][j] + (tdr * r4 * (n_cluster-1))

            Si no:
               universe[i][j] = universe[best_universe][j] - (tdr * r4 * (n_cluster-1))

            reparar_solucion(universe[i])

   evaluate_fitness()
   sort_universes()

   comp = 0
   exit = false

   Para k = 0 hasta tam_multiverse:
      comp += sorted_universe[k].first

   Si (comp / tam_multiverse) == 1.0 Y (tdr * (n_cluster-1)) == 0:
      exit = true

   iter++
}
solucion = universe[sorted_universe[0].second]
leer_solucion()
```

§3.1.6: Operadores propios

- sort_universes(): Ordena los universos en el vector sorted_universe.

```
double: max = 0.0;

Para i = 0 hasta tam_multiverse:
   Si max < f_universe[i]:
      max = f_universe[i]

Para i = 0 hasta tam_multiverse:
   Añadir pareja(f_universe[i]/max , i) a sorted_universe
   Añadir f_universe[i]/max a f_universe_normalized[i]

Ordenar sorted_universe
```

- evaluate_fitness(): Calcula el valor de la función objetivo de los universos.

```
Recorrer el vector universe:
   Añadir evaluar_solucion(universe) a f_universe
```

- roulette_wheel_selection(): Metodo de selección por ruleta.

```
double: suma = 0;

Para i = 0 hasta tam_multiverse:
   suma += (1.0 - sorted_universe[i].first);

random = numero aleatorio entre 0 y suma
float: acumulado = 0.0
int: index = 0

Mientras random >= acumulado:
   acumulado += (1.0 - sorted_universe[index].first)
   index++

Devolver index - 1
```

- local_search_mvo(sol): Búsqueda local que se aplica a MVO.

```
int: iter = 0, iter_max = 1000;
int: fallos = 0;
double: f_ant;
int: vector sol_ant;
int infactibilidad_ant;

solucion = sol;

Mientras iter < iter_max && fallos < 500:
   sol_ant = solucion
   f_ant = f_objetivo
   infactibilidad_ant = infactibilidad
   salir = false

   generar_vecino()

   Si f_objetivo > f_ant:
      f_objetivo = f_ant
      solucion = sol_ant
      infactibilidad = infactibilidad_antt
      fallos++

   iter++;
}

sol = solucion
```

- Mejora 1: Reincorporar la mejor solución.

```
f_mejor_sol = evaluar_solucion(mejor_sol)
Si f_mejor_sol < evaluar_solucion(universe[sorted_universe[0].second]):
   universe[sorted_universe[tam_multiverse-1].second] = mejor_sol
```

```
        f_universe[sorted_universe[tam_multiverse-1].second] = f_mejor_sol
        sort_universes()
```

- Mejora 2: Aplicar búsqueda local.

```
    Si iter > 0 Y iter mod 10 == 0:
      Para i = 0 hasta 0.1*tam_multiverse:
          local_search_mvo(universe[sorted_universe[i].second])
          f_universe[sorted_universe[i].second] = evaluar_solucion(universe[sorted_universe[i].second])
          sort_universes()
```

Apartado 4:
Procedimiento

§4.1: Estructura de datos

La implementación de la práctica se ha llevado a cabo en c++.
Para la estructura de datos he optado por una sola clase, llamada CCP — Constrained Clustering Problem — en la que están todos los datos necesarios para realizar el problema:

```
int n_cluster;
std::vector<std::vector<double>> posiciones;
std::vector<std::vector<double>> centroides;
std::map<std::pair<int,int>,int> restricciones;
std::vector<double> d_intracluster;
std::vector<int> solucion;

std::set<std::pair<int,int>> vecindario;
std::vector<std::vector<int>> clusters;

int poblacion;
int ind_eval;
std::vector<std::vector<int>> generacion;
std::vector<std::vector<int>> seleccion;
std::vector<double> f_generacion;
std::vector<double> f_seleccion;
std::vector<int> mejor_generacion;
double f_mejor_generacion;

double desv_gen;
double infactibilidad;
double lambda;
double f_objetivo;
```

He utilizado las clases map, set y vector de la STL.
Las restricciones se almacenan en un map debido a que al ser una estructura de datos de la STL, es posible recorrerlo de forma secuencial con un iterador y ademas, cuenta con el operador find, que permite saber si existe determinada combinación de elementos x_i y si la restricción es de tipo ML o CL. Por tanto me pareció mejor implementación que la propuesta de matriz y lista.
El vecindario se utiliza como una manera para poder saber cuando ha terminado el algoritmo de búsqueda local y no volver a explorar vecinos que ya he explorado previamente. Utilizo un set porque a diferencia del vector, no permite que existan parejas (elemento, cluster) duplicadas y ademas estas se ordenan automáticamente en orden ascendente.
Las matrices de generación y selección se implementan como vectores de vectores de enteros y las funciones objetivo asociadas a los cromosomas se almacenan en vectores de doubles.
Los operadores de los algoritmos descritos anteriormente se implementan como métodos de la clase CCP.

§4.2: Guía de Uso

El programa se compila utilizando la orden **make**.
Para ejecutar el programa es necesario ejecutar el archivo:
BIN/clustering_exe [*semilla*] [*conjunto*] [*algoritmo*] [*v*].
Estos parámetros se configuran de la siguiente manera:

- [*semilla*]: Es el número de semilla que se quiere ejecutar.
  - $[1-5]$: Ejecutan desde la primera semilla hasta la que se pase como argumento
  - [0]: Si se introduce el 0, el programa entra en modo gráfico. Esto significa que genera un output de datos para poder pintar una gráfica de la evolución del valor de la función objetivo con el paso de las generaciones —Solo funciona para algoritmos genéticos y meméticos, búsqueda local y enfriamiento simulado—. Si se utiliza para enfriamiento simulado debe proporcinarse un numero [0-2] para realizar los distintos gráficos.
- [*conjunto*]: Numero del conjunto que se quiere ejecutar.
  - [*Rand*]: 1 para 10 % y 5 para 20 %
  - [$Iris$]: 2 para 10 % y 6 para 20 %
  - [$Ecoli$]: 3 para 10 % y 7 para 20 %
  - [$Newthyroid$]: 4 para 10 % y 8 para 20 %
  - [0]: Ejecuta todos los conjuntos anteriores
- [*algoritmo*]: Algoritmo que se quiere utilizar:
  - [G]: K-medias Restringido Débil.
  - [BL]: Búsqueda local.
  - [AGG_UN]: Algoritmo genético con operador de cruce uniforme y esquema generacional.
  - [AGG_SF]: Algoritmo genético con operador de cruce segmento fijo y esquema generacional.
  - [AGE_UN]: Algoritmo genético con operador de cruce uniforme y esquema estacionario.
  - [AGE_SF]: Algoritmo genético con operador de cruce segmento fijo y esquema estacionario.
  - $[AM\_10-1{,}0]$: Algoritmo memético que aplica la búsqueda local cada 10 generaciones a todos los cromosomas
  - $[AM\_10-0{,}1]$: Algoritmo memético que aplica la búsqueda local cada 10 generaciones al 10 % de los cromosomas
  - $[AM\_10-0{,}1mej]$: Algoritmo memético que aplica la búsqueda local cada 10 generaciones al 10 % de los mejores cromosomas
  - [BMB]: Búsqueda Multiarranque Básica.
  - [ES]: Enfriamiento Simulado.
  - [ILS]: Iterated Local Search.
  - [$ILS-ES$]: Iterated Local Search con enfriamiento simulado.
  - [MVO]: Multi-Verse Optimizer.
  - [SEM]: Activa el modo búsqueda de semillas.
- [*v*](Verbose): Muestra por pantalla la traza de ejecución del algoritmo.
- $[0-3]$: Ejecutan las distintas versiones de MVO.

Como ejemplo básico, para obtener los resultados que se muestran previamente, se debe ejecutar:
BIN/clustering_exe 5 0 [*algoritmo*]

La estructura de ficheros es la siguiente:

- cc.h: Cabecera de la clase CCP.
- cc.p: Implementación de los métodos de la clase CCP.
- main.cpp: Implentación de la ejecución de los algoritmos greedy y BL.
- random.h y random.cpp : Cabeceras e implementación del generador de aleatorios.

Apartado 5:
Experimentos y análisis de resultados

§5.1: Semillas

Para la ejecución de los algoritmos se han seleccionado las siguientes semillas, utilizando un algoritmo para probar que no producen ciclos en ninguna de las ejecuciones del algoritmo greedy para ninguno de los conjuntos de datos. Toda semilla que en menos de 1000 iteraciones del algoritmo greedy no obtenga resultado es rechazada. Las semillas se han cambiado respecto a la práctica anterior debido al cambio de conjuntos de datos:

- 2024614690
- 2024676296
- 2024677261
- 2024740484
- 2024740899

El código utilizado para encontrar semillas se encuentra en la función **buscar_semilla()** que se incluye en el fichero main.cpp

§5.2: Análisis

A continuación realizaré un análisis de los resultados aportados por MVO para el problema PAR. Analizaré los conjuntos de *Iris* y *Rand* de manera separada a *Ecoli* y *Newthyroid* ya que considero que la primera pareja de conjuntos poseen menor complejidad para encontrar soluciones optimas al problema, mientras que la segunda pareja — sobre todo *Ecoli* — presentan una mayor complejidad para obtener soluciones de calidad.

§5.2.1: Iris & Rand

Podemos ver que normalmente MVO es capaz de converger a la solución optima encontrada para *Iris* y *Rand* salvo ciertas iteraciones. Esto se debe a que en MVO, es posible una vez alcanzada una buena solución, que esta se pierda al recibir información de otras soluciones, porque como podemos ver en la figura 5.1 y 5.2, aunque MVO alcanza la solución optima, esta se pierde al recibir objetos de otros universos con peor ratio de inflación.

Justo por este motivo se diseñó la mejora basada en reintroducir la mejor solución hasta el momento. Podemos ver que la mejora funciona ya que la convergencia ha mejorado en la mayoría de iteraciones.

Si nos fijamos en la versión mejorada que aplica una búsqueda local (MVO-BL), vemos que la aplicación de una búsqueda local es altamente efectiva, ya que conseguimos que todas las iteraciones converjan en la solución optima.

Como es lógico, la versión que combina ambas mejoras, también converge debido a la búsqueda local.

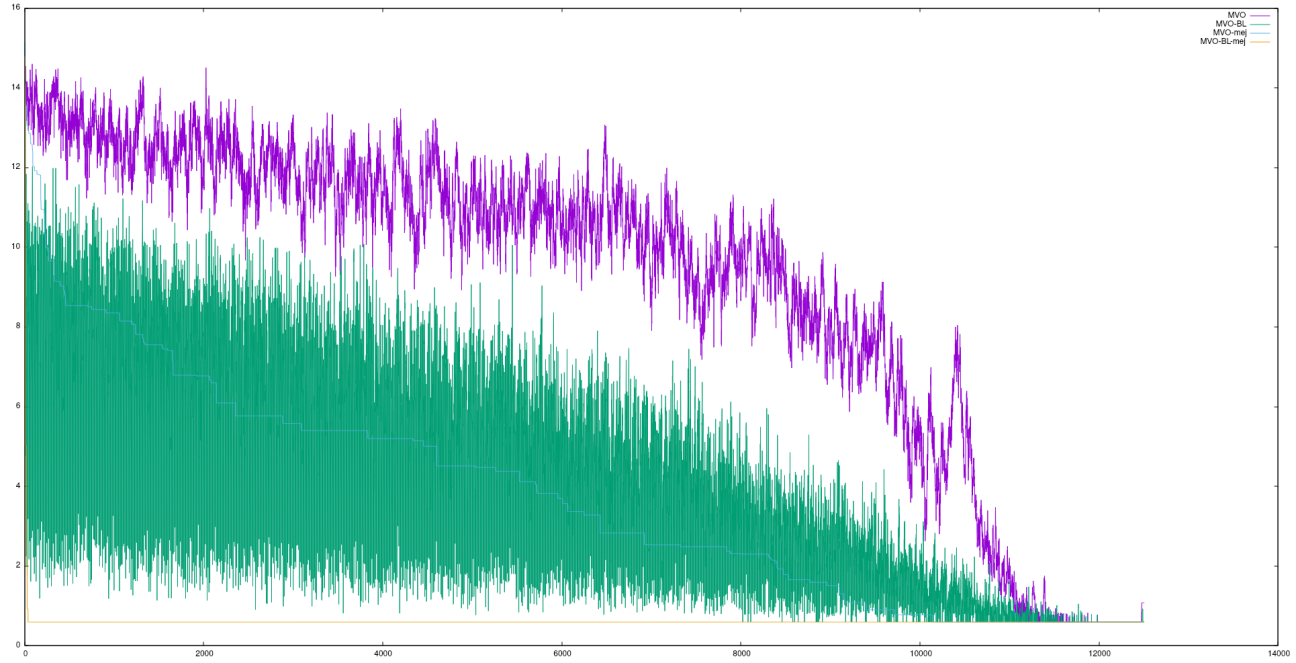

Figura 5.1: Iris con 10 % de Restricciones

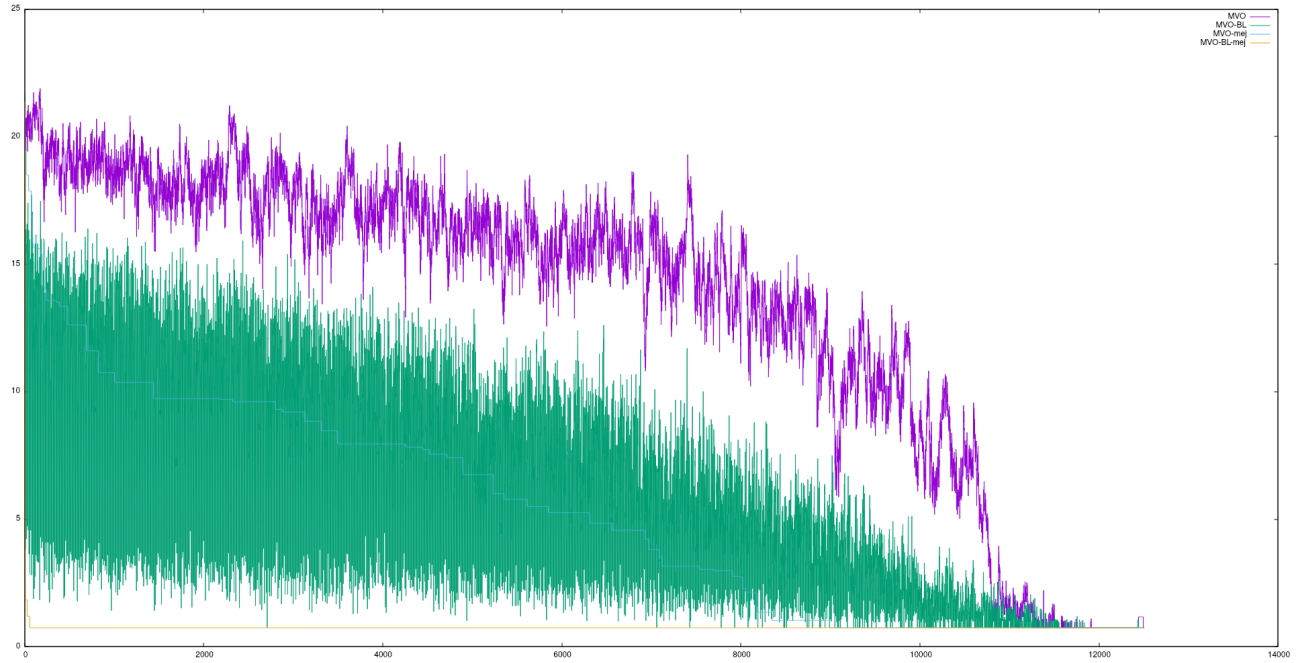

Figura 5.2: Rand con 10 % de Restricciones

§5.2.2: Ecoli & Newthyroid

A diferencia de *Iris* y *Rand*, *Ecoli* y *Newthyroid* si que presentan más dificultad a la hora de encontrar una solución optima.

Como se puede apreciar, sobre todo en la figura 5.4, la versión basica de MVO tiene el problema de perder la mejor solución. Por esto, las soluciones que aporta para *Ecoli* son nefastas, ya que el algoritmo no explota las soluciones lo suficiente para que converjan a una buena solución, mientras que las soluciones encontradas para *Newthyroid* son malas pero no tan nefastas como las de *Ecoli* porque como vemos en la figura 5.4, las soluciones si pueden converger a buenos valores.

La versión que reintroduce la mejor solución, soluciona el problema de perder la mejor solución encontrada, mejora las soluciones encontradas, pero no es capaz de solventar el problema de la falta de explotación sobre todo en *Ecoli*.

La versión con búsqueda local podemos ver que es igual de efectiva que en los conjuntos *Iris* y *Rand*, consiguiendo que todas las soluciones para *Newthyroid* converjan al optimo conocido y mejorando enormemente las soluciones que aporta MVO para *Ecoli* ya que estas soluciones tienen bastante más calidad que las aportadas por otros algoritmos hechos en las prácticas.

Como es lógico, la versión que combina ambas mejoras, es capaz de aportar las mejores soluciones para *Ecoli* — ya que para *Newthyroid* converge al óptimo—.

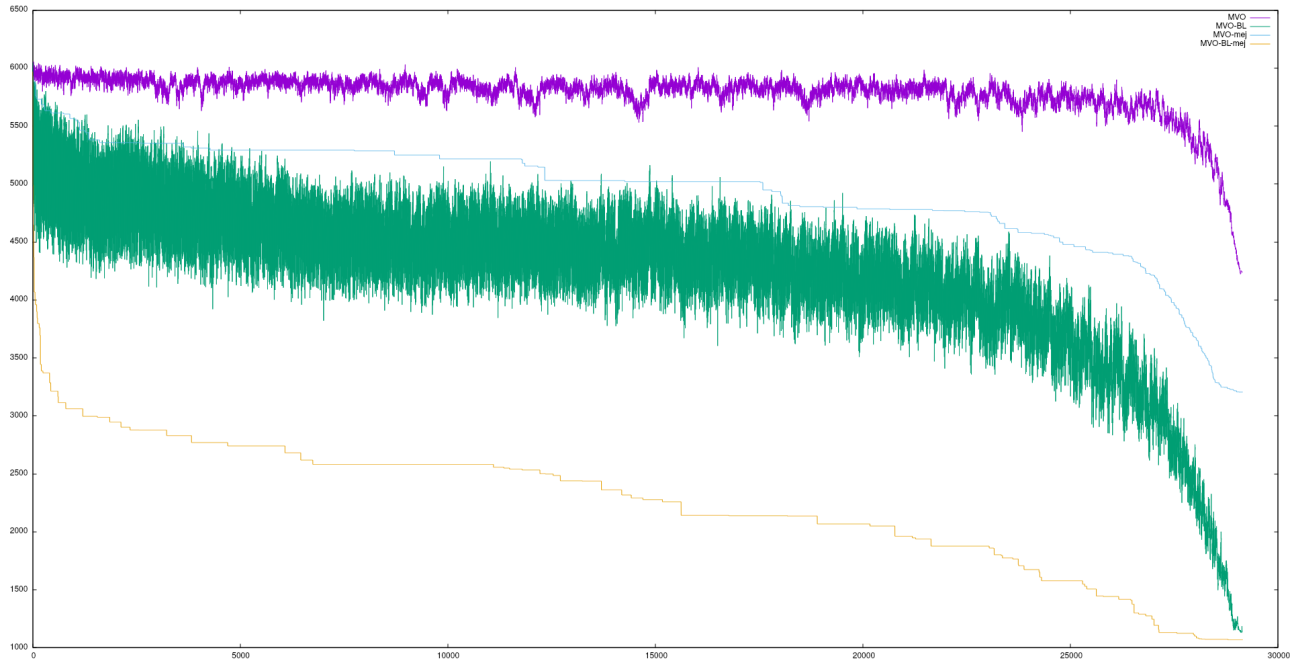

Figura 5.3: Ecoli con 10 % de Restricciones

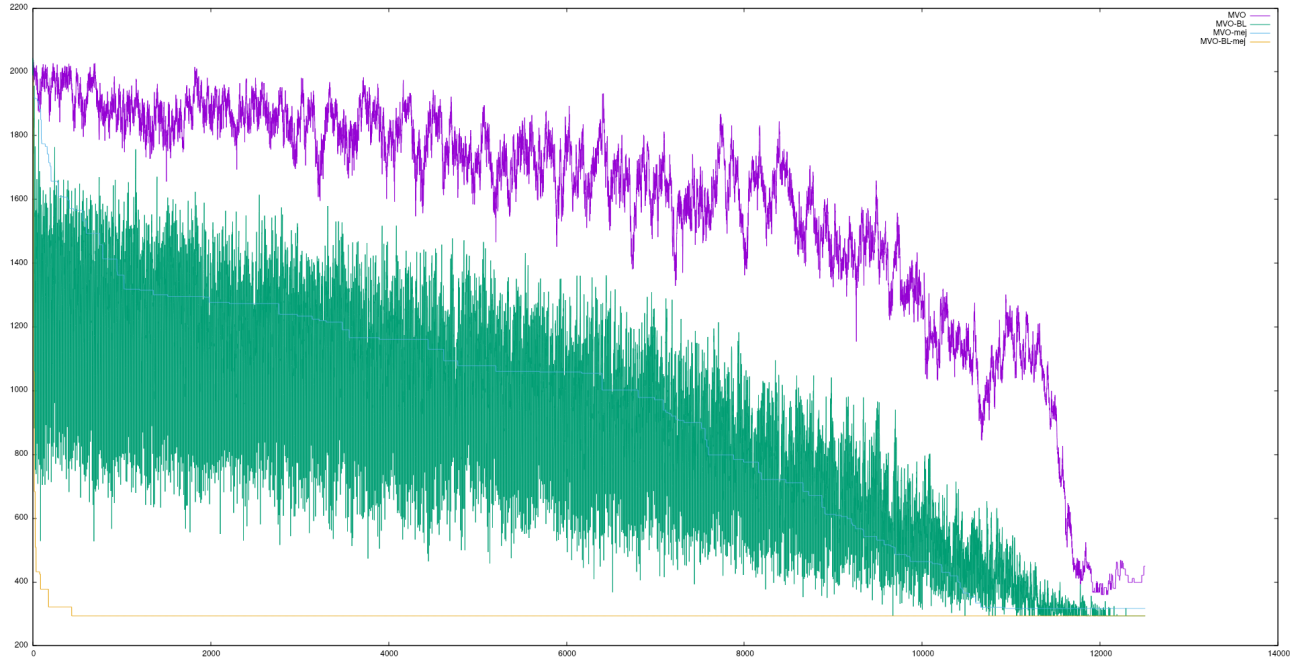

Figura 5.4: Newthyroid con 10 % de Restricciones

§5.2.3: Conclusión

Finalmente como conclusión, he de decir que el rendimiento de MVO ha sido más bajo del esperado, siendo necesarias varias mejoras para que este algoritmo aporte soluciones que puedan competir con las aportadas por las metaheurísticas con las que hemos trabajado durante las prácticas.

El mayor problema de MVO a mi parecer, es la posibilidad de perder una buena solución y que la explotación de las soluciones no es la suficiente para que las soluciones puedan converger a soluciones que sean de calidad.

§5.3: Tablas

§5.3.1: Valores medios

Resultados globales en el PAR con 10 % de restricciones

	Iris				Rand			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
K-medias Restringido Débil	0,62	15	0,9	0,01	0,91	16	1,37	0,00
Búsqueda Local	0,60	0	0,6	0,70	0,75	0	0,75	0,61
MVO	0,64	7	0,8	134,36	0,92	14	1,32	135,00
MVO-mej	0,60	0	0,6	140,86	0,79	4	0,91	141,60
MVO-BL	0,60	0	0,6	172,54	0,75	0	0,75	172,62
MVO-BL-mej	0,60	0	0,6	177,68	0,75	0	0,75	177,65
AGG_UN	0,60	0	0,60	29,11	0,75	0	0,75	29,13
AGG_SF	0,60	0	0,60	29,24	0,75	0	0,75	29,27
AGE_UN	0,60	0	0,60	29,67	0,75	0	0,75	29,62
AGE_SF	0,60	0	0,60	29,12	0,75	0	0,75	29,12
AM(10,1.0)	0,60	0	0,60	27,82	0,75	0	0,75	27,16
AM(10,0.1)	0,60	0	0,60	28,04	0,75	0	0,75	27,39
AM(10,0.1mej)	0,60	0	0,60	28,05	0,75	0	0,75	27,45
BMB	0,60	0	0,6	0,32	0,75	0	0,75	0,32
ES	0,60	0	0,7	0,35	0,75	0	0,75	0,34
ILS	0,60	0	0,6	0,15	0,75	0	0,75	0,16
ILS-ES	0,60	0	0,6	1,54	0,75	0	0,75	1,46

Resultados globales en el PAR con 10 % de restricciones

	Ecoli				Newthyroid			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
K-medias Restringido Débil	1.669,95	179	2.033,34	0,34	334,76	88	423,16	0,01
Búsqueda Local	1.791,66	329	2.457,27	38,34	430,97	43	497,11	2,41
MVO	1.822,71	1.143	4.137,55	1.696,41	274,65	88	363,05	273,02
MVO-mej	1.493,83	938	3.393,83	1.805,98	262,07	63	325,07	286,65
MVO-BL	920,65	93	1.109,03	1.987,94	270,80	15	285,80	333,16
MVO-BL-mej	926,90	66	1.061,40	2.030,80	270,80	15	285,80	343,83
AGG_UN	982,47	111	1.206,91	151,71	273,11	31	320,18	60,06
AGG_SF	977,12	149	1.278,12	152,94	276,95	29	321,87	60,18
AGE_UN	952,53	128	1.212,21	156,84	269,58	29	314,80	60,79
AGE_SF	947,54	124	1.198,31	155,18	267,36	45	337,18	60,11
AM(10,1.0)	908,77	69	1.048,54	157,38	270,80	15	293,88	59,62
AM(10,0.1)	885,09	76	1.039,84	155,78	270,80	15	293,88	58,47
AM(10,0.1mej)	887,64	73	1.034,70	152,41	272,32	64	370,45	58,53
BMB	962,44	141	1.248,86	11,52	270,80	15	285,80	0,80
ES	919,79	92	1.106,14	4,36	270,80	15	285,80	1,28
ILS	928,67	75	1.081,40	4,61	270,80	15	285,80	0,37
ILS-ES	2.308,24	1.734	5.821,02	4,66	364,11	145	509,11	2,26

Resultados globales en el PAR con 20 % de restricciones

	Iris				Rand			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
K-medias Restringido Débil	0,60	15	0,8	0,01	0,77	11	0,93	0,01
Búsqueda Local	0,60	0	0,6	0,67	0,75	0	0,75	0,64
MVO	0,64	14	0,8	178,63	0,76	4	0,81	177,79
MVO-mej	0,62	2	0,6	187,76	0,91	26	1,28	186,92
MVO-BL	0,60	0	0,6	225,81	0,75	0	0,75	223,09
MVO-BL-mej	0,60	0	0,6	232,34	0,75	0	0,75	230,42
AGG_UN	0,60	0	0,60	38,79	0,75	0	0,75	38,87
AGG_SF	0,60	0	0,60	39,05	0,75	0	0,75	39,12
AGE_UN	0,60	0	0,60	38,81	0,75	0	0,75	38,81
AGE_SF	0,60	0	0,60	39,17	0,75	0	0,75	39,13
AM(10,1.0)	0,60	0	0,60	37,04	0,75	0	0,75	37,10
AM(10,0.1)	0,60	0	0,60	37,11	0,75	0	0,75	37,15
AM(10,0.1mej)	0,60	0	0,60	37,10	0,75	0	0,75	37,18
BMB	0,60	0	0,6	0,34	0,75	0	0,75	0,33
ES	0,60	0	0,6	0,42	0,75	0	0,75	0,42
ILS	0,60	0	0,6	0,20	0,75	0	0,75	0,19
ILS-ES	0,60	0	0,6	1,76	0,75	0	0,75	1,75

Resultados globales en el PAR con 20 % de restricciones

	Ecoli				Newthyroid			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
K-medias Restringido Débil	1.642,50	146	1.790,36	0,23	318,22	103	397,72	0,01
Búsqueda Local	1.673,46	340	2.018,01	48,48	311,87	20	327,56	3,19
MVO	1.804,98	2.330	4.165,19	2.251,30	270,81	142	380,15	363,69
MVO-mej	1.592,50	1.988	3.605,52	2.382,98	307,39	120	399,96	380,18
MVO-BL	961,92	177	1.141,39	2.536,94	270,80	41	302,33	413,98
MVO-BL-mej	939,16	133	1.073,66	2.603,10	270,80	41	302,33	428,14
AGG_UN	948,18	165	1.115,70	204,08	269,39	62	316,75	80,76
AGG_SF	946,03	175	1.123,67	204,81	270,80	41	302,33	81,46
AGE_UN	942,56	189	1.134,18	207,88	268,46	77	327,51	80,52
AGE_SF	950,07	148	1.099,96	206,69	270,80	41	302,33	81,01
AM(10,1.0)	917,43	123	1.041,60	223,52	270,80	41	302,33	78,11
AM(10,0.1)	927,62	98	1.026,67	201,03	270,80	41	302,33	77,53
AM(10,0.1mej)	939,57	104	1.044,90	203,37	270,80	41	302,33	77,54
BMB	977,56	219	1.199,36	12,42	270,80	41	302,33	0,76
ES	946,10	164	1.112,00	5,45	270,80	41	302,33	1,65
ILS	946,08	109	1.056,68	5,21	270,80	41	302,33	0,49
ILS-ES	2.303,54	3.467	5.814,47	5,63	292,10	186	434,96	2,82

§5.3.2: MVO

Restricciones del 10 %

Resultados obtenidos por el MVO en el PAR con 10 % de restricciones

	Iris				Rand			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0,67	18	1,1	134,73	0,75	0	0,75	135,01
Ejecución 2	0,66	8	0,8	134,68	0,75	0	0,75	135,15
Ejecución 3	0,67	7	0,8	134,47	0,86	11	1,18	135,17
Ejecución 4	0,60	0	0,6	133,99	1,31	44	2,59	135,12
Ejecución 5	0,60	0	0,6	133,94	0,92	14	1,33	134,57
Media	0,64	7	0,8	134,36	0,92	14	1,32	135,00

Resultados obtenidos por el MVO en el PAR con 10 % de restricciones

	Ecoli				Newthyroid			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	1.876,59	1.167	4.240,45	1.716,69	280,39	110	390,39	272,97
Ejecución 2	1.929,49	1.096	4.149,54	1.717,24	277,31	165	442,31	272,76
Ejecución 3	1.853,82	1.208	4.300,73	1.731,82	271,84	82	353,84	273,83
Ejecución 4	1.718,67	1.229	4.208,12	1.659,41	271,34	59	330,34	273,11
Ejecución 5	1.734,97	1.014	3.788,92	1.656,91	272,38	26	298,38	272,42
Media	1.822,71	1.143	4.137,55	1.696,41	274,65	88	363,05	273,02

Restricciones del 20 %

Resultados obtenidos por el MVO en el PAR con 20 % de restricciones

	Iris				Rand			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0,60	0	0,6	178,72	0,79	20	1,08	177,76
Ejecución 2	0,78	53	1,4	178,61	0,75	0	0,75	177,78
Ejecución 3	0,60	0	0,6	178,59	0,75	0	0,75	177,81
Ejecución 4	0,60	0	0,6	178,60	0,75	0	0,75	177,84
Ejecución 5	0,62	19	0,8	178,63	0,75	0	0,75	177,78
Media	0,64	14	0,8	178,63	0,76	4	0,81	177,79

Resultados obtenidos por el MVO en el PAR con 20 % de restricciones

	Ecoli				Newthyroid			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	1.732,50	2.612	4.377,91	2.247,43	274,18	210	435,64	364,36
Ejecución 2	1.813,65	2.315	4.158,26	2.249,56	269,35	141	377,76	363,47
Ejecución 3	1.932,11	2.160	4.119,74	2.251,24	270,80	41	302,33	363,49
Ejecución 4	1.801,04	2.169	3.997,78	2.251,73	271,66	140	379,30	363,55
Ejecución 5	1.745,61	2.396	4.172,25	2.256,56	268,08	179	405,71	363,58
Media	1.804,98	2.330	4.165,19	2.251,30	270,81	142	380,15	363,69

§5.3.3: MVO-mej

Restricciones del 10 %

Resultados obtenidos por el MVO-mej en el PAR con 10 % de restricciones

	Iris				Rand			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0,60	0	0,6	140,96	0,75	0	0,75	141,48
Ejecución 2	0,60	0	0,6	141,32	0,75	0	0,75	141,58
Ejecución 3	0,60	0	0,6	140,98	0,75	0	0,75	141,71
Ejecución 4	0,60	0	0,6	140,79	0,75	0	0,75	141,71
Ejecución 5	0,60	0	0,6	140,24	0,97	20	1,55	141,52
Media	0,60	0	0,6	140,86	0,79	4	0,91	141,60

Resultados obtenidos por el MVO-mej en el PAR con 10 % de restricciones

	Ecoli				Newthyroid			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	1.477,52	853	3.205,35	1.817,29	272,89	29	301,89	286,94
Ejecución 2	1.506,18	986	3.503,41	1.816,06	259,06	38	297,06	286,55
Ejecución 3	1.424,85	1.026	3.503,11	1.820,29	248,17	97	345,17	286,77
Ejecución 4	1.485,08	848	3.202,78	1.777,89	260,54	122	382,54	286,70
Ejecución 5	1.575,51	977	3.554,51	1.798,37	269,69	29	298,69	286,31
Media	1.493,83	938	3.393,83	1.805,98	262,07	63	325,07	286,65

Restricciones del 20 %

Resultados obtenidos por el MVO-mej en el PAR con 20 % de restricciones

	Iris				Rand			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0,60	0	0,6	188,12	0,75	0	0,75	186,52
Ejecución 2	0,60	0	0,6	187,23	0,97	18	1,23	186,98
Ejecución 3	0,60	0	0,6	187,81	0,75	0	0,75	186,86
Ejecución 4	0,60	0	0,6	187,85	0,75	0	0,75	186,56
Ejecución 5	0,73	12	0,9	187,79	1,33	110	2,93	187,68
Media	0,62	2	0,6	187,76	0,91	26	1,28	186,92

Resultados obtenidos por el MVO-mej en el PAR con 20 % de restricciones

	Ecoli				Newthyroid			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	1.638,14	2.035	3.699,17	2.374,45	270,74	101	348,40	385,61
Ejecución 2	1.643,63	1.690	3.355,25	2.382,06	464,35	171	595,82	378,34
Ejecución 3	1.645,06	1.920	3.589,62	2.378,52	271,48	97	346,06	384,23
Ejecución 4	1.484,82	1.888	3.396,97	2.392,62	259,55	192	407,18	376,98
Ejecución 5	1.550,83	2.405	3.986,59	2.387,24	270,80	41	302,33	375,75
Media	1.592,50	1.988	3.605,52	2.382,98	307,39	120	399,96	380,18

§5.3.4: MVO-BL

Restricciones del 10 %

Resultados obtenidos por el MVO-BL en el PAR con 10 % de restricciones

	Iris				**Rand**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0,60	0	0,6	172,81	0,75	0	0,75	172,36
Ejecución 2	0,60	0	0,6	172,98	0,75	0	0,75	173,26
Ejecución 3	0,60	0	0,6	173,17	0,75	0	0,75	173,07
Ejecución 4	0,60	0	0,6	171,97	0,75	0	0,75	171,70
Ejecución 5	0,60	0	0,6	171,75	0,75	0	0,75	172,70
Media	0,60	0	0,6	172,54	0,75	0	0,75	172,62

Resultados obtenidos por el MVO-BL en el PAR con 10 % de restricciones

	Ecoli				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	929,45	105	1.142,13	2.003,73	270,80	15	285,80	333,73
Ejecución 2	936,89	76	1.090,83	2.003,30	270,80	15	285,80	332,94
Ejecución 3	945,63	76	1.099,57	2.008,64	270,80	15	285,80	332,79
Ejecución 4	866,25	116	1.101,22	1.958,50	270,80	15	285,80	332,97
Ejecución 5	925,02	92	1.111,38	1.965,55	270,80	15	285,80	333,35
Media	920,65	93	1.109,03	1.987,94	270,80	15	285,80	333,16

Restricciones del 20 %

Resultados obtenidos por el MVO-BL en el PAR con 20 % de restricciones

	Iris				**Rand**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0,60	0	0,6	225,87	0,75	0	0,75	222,93
Ejecución 2	0,60	0	0,6	225,59	0,75	0	0,75	223,05
Ejecución 3	0,60	0	0,6	225,53	0,75	0	0,75	223,01
Ejecución 4	0,60	0	0,6	226,26	0,75	0	0,75	223,34
Ejecución 5	0,60	0	0,6	225,79	0,75	0	0,75	223,12
Media	0,60	0	0,6	225,81	0,75	0	0,75	223,09

Resultados obtenidos por el MVO-BL en el PAR con 20 % de restricciones

	Ecoli				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	902,87	215	1.120,62	2.606,02	270,80	41	302,33	413,65
Ejecución 2	989,73	147	1.138,61	2.602,72	270,80	41	302,33	413,31
Ejecución 3	920,30	180	1.102,60	2.520,41	270,80	41	302,33	413,65
Ejecución 4	930,53	159	1.091,56	2.516,54	270,80	41	302,33	413,42
Ejecución 5	1.066,18	185	1.253,55	2.438,99	270,80	41	302,33	415,89
Media	961,92	177	1.141,39	2.536,94	270,80	41	302,33	413,98

§5.3.5: MVO-BL-mej

Restricciones del 10 %

Resultados obtenidos por el MVO-BL-mej en el PAR con 10 % de restricciones

	Iris				**Rand**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0,60	0	0,6	178,03	0,75	0	0,75	178,02
Ejecución 2	0,60	0	0,6	178,28	0,75	0	0,75	178,31
Ejecución 3	0,60	0	0,6	176,96	0,75	0	0,75	178,20
Ejecución 4	0,60	0	0,6	177,63	0,75	0	0,75	176,84
Ejecución 5	0,60	0	0,6	177,50	0,75	0	0,75	176,88
Media	0,60	0	0,6	177,68	0,75	0	0,75	177,65

Resultados obtenidos por el MVO-BL-mej en el PAR con 10 % de restricciones

	Ecoli				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	918,84	74	1.068,73	2.055,64	270,80	15	285,80	344,17
Ejecución 2	909,09	87	1.085,32	2.050,47	270,80	15	285,80	344,20
Ejecución 3	952,56	66	1.086,24	2.052,04	270,80	15	285,80	343,43
Ejecución 4	930,52	59	1.050,03	2.020,46	270,80	15	285,80	343,32
Ejecución 5	923,49	46	1.016,67	1.975,38	270,80	15	285,80	344,02
Media	926,90	66	1.061,40	2.030,80	270,80	15	285,80	343,83

Restricciones del 20 %

Resultados obtenidos por el MVO-BL-mej en el PAR con 20 % de restricciones

	Iris				**Rand**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	0,60	0	0,6	232,05	0,75	0	0,75	230,30
Ejecución 2	0,60	0	0,6	231,83	0,75	0	0,75	230,09
Ejecución 3	0,60	0	0,6	232,01	0,75	0	0,75	230,70
Ejecución 4	0,60	0	0,6	233,13	0,75	0	0,75	230,39
Ejecución 5	0,60	0	0,6	232,70	0,75	0	0,75	230,65
Media	0,60	0	0,6	232,34	0,75	0	0,75	230,42

Resultados obtenidos por el MVO-BL-mej en el PAR con 20 % de restricciones

	Ecoli				**Newthyroid**			
	Tasa_C	***Tasa_inf***	***Agr.***	**T**	***Tasa_C***	***Tasa_inf***	***Agr.***	**T**
Ejecución 1	953,97	96	1.051,20	2.679,66	270,80	41	302,33	427,39
Ejecución 2	956,18	100	1.057,46	2.676,92	270,80	41	302,33	427,52
Ejecución 3	940,83	138	1.080,59	2.594,93	270,80	41	302,33	426,81
Ejecución 4	930,60	183	1.115,94	2.585,00	270,80	41	302,33	429,85
Ejecución 5	914,22	147	1.063,10	2.478,99	270,80	41	302,33	429,11
Media	939,16	133	1.073,66	2.603,10	270,80	41	302,33	428,14