



UNIVERSIDAD DE GRANADA

METAHEURÍSTICAS
GRADO EN INGENIERÍA INFORMÁTICA

PRÁCTICA 2

TÉCNICAS DE BÚSQUEDA BASADAS EN POBLACIONES PARA
EL PROBLEMA DEL AGRUPAMIENTO CON RESTRICCIONES

Autor

Ignacio Vellido Expósito
ignacioove@correo.ugr.es
79056166Z



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

CURSO 2019-2020

Índice

1. Introducción	2
1.1. Descripción del problema	2
1.2. Consideraciones previas	2
1.2.1. Función objetivo	2
1.2.2. Representación de la solución	3
2. Algoritmos	4
2.1. Greedy COPKM v1	5
2.2. Búsqueda Local	6
2.3. Greedy COPKM v2	7
2.4. Algoritmos Genéticos	8
2.4.1. Variantes Elitistas	9
2.4.2. Variantes Estacionarias	9
2.5. Algoritmos Meméticos	10
3. Experimentación	11
3.1. Greedy COPKM v1	13
3.2. Búsqueda Local	19
3.3. Greedy COPKM v2	25
3.4. Algoritmo Generacional Elitista Uniforme	31
3.5. Algoritmo Generacional Elitista por Segmento Fijo	37
3.6. Algoritmo Generacional Estacionario Uniforme	43
3.7. Algoritmo Generacional Estacionario por Segmento Fijo	49
3.8. Algoritmo Memético (10, 1.0)	55
3.9. Algoritmo Memético (10, 0.1)	61
3.10. Algoritmo Memético (10, 0.1mej)	67
4. Conclusiones	73
4.1. Greedy COPKM v1	73
4.2. Greedy COPKM v2	73
4.3. Búsqueda Local	74
4.4. Algoritmos Genéticos	75
4.5. Algoritmos Meméticos	76
A. Representación visual de primeras dimensiones	77
A.1. Greedy COPKM	77
A.2. Búsqueda Local	79
A.3. Greedy COPKM v2	81
A.4. Algoritmo Generacional Elitista Uniforme	83
A.5. Algoritmo Generacional Elitista por Segmento Fijo	85
A.6. Algoritmo Generacional Estacionario Uniforme	87
A.7. Algoritmo Generacional Estacionario por Segmento Fijo	89
A.8. Algoritmo Memético (10, 1.0)	91
A.9. Algoritmo Memético (10, 0.1)	93
A.10. Algoritmo Memético (10, 0.1mej)	95
Referencias	98

1. Introducción

1.1. Descripción del problema

El Problema del Agrupamiento (PA) es un problema clásico de aprendizaje no supervisado, que consiste agrupar una serie de instancias en un número concreto de clústers de forma lógica. En estas prácticas añadimos restricciones al problema convirtiéndolo en el **Problema del Agrupamiento con Restricciones** (PAR), una variante NP-Completa semi-supervisada de PA.

En PAR por tanto se debe agrupar una serie de datos en un número predefinido de clústers, teniendo que contener cada clúster como mínimo un elemento. Además, tenemos dos tipos de restricciones, asociadas a pares de elementos:

- Must-Link (ML): Ambos elementos deben pertenecer al mismo clúster.
- Cannot-Link (CL): Ambos elementos deben pertenecer a distintos clústers.

A la hora de implementar los algoritmos se considerarán estas restricciones como débiles, es decir, serán relevantes a la hora de determinar la calidad de la solución pero no la consideración de si una posible solución lo es.

Trabajaremos con 4 instancias del problema:

1. **Iris**: Características de tres tipos de flor de Iris. Contiene 3 clases y 4 dimensiones.
2. **Ecoli**: Características de células. 8 clases y 7 dimensiones.
3. **Rand**: Conjunto de datos artificial de dos dimensiones formado por 3 clústers y 2 dimensiones.
4. **Newthyroid**: Glándulas tiroides de 2015 pacientes. 3 clases y 2 dimensiones

1.2. Consideraciones previas

1.2.1. Función objetivo

La función objetivo en el problema PAR se calcula en base a la fórmula:

$$f = \bar{C} + (\text{infeasibility} * \lambda) \quad (1)$$

Siendo:

$$\lambda = \frac{\max\{d_i \in D\}}{|R|} \quad \text{tal que } D = \text{Distancias} \quad (2)$$

$$\text{infeasibility} = \sum_{i=0}^{|ML|} \mathbb{1}(h_C(\overrightarrow{ML_{[i,1]}}) \neq (h_C(\overrightarrow{ML_{[i,2]}})) + \sum_{i=0}^{|CL|} \mathbb{1}(h_C(\overrightarrow{CL_{[i,1]}}) = (h_C(\overrightarrow{CL_{[i,2]}})) \quad (3)$$

$$\bar{C} = \frac{1}{k} \sum_{c_i \in C} \|\vec{x}_j - \vec{u}_j\|_2 \quad (4)$$

Que es calculada en pseudocódigo:

Algorithm 1: Función objetivo

```
C = Distancia media intra-cluster ;
λ = Distancia máxima en el conjunto de datos / nº de restricciones ;
inf = Nº restricciones no cumplidas ;
return C + (λ * inf)
```

Algorithm 2: Distancia media intra-cluster

```
Input: Conjunto de datos, solución, centroides
Separar conjunto de datos según su cluster ;
for particion del conjunto de datos do
| Calcular distancia media de sus elementos al centroide correspondiente ;
end
return C
```

Algorithm 3: Infeasibility

```
Input: s: solución
inf = 0 ;
for r in lista_restricciones do
| if r = ML and s[r[0]] ≠ s[r[1]] then
| | inf++ ;
| else if r = CL and s[r[0]] = s[r[1]] then
| | inf++ ;
end
return inf
```

1.2.2. Representación de la solución

Una solución se representa como un vector de igual longitud que el conjunto de datos, indicando en cada casilla el clúster al que pertenece el elemento i-ésimo. Adicionalmente, es necesario que cada clúster cuente como mínimo con un elemento.

2. Algoritmos

Todos los algoritmos se han implementado a mano en el lenguaje Python con la ayuda de los framework Scikit-Learn y Numpy. En algunos casos se ha seguido código de terceros (de StackOverflow) para la resolución de pequeños problemas (mejor forma de calcular distancias, cómo generar permutaciones de arrays, etc.), y sólamente tras la comprobación de su correctitud. Adicionalmente, el cálculo de los centroides se ha hecho a partir del módulo *NearestCentroid* de Scikit.

Para la ejecución del código es necesario tener instalado Python en su versión 3, y se puede ejecutar de la siguiente manera:

```
$python main.py [-h] -a A -p P [-s S] [-ss SS]
```

>Argumentos :

-h, --help	Mensaje de ayuda con esta descripción
-a A	Algoritmo a utilizar (copkm, bl, copkm2)
-p P	Problema a ejecutar (iris10, ecoli20...)
-s S	Archivo que contiene la semilla
-ss SS	Valor de la semilla

Adicionalmente, es necesario tener incluido Numpy, Scikit-Learn y Matplotlib en el entorno de Python donde se ejecute.

La generación de soluciones aleatorias hace uso de la librería Numpy y genera un vector aleatorio lleno con posibles clústeres. Este proceso se llama repetidamente hasta que la solución es válida (no deja ningún clúster vacío).

Algorithm 4: Generación de soluciones aleatorias

```
Input: Tamaño de la solución, número de clusters
do
| solution = X donde  $\forall x_i \in X, 0 \leq x_i < numClust$  ;
while solution no es válida;
return solution
```

2.1. Greedy COPKM v1

El primer algoritmo con el que afrontamos PAR es la técnica *Greedy COPKM*, variante de la versión Greedy clásica adaptada al problema PAR.

El proceso que sigue el algoritmo es el siguiente:

Algorithm 5: Algoritmo COPKM

```

Input: Conjunto de datos, lista de restricciones
solution = solución aleatoria ;
indexes = Conjunto de índices (uno por cada dato) ;
centroids = Conjunto con los centroides de cada clúster ;
Barajar indexes ;
while solution cambie do
    actualSol = solución vacía ;
    for i in indexes do
        inf = máxima infeasibility posible ;
        foreach cluster valido do
            newSol = actualSol modificando el cluster del índice i ;
            newInf = infeasibility de newSol ;
            if newInf < inf then
                | Almacenar como cluster posible ;
            else if newInf < inf then
                | Añadir a la lista de clusters posibles ;
            end
            distances = Distancias de los elementos a cada cluster ;
            actualSol se modifica con el cluster de los posibles con menor distancia ;
        end
        solution = actualSol ;
    end
    return solution, centroids

```

2.2. Búsqueda Local

La generación de vecinos se separa en dos pasos. Puesto que creamos un vecindario virtual (de forma compacta) como pares índice-cluster, al inicio del proceso BL se genera todo el posible vecindario, que corresponde a una permutación del número de elementos del problema con el número de clústeres. Una vez generado esta lista inmutable de pares, en cada iteración se seleccionan aquellos que sean válidos en la solución actual. En este caso un vecino es válido cuando no deja ningún clúster vacío y no pertenece a la solución actual.

Algorithm 6: Generación de vecinos

```
Input: Conjunto de vecinos virtuales posibles, en forma de lista de pares, solution
vecindario = listaVecinosVirtuales ;
for v in vecindario do
    | soluciónVecina = Solución producida aplicando el vecino v ;
    | Eliminar v de vecindario si soluciónVecina deja algún cluster vacío o es igual
    | que solution ;
end
return vecindario
```

El proceso de búsqueda queda de la siguiente manera:

Algorithm 7: Proceso de búsqueda

```
Input: Conjunto de datos, lista de restricciones
solution = Solución aleatoria ;
centroids = Conjunto con los centroides de cada clúster ;
neighborhood = Permutación con todos los vecinos virtuales posibles ;
evaluations = 0 ;
cost = Valor de la función objetivo para la solución actual ;
while solution cambie and evaluations < 100 000 do
    | neigh = Vecinos virtuales válidos para la solution actual ;
    | for n in neigh do
        |     evaluations++ ;
        |     newSolution = Solución aplicando el vecino n ;
        |     newCentroids = Centroides de newSolution ;
        |     newCost = Valor de la función objetivo para newSolution ;
        |     if newCost < cost then
        |         |         cost = newCost ;
        |         |         solution = newSolution ;
        |         |         Saltar a la siguiente iteración del bucle while ;
        |     end
    | end
| end
return solution, centroids
```

2.3. Greedy COPKM v2

Puesto que COPKM desecha la solución de la iteración anterior y solo actualiza en base a los nuevos centroides, haciendo que en ocasiones la infeasibility aumente y el algoritmo cicle, se decide modificar el código para no desperdiciar los resultados anteriores.

En términos de programación, el único cambio es no comenzar con una solución vacía en cada iteración y calcular la infeasibility no hasta el índice actual sino con toda.

Algorithm 8: Algoritmo COPKM v2

```

Input: Conjunto de datos, lista de restricciones
solution = Solución aleatoria ;
indexes = Conjunto de índices (uno por cada dato) ;
centroids = Conjunto con los centroides de cada clúster ;
Barajar indexes ;
while solution cambie do
    for i in indexes do
        inf = infeasibility de la solución actual ;
        foreach cluster valido do
            newSol = solution modificando el cluster del índice i ;
            newInf = infeasibility de newSol ;
            if newInf < inf then
                | Almacenar como cluster posible ;
            else if newInf < inf then
                | Añadir a la lista de clusters posibles ;
            end
            distances = Distancias de los elementos a cada cluster ;
            solution se modifica con el cluster de los posibles con menor distancia ;
        end
    end
return solution, centroids

```

2.4. Algoritmos Genéticos

Dentro de los algoritmos genéticos nos encontramos con 4 variantes diferentes: dos elitistas y dos estacionarias. En común estos algoritmos se caracterizan por seguir el esquema selección-cruce-mutación-reemplazamiento, implementado siguiendo el siguiente proceso:

Algorithm 9: Implementación abstracta de los algoritmos genéticos

Input: Conjunto de datos, lista de restricciones
 population = Población inicial de soluciones aleatorias válidas (ordenada en base al coste) ;
 ev = Número de evaluaciones inicial ;
 numT = Número de veces a aplicar torneo binario ;
while $ev < 100000$ **do**
 | newPopulation = Población obtenida aplicando torneo binario numT veces ;
 | Recombinar newPopulation ;
 | Mutar newPopulation ;
 | Aplicar esquema de reemplazamiento ;
 | Evaluar población ;
 | $ev +=$ Tamaño de la población ;
end
return Mejor solución de la población, centroids

La recombinación y la mutación se aplica un número determinado de veces (dependiente del algoritmo concreto) para reducir el coste computacional que añade la generación de números aleatorios.

Algorithm 10: Torneo binario

Input: population, numT: Número de veces a aplicar el torneo
 newPopulation = [] ;
repeat numT **times**
 | Seleccionar dos cromosomas aleatorios de population ;
 | best = Cromosoma con menor coste de los dos ;
 | Añadir best a newPopulation ;
end
return newPopulation

Algorithm 11: Reparación de cromosomas

Input: chromosome
while existe clúster vacío en chromosome **do**
 | Añadir clústers restantes a posiciones aleatorias de chromosome ;
end
return chromosome

Algorithm 12: Operador de mutación

```
Input: chromosome
do
| Modificar clúster de posición aleatoria ;
while exista clúster vacío en chromosome;
return Mejor solución de la población, centroids
```

Algorithm 13: Cruce uniforme

```
Input: parent1, parent2
n = Longitud de un cromosoma ;
indexes = Array con  $n/2$  índices aleatorios ;
child = parent1 ;
child[indexes] = parent2[indexes] ;
return child
```

Algorithm 14: Cruce por segmento fijo

```
Input: parent1, parent2
index = Posición aleatoria ;
n = Longitud de un cromosoma ;
Asignar a child  $n/2$  clústers de un padre y  $n/2$  del otro a partir de index ;
return child
```

Los hiperparámetros usados son: población de 50 cromosomas; probabilidad de cruce del 0,7 en las variantes elitistas y del 1 en las estacionarias; probabilidad de mutación del 0,001; 100.000 evaluaciones de la función objetivo como criterio de parada.

2.4.1. Variantes Elitistas

Las variantes elitistas generan una nueva población en cada iteración del algoritmo mediante el torneo binario. Esta población sustituye totalmente a la anterior, pero antes se comprueba si el mejor cromosoma ha sobrevivido en la iteración. En caso de que no lo haya hecho, se sustituye por el peor de la nueva población.

2.4.2. Variantes Estacionarias

Las variantes estacionarias solo eligen dos padres mediante torneo binario, siendo estos los que sufren el proceso evolutivo. Al final de la solución sustituyen a los peores cromosomas de la población, independientemente de que sean mejores o peores que ellos.

2.5. Algoritmos Meméticos

Tras la mutación, optimizamos cromosomas de la población mediante una búsqueda local suave. Aplicada cada 10 iteraciones a un porcentaje específico de la población (dependiente del algoritmo).

Algorithm 15: Optimización mediante Búsqueda Local suave

```

Input: population: Población a optimizar
n = Longitud de un cromosoma ;
maxErrors =  $n/2 * 0.1$  ;
for chrom in population do
    indexes = Conjunto de índices aleatorios ;
    errors = 0 ;
    improving = True ;
    for i in indexes do
        if !improving and errors  $\geq maxErrors$  then
            | break ;
        end
        improving = False ;
        cost = Coste del cromosoma chrom ;
        for c in nuevos clústers posibles do
            newCost = Coste asignando c a chrom ;
            if newCost < cost then
                | improving = True ;
                Asignar c a chrom ;
                cost = newCost ;
            end
        end
        if !improving then
            | errors++ ;
        end
    end

```

Como vemos en la primera página de *Experimentación*, el algoritmo AGG-UN es el que tiene mejor valor de agregado en media. Es por tanto con este esquema genético bajo el que se implementan los algoritmos meméticos.

Contamos con 3 variantes de estos algoritmos, que difieren en la población que recibe. Estos tipos son:

1. AM-(10,1.0): Se optimiza cada cromosoma de la población.
2. AM-(10,0.1): Se optimiza un 10 % de la población.
3. AM-(10,0.1mej): Se optimiza el 10 % con mejor coste.

Para todas las subvariantes se utilizan los mismos hiperparámetros: población de 10 cromosomas; probabilidad de cruce de 0,7 y de mutación de 0,001. El número máximo de evaluaciones permitidas es, al igual que en los genéticos, de 100.000.

3. Experimentación

Semillas	
Ejecución 1	949004259
Ejecución 2	589741062
Ejecución 3	277451237
Ejecución 4	49258669
Ejecución 5	3773969821

	Agregado
BL	26.63
COPKM v1	14.67
COPKM v2	12.69
AGG-UN	12.17
AGG-SF	13.62
AGE-UN	27.94
AGE-SF	29.24
AM-(10,1.0)	12.43
AM-(10,0.1)	13.77
AM-(10,0.1mej)	13.09

Cuadro 1: Valores medios del agregado para cada algoritmo

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
BL	0.92	141.20	1.82	27.50	46.74	1212.80	79.28	733.33	1.12	106.00	1.88	29.21	14.16	296.80	25.02	132.10
COPKM v1	0.67	3.60	0.56	2.24	37.04	201.40	42.44	432.70	0.77	6.20	0.81	1.60	14.30	144.20	19.57	6.18
COPKM v2	0.67	0.00	0.67	2.44	37.50	57.20	39.01	117.84	0.76	0.00	0.76	2.02	14.29	0.00	14.29	6.88
COPKM v2	0.67	0.00	0.67	2.44	37.50	57.20	39.01	117.84	0.76	0.00	0.76	2.02	14.29	0.00	14.29	6.88
AGG-UN	0.67	0.00	0.67	126.31	24.85	208.00	30.43	492.50	0.72	2.20	0.74	125.32	11.94	59.80	14.13	219.31
AGG-SF	0.67	0.00	0.67	115.38	28.30	380.20	38.50	450.45	0.72	0.00	0.72	115.90	12.98	45.60	14.65	201.60
AGE-UN	0.84	119.60	1.60	12.35	41.41	1374.80	78.30	35.30	0.97	63.00	1.43	12.31	15.77	373.72	32.54	18.53
AGE-SF	0.87	148.20	1.81	13.00	42.31	1420.40	80.42	36.48	1.25	141.00	2.27	13.21	15.05	549.20	35.13	19.20
AM-(10,1.0)	0.67	0.00	0.67	180.30	26.08	171.40	30.68	706.73	0.72	4.40	0.76	182.13	13.01	53.80	14.98	313.52
AM-(10,0.1)	0.67	0.00	0.67	139.89	28.46	269.40	35.69	640.27	0.72	0.00	0.72	138.58	13.58	104.00	17.38	257.87
AM-(10,0.1mej)	0.67	0.00	0.67	137.71	26.33	252.20	33.09	648.11	0.72	0.00	0.72	138.53	13.36	73.40	16.05	252.86

Cuadro 2: Resultados medios de todos los algoritmos para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
BL	0.90	264.80	1.74	40.18	46.85	2175.20	76.03	1242.45	1.15	293.20	2.21	38.25	14.54	576.20	25.08	182.36
COPKM v1	0.67	3.40	0.68	3.55	35.08	169.40	37.36	274.13	0.77	13.60	0.82	2.66	14.18	40.23	15.12	10.54
COPKM v2	0.67	0.00	0.67	3.69	31.10	0.00	31.10	145.25	0.76	0.00	0.76	3.26	14.29	0.00	14.29	12.13
AGG-UN	0.67	0.00	0.67	184.51	26.93	627.80	35.35	841.16	0.72	0.00	0.72	180.31	12.86	96.80	14.63	363.59
AGG-SF	0.67	3.60	0.68	172.73	29.68	510.80	36.53	776.87	0.72	0.00	0.72	179.15	14.81	90.80	16.47	327.34
AGE-UN	0.79	217.20	1.48	16.57	40.49	2822.60	78.36	55.53	0.86	86.80	1.17	17.35	15.31	729.20	28.64	26.65
AGE-SF	0.83	265.00	1.67	17.30	40.73	2868.60	79.21	57.20	1.01	168.00	1.62	17.30	15.92	869.20	31.81	27.76
AM-(10,1.0)	0.67	0.00	0.67	276.60	29.98	353.40	34.72	1216.98	0.72	0.00	0.72	273.49	12.99	176.40	16.21	508.64
AM-(10,0.1)	0.67	5.40	0.69	211.32	28.20	561.20	35.73	1121.72	0.72	0.00	0.72	212.34	13.92	254.40	18.57	419.69
AM-(10,0.1mej)	0.69	14.00	0.73	210.80	28.59	576.20	36.32	1125.28	0.72	2.40	0.73	211.14	14.24	120.80	16.45	418.82

Cuadro 3: Resultados medios de todos los algoritmos para 20 % de restricciones

3.1. Greedy COPKM v1

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.67	7.00	0.06	2.27	36.48	153.00	40.59	184.56	0.76	0.00	0.76	2.59	14.25	41.00	15.75	5.08
Ejecución 2	0.67	0.00	0.67	2.33	38.22	168.00	42.73	148.97	0.76	0.00	0.76	1.31	14.65	559.00	35.10	5.53
Ejecución 3	0.67	0.00	0.67	2.41	35.61	216.00	41.40	868.75	0.81	31.00	1.04	1.77	14.42	43.00	16.00	6.69
Ejecución 4	0.67	11.00	0.74	2.33	37.99	296.00	45.93	395.77	0.76	0.00	0.76	1.14	14.06	4.00	14.20	5.05
Ejecución 5	0.67	0.00	0.67	1.88	36.90	174.00	41.57	565.45	0.76	0.00	0.76	1.19	14.09	74.00	16.80	8.55
Media	0.67	3.60	0.56	2.24	37.04	201.40	42.44	432.70	0.77	6.20	0.81	1.60	14.30	144.20	19.57	6.18

Cuadro 4: Resultados COPKM v1 para 10 % de restricciones

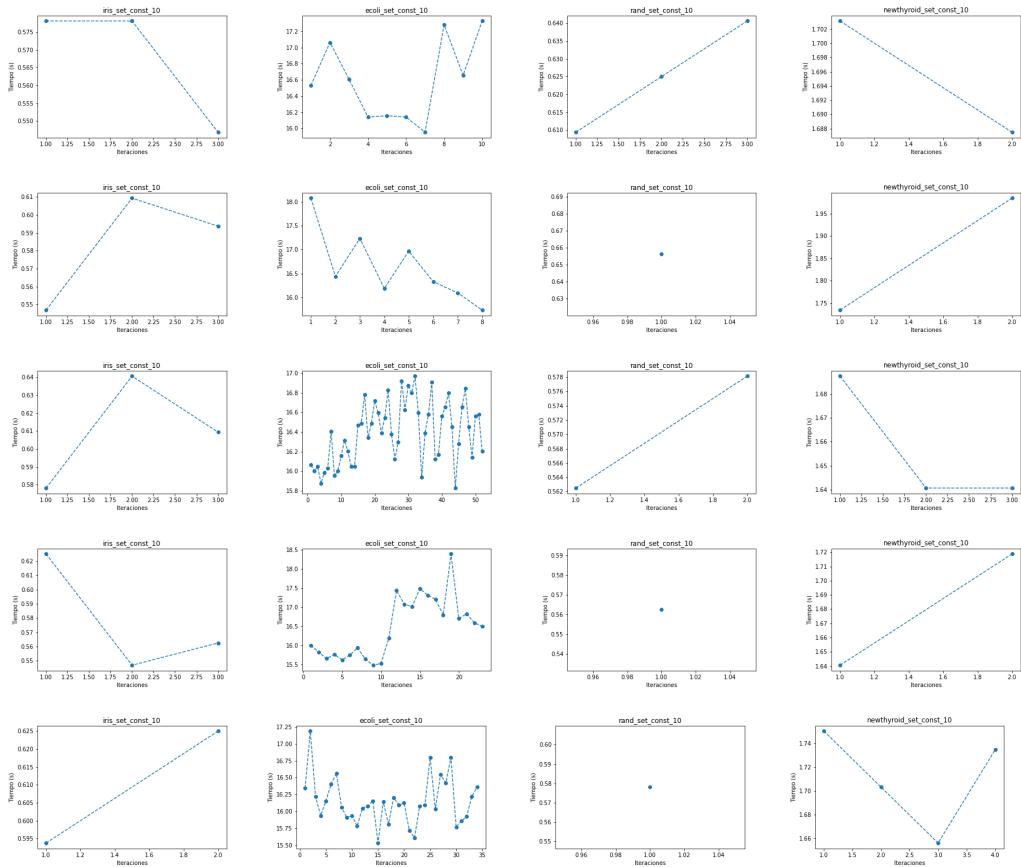


Figura 1: Tiempos de COPKM v1 para 10 % de restricciones

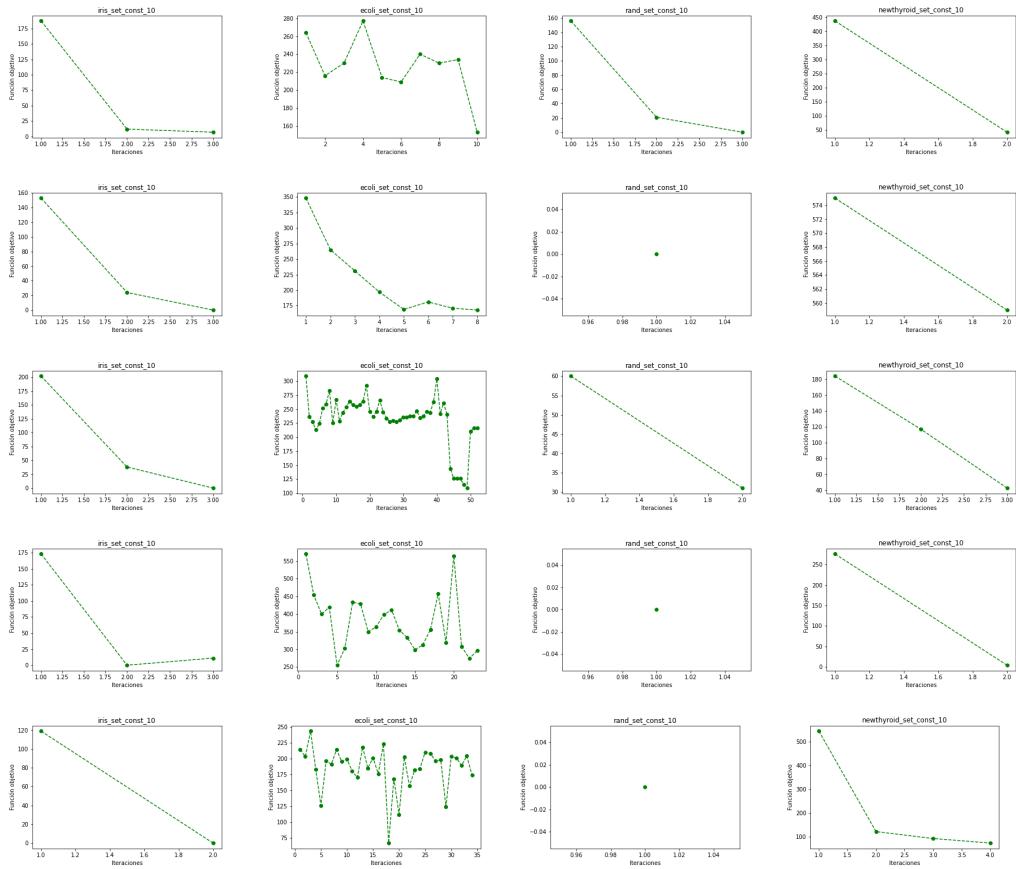


Figura 2: Agregado de COPKM v1 para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.67	17.00	0.72	3.36	34.44	351.00	39.15	169.77	0.76	0.00	0.76	3.38	14.20	67.00	15.42	10.00
Ejecución 2	0.67	0.00	0.67	2.33	37.64	158.00	39.76	283.50	0.76	0.00	0.76	2.19	14.13	30.00	14.68	9.89
Ejecución 3	0.67	0.00	0.67	5.56	33.53	128.00	35.25	366.92	0.83	68.00	1.08	3.28	14.23	60.00	15.33	9.83
Ejecución 4	0.67	0.00	0.67	3.22	32.35	92.00	33.58	392.08	0.76	0.00	0.76	2.31	14.23	30.00	14.78	13.19
Ejecución 5	0.67	0.00	0.67	3.30	37.46	118.00	39.04	158.39	0.76	0.00	0.76	2.14	14.13	14.13	15.39	9.81
Media	0.67	3.40	0.68	3.55	35.08	169.40	37.36	274.13	0.77	13.60	0.82	2.66	14.18	40.23	15.12	10.54

Cuadro 5: Resultados COPKM v1 para 20 % de restricciones

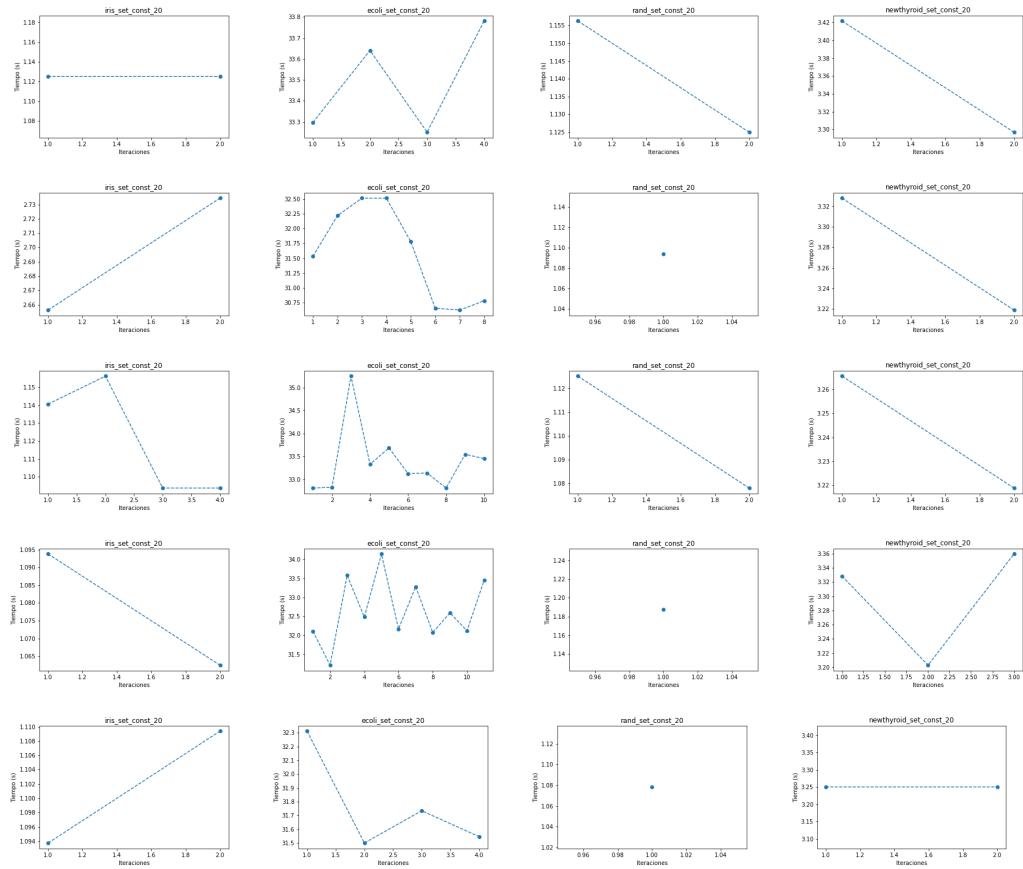


Figura 3: Tiempos de COPKM v1 para 20 % de restricciones

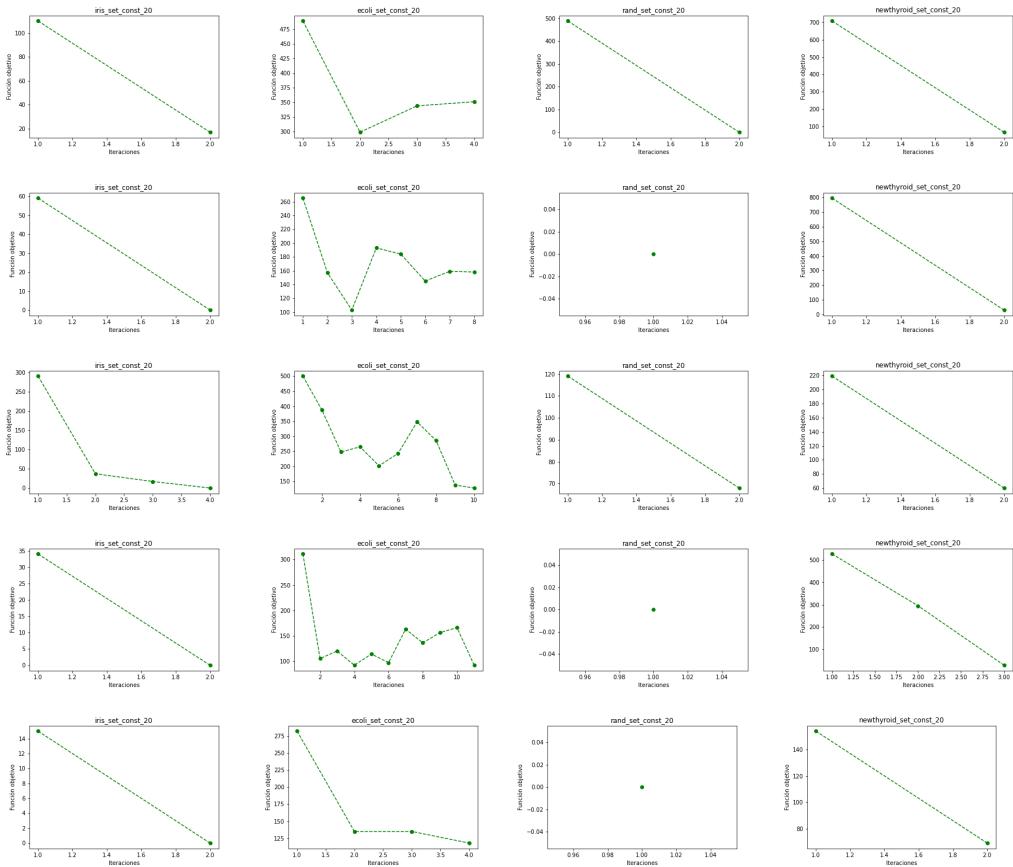


Figura 4: Agregado de COPKM v1 para 20 % de restricciones

3.2. Búsqueda Local

En todas las ejecuciones se utilizó un valor λ igual al cociente entre la mayor distancia entre dos elementos cualesquiera del conjunto de datos y el número de restricciones del problema.

$$\lambda = \frac{\max D}{|R|} \quad (5)$$

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	1.07	241.00	2.60	17.27	46.46	1188.00	78.34	747.25	1.22	131.00	2.17	27.70	14.87	271.00	24.78	157.25
Ejecución 2	1.02	193.00	2.24	20.77	46.63	1207.00	79.01	712.41	1.23	133.00	2.19	21.05	14.27	221.00	22.36	128.27
Ejecución 3	1.02	133.00	1.86	30.67	46.63	1228.00	79.58	741.86	1.14	133.00	2.10	35.20	14.40	388.00	28.59	122.80
Ejecución 4	0.76	75.00	1.23	29.66	47.39	1197.00	79.50	753.13	1.04	78.00	1.61	29.38	13.05	302.00	24.10	141.84
Ejecución 5	0.76	64.00	1.16	39.13	46.60	1244.00	79.98	712.03	0.95	55.00	1.35	32.73	14.22	302.00	25.27	110.33
Media	0.92	141.20	1.82	27.50	46.74	1212.80	79.28	733.33	1.12	106.00	1.88	29.21	14.16	296.80	25.02	132.10

Cuadro 6: Resultados BL para 10 % de restricciones

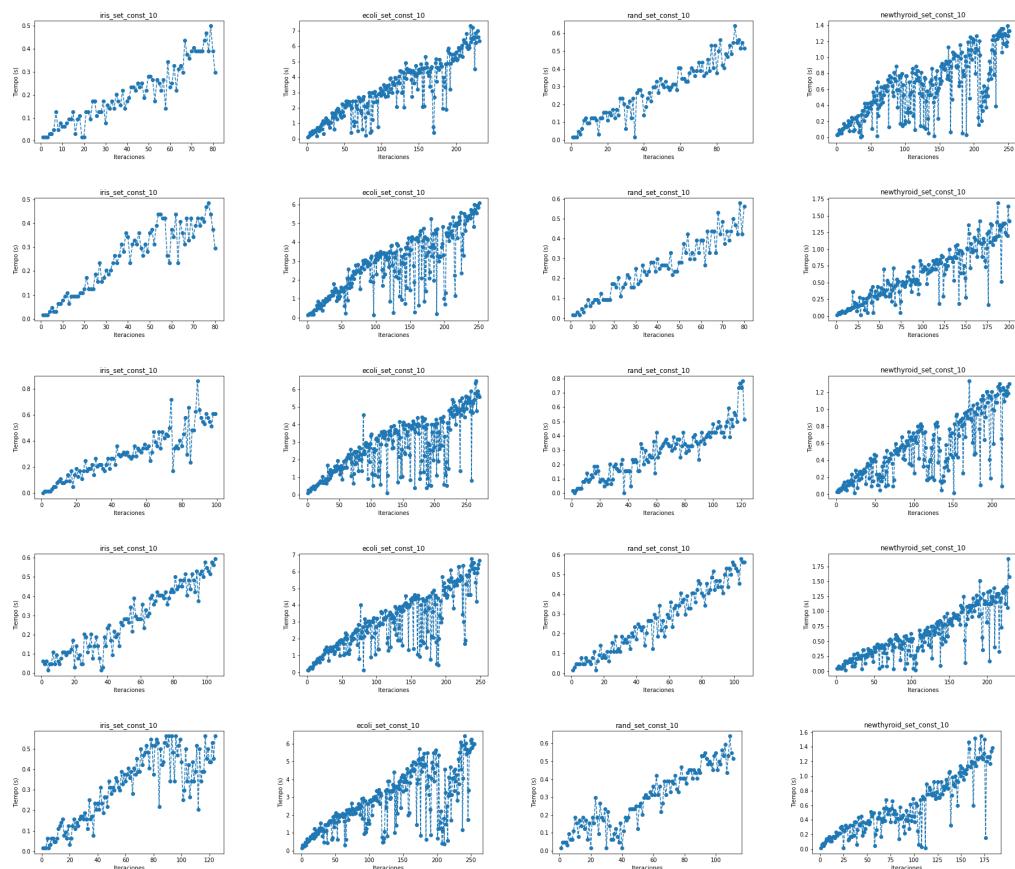


Figura 5: Tiempos de BL para 10 % de restricciones

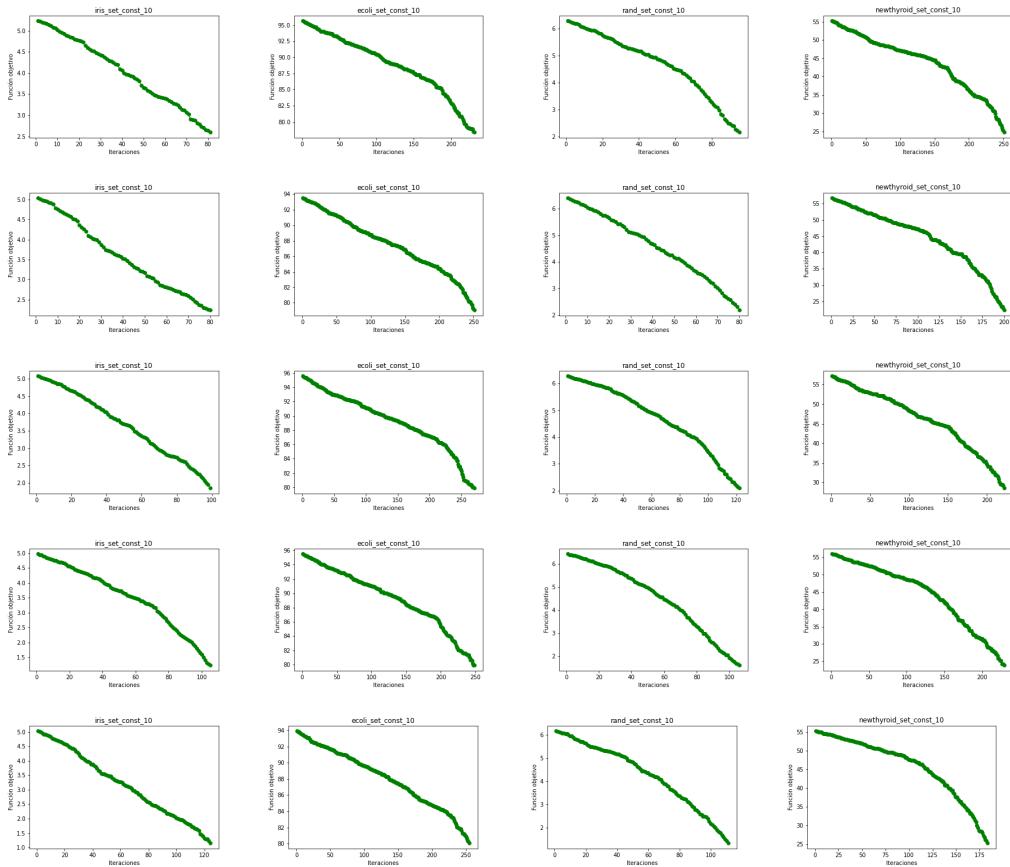


Figura 6: Agregado de BL para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.82	279.00	1.70	42.30	46.28	2133.00	74.89	1256.20	1.14	272.00	2.12	40.48	14.82	602.00	25.83	235.84
Ejecución 2	0.94	325.00	1.97	32.75	46.64	2194.00	76.07	1220.25	1.25	366.00	2.57	33.13	14.83	519.00	24.31	159.41
Ejecución 3	1.11	338.00	2.18	36.50	46.88	2178.00	76.10	1234.64	1.07	258.00	2.00	40.19	14.37	597.00	25.29	175.42
Ejecución 4	0.93	276.00	1.80	38.41	47.14	2190.00	76.52	1223.83	1.02	194.00	1.71	44.39	14.10	610.00	25.25	170.97
Ejecución 5	0.72	106.00	1.05	50.94	47.30	2181.00	76.56	1277.31	1.30	376.00	2.65	33.05	14.61	553.00	24.72	170.14
Media	0.90	264.80	1.74	40.18	46.85	2175.20	76.03	1242.45	1.15	293.20	2.21	38.25	14.54	576.20	25.08	182.36

Cuadro 7: Resultados BL para 20 % de restricciones

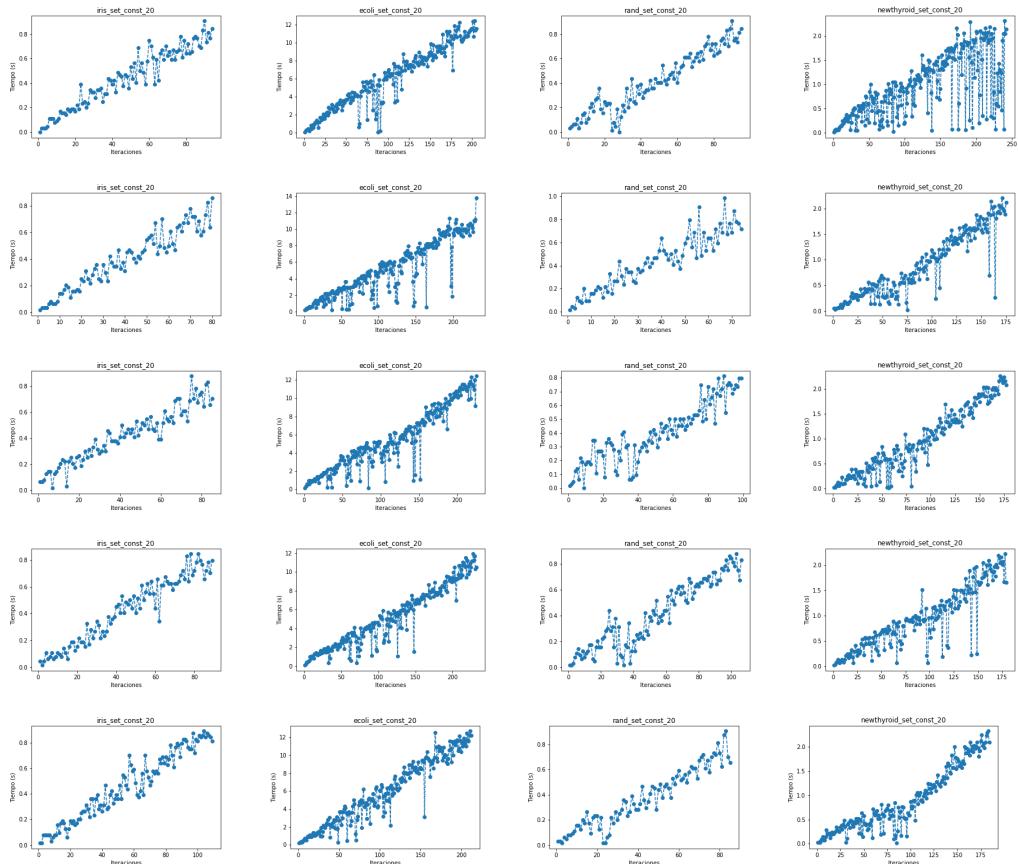


Figura 7: Tiempos de BL para 20 % de restricciones

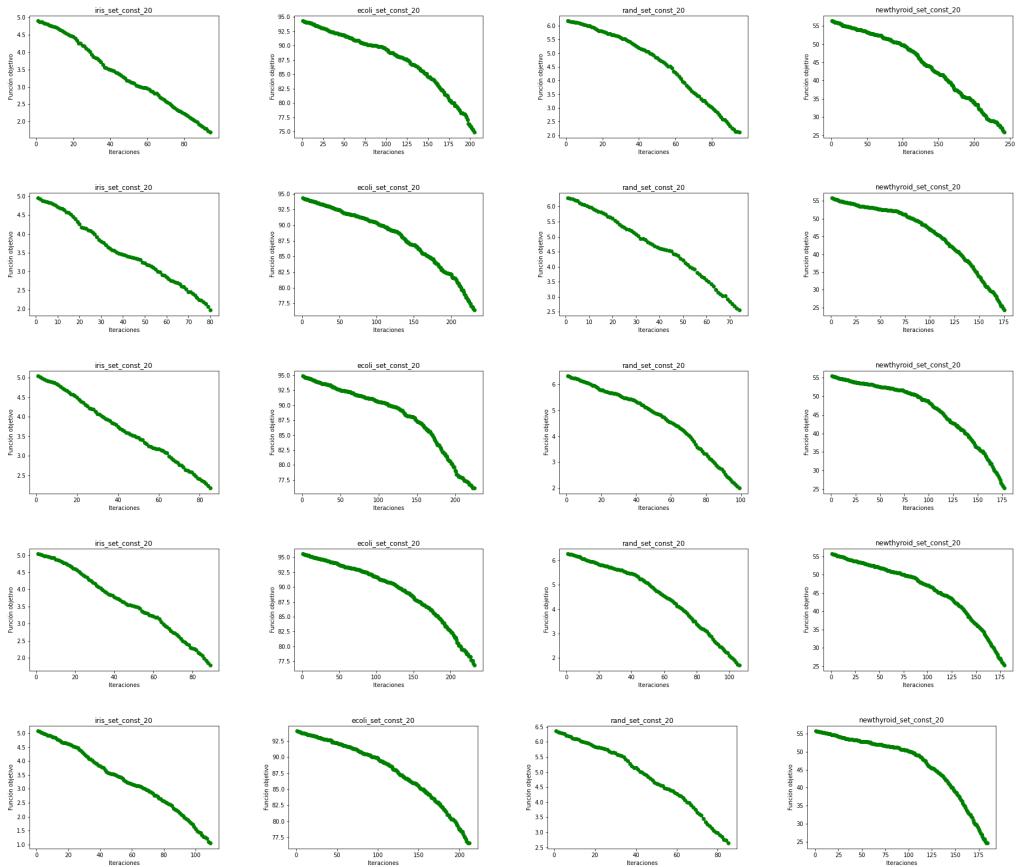


Figura 8: Agregado de BL para 20 % de restricciones

3.3. Greedy COPKM v2

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.67	0.00	0.67	2.11	34.74	2.00	34.80	89.83	0.76	0.00	0.76	2.25	14.29	0.00	14.29	6.23
Ejecución 2	0.67	0.00	0.67	2.72	29.37	0.00	29.37	152.09	0.76	0.00	0.76	1.66	14.29	0.00	14.29	6.39
Ejecución 3	0.67	0.00	0.67	2.89	57.53	277.00	64.96	59.25	0.76	0.00	0.76	2.27	14.29	0.00	14.29	8.56
Ejecución 4	0.67	0.00	0.67	2.22	33.71	4.00	33.71	106.02	0.76	0.00	0.76	1.69	14.29	0.00	14.29	6.97
Ejecución 5	0.67	0.00	0.67	2.25	32.15	3.00	32.23	182.00	0.76	0.00	0.76	2.25	14.29	0.00	14.29	6.22
Media	0.67	0.00	0.67	2.44	37.50	57.20	39.01	117.84	0.76	0.00	0.76	2.02	14.29	0.00	14.29	6.88

Cuadro 8: Resultados de COPKM v2 para 10 % de restricciones

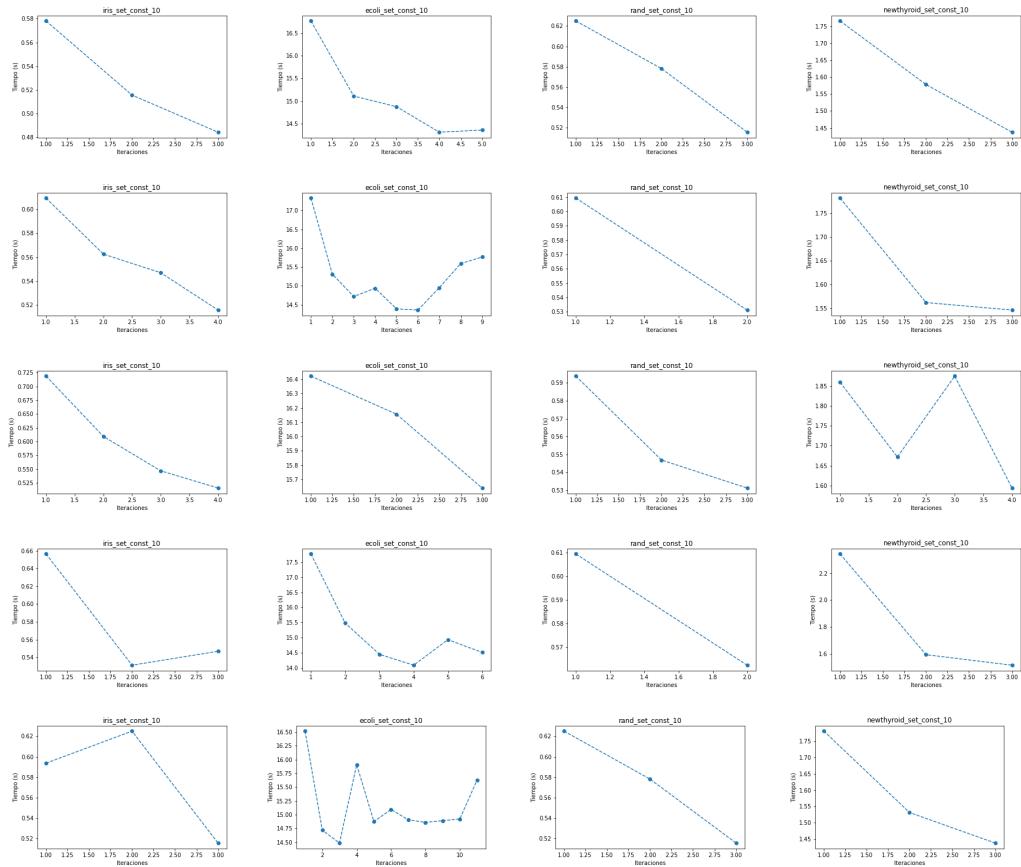


Figura 9: Tiempos de COPKM v2 para 10 % de restricciones

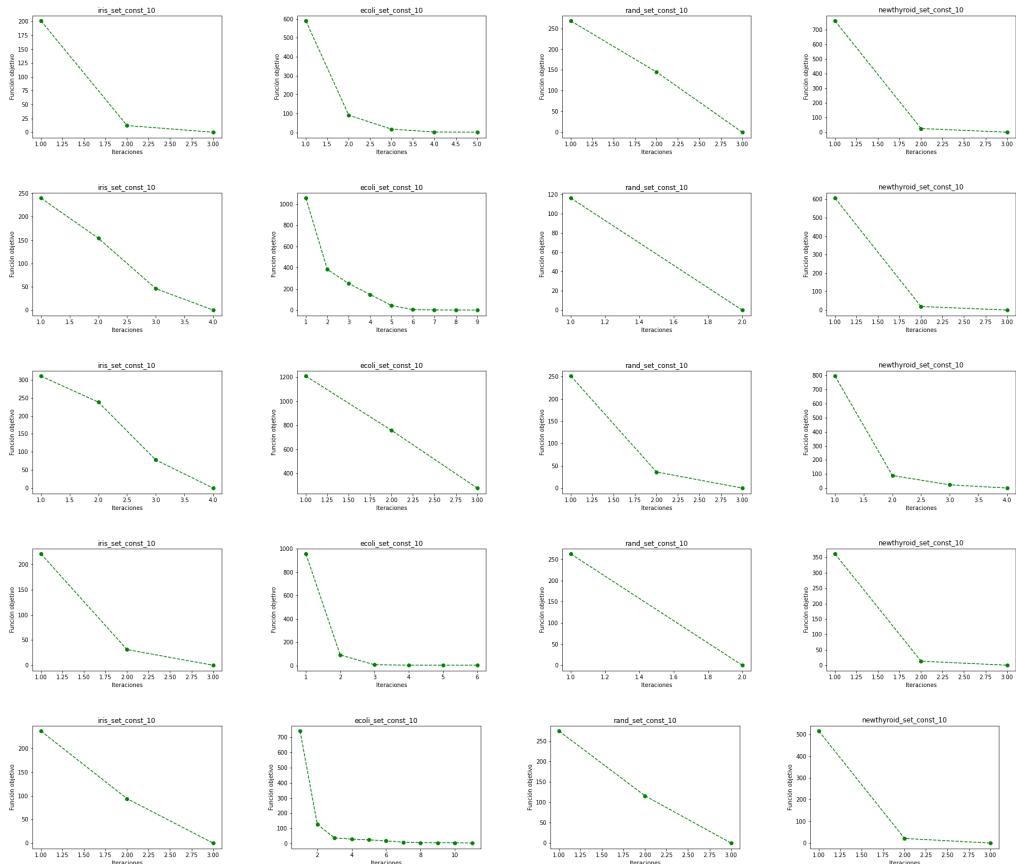


Figura 10: Agregado de COPKM v2 para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.67	0.00	0.67	3.25	29.61	0.00	29.61	183.00	0.76	0.00	0.76	3.19	14.29	0.00	14.29	14.91
Ejecución 2	0.67	0.00	0.67	4.42	33.46	0.00	33.46	149.48	0.76	0.00	0.76	3.34	14.29	0.00	14.29	12.33
Ejecución 3	0.67	0.00	0.67	4.33	34.43	0.00	34.43	93.83	0.76	0.00	0.76	3.30	14.29	0.00	14.29	9.36
Ejecución 4	0.67	0.00	0.67	3.17	29.61	0.00	29.61	149.45	0.76	0.00	0.76	3.34	14.29	0.00	14.29	11.91
Ejecución 5	0.67	0.00	0.67	3.27	28.40	0.00	28.40	150.48	0.76	0.00	0.76	3.14	14.29	0.00	14.29	12.14
Media	0.67	0.00	0.67	3.69	31.10	0.00	31.10	145.25	0.76	0.00	0.76	3.26	14.29	0.00	14.29	12.13

Cuadro 9: Resultados de COPKM v2 para 20 % de restricciones

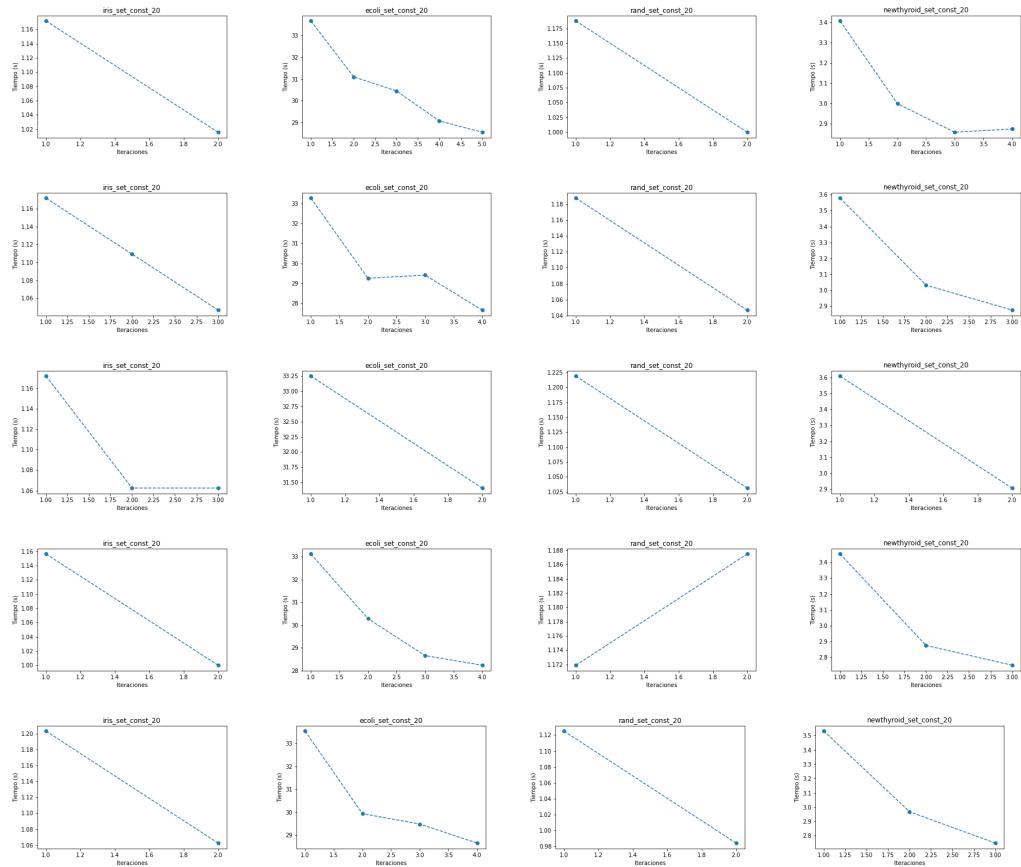


Figura 11: Tiempos de COPKM v2 para 20 % de restricciones

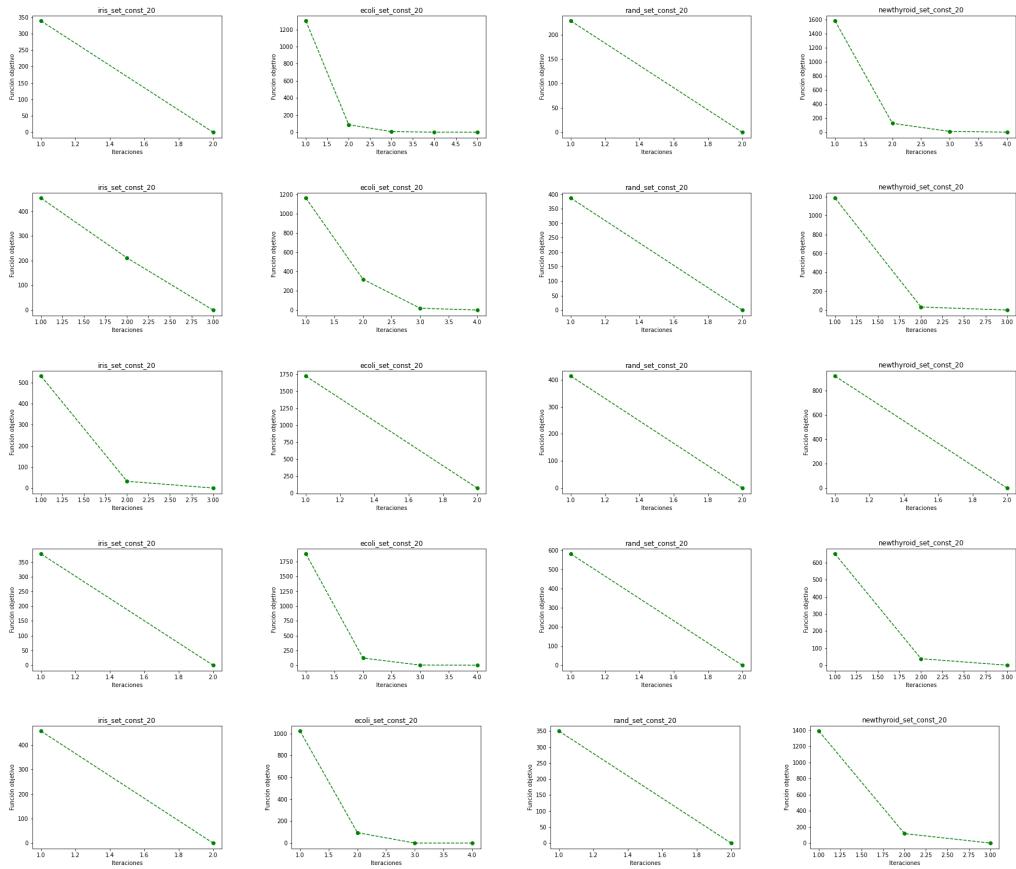


Figura 12: Agregado de COPKM v2 para 20 % de restricciones

3.4. Algoritmo Generacional Elitista Uniforme

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.67	0.00	0.67	119.33	27.93	229.00	34.07	465.02	0.76	11.00	0.83	121.05	10.69	95.00	14.16	207.22
Ejecución 2	0.67	0.00	0.67	121.77	22.30	188.00	27.34	471.38	0.72	0.00	0.72	113.72	10.74	96.00	14.25	210.44
Ejecución 3	0.67	0.00	0.67	121.08	24.96	122.00	28.24	469.80	0.72	0.00	0.72	123.89	10.60	96.00	14.12	206.75
Ejecución 4	0.67	0.00	0.67	135.66	23.70	198.00	29.01	532.44	0.72	0.00	0.72	135.42	13.83	6.00	14.05	249.30
Ejecución 5	0.67	0.00	0.67	133.72	25.34	303.00	33.47	523.86	0.72	0.00	0.72	132.50	13.83	6.00	14.05	222.83
Media	0.67	0.00	0.67	126.31	24.85	208.00	30.43	492.50	0.72	2.20	0.74	125.32	11.94	59.80	14.13	219.31

Cuadro 10: Resultados AGG-UN para 10 % de restricciones

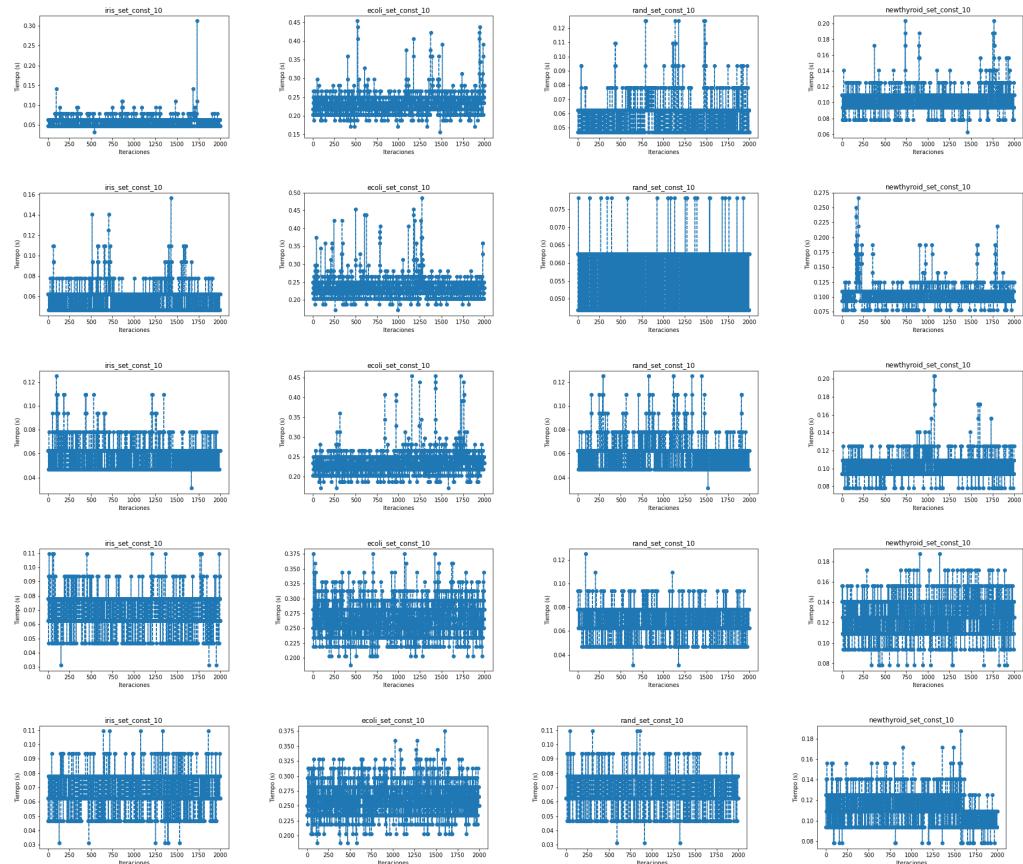


Figura 13: Tiempos de AGG-UN para 10 % de restricciones

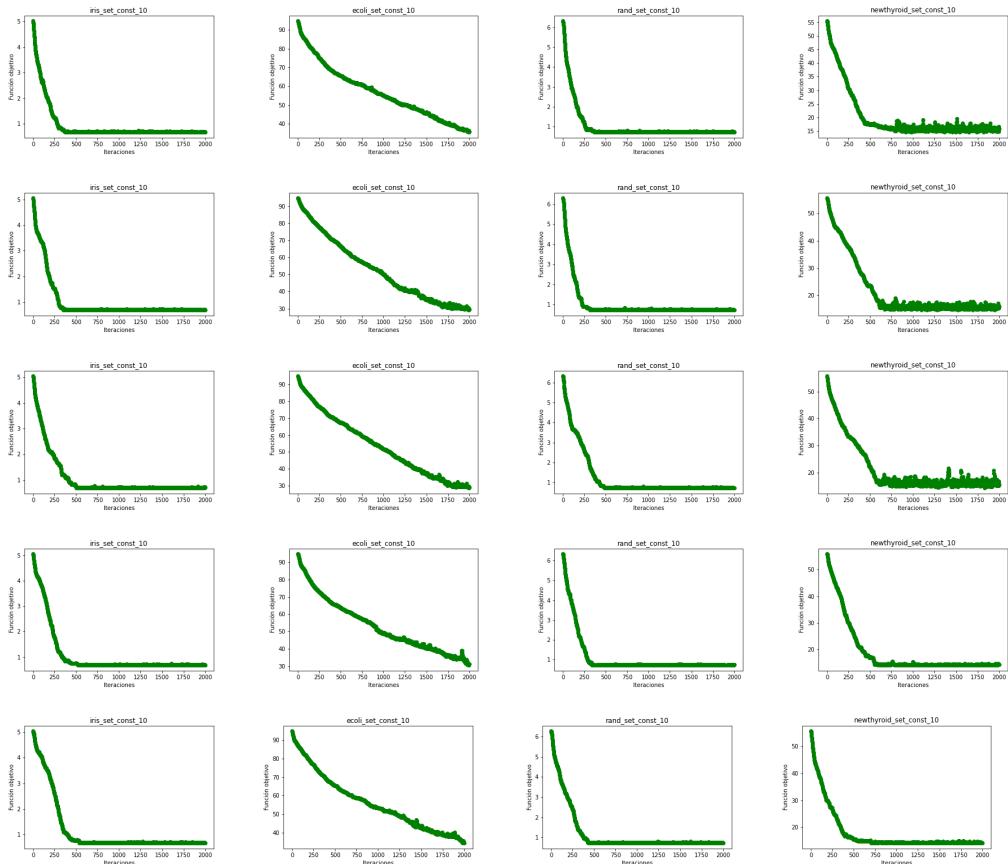


Figura 14: Agregado de AGG-UN para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.67	0.00	0.67	180.94	30.62	744.00	40.60	824.81	0.72	0.00	0.72	183.75	10.72	223.00	14.80	350.17
Ejecución 2	0.67	0.00	0.67	180.63	33.17	541.00	40.42	801.38	0.72	0.00	0.72	183.30	14.29	0.00	14.29	353.50
Ejecución 3	0.67	0.00	0.67	179.92	25.89	1029.00	39.69	809.19	0.72	0.00	0.72	185.27	14.33	34.00	14.95	384.67
Ejecución 4	0.67	0.00	0.67	205.75	22.92	480.00	29.36	967.41	0.72	0.00	0.72	171.22	10.66	227.00	14.81	381.72
Ejecución 5	0.67	0.00	0.67	175.33	22.05	345.00	26.67	803.00	0.72	0.00	0.72	178.00	14.29	0.00	14.29	347.89
Media	0.67	0.00	0.67	184.51	26.93	627.80	35.35	841.16	0.72	0.00	0.72	180.31	12.86	96.80	14.63	363.59

Cuadro 11: Resultados AGG-UN para 20 % de restricciones

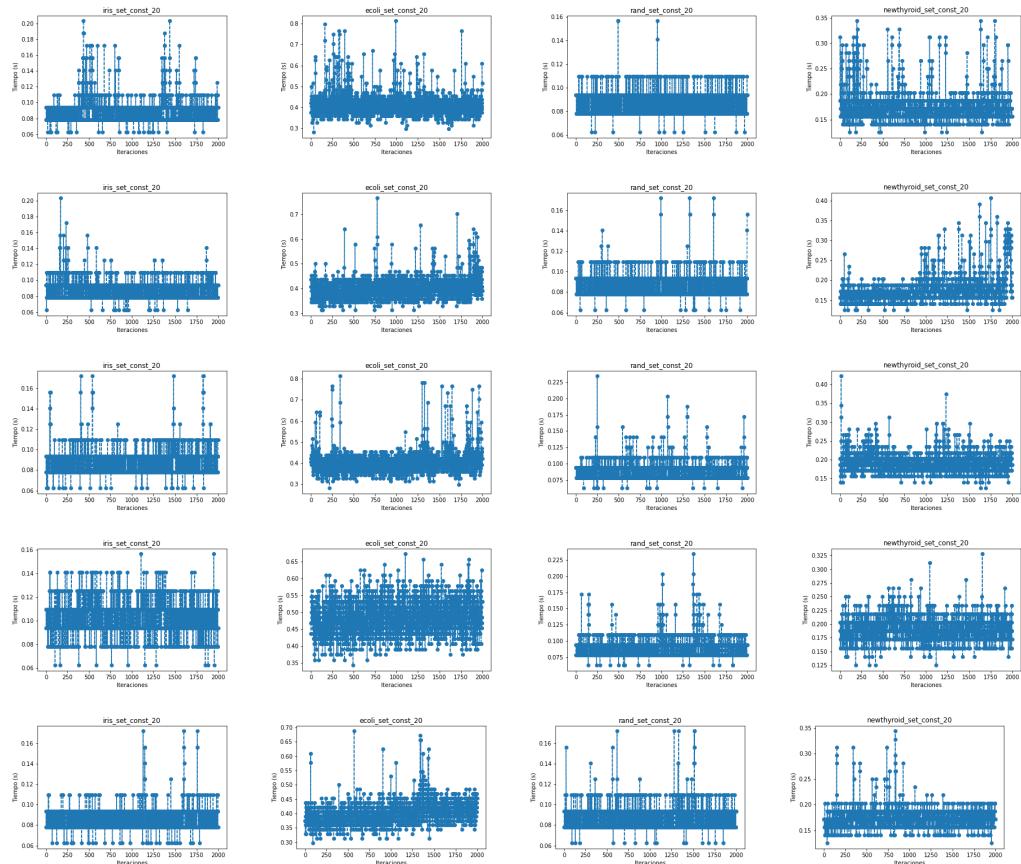


Figura 15: Tiempos de AGG-UN para 20 % de restricciones

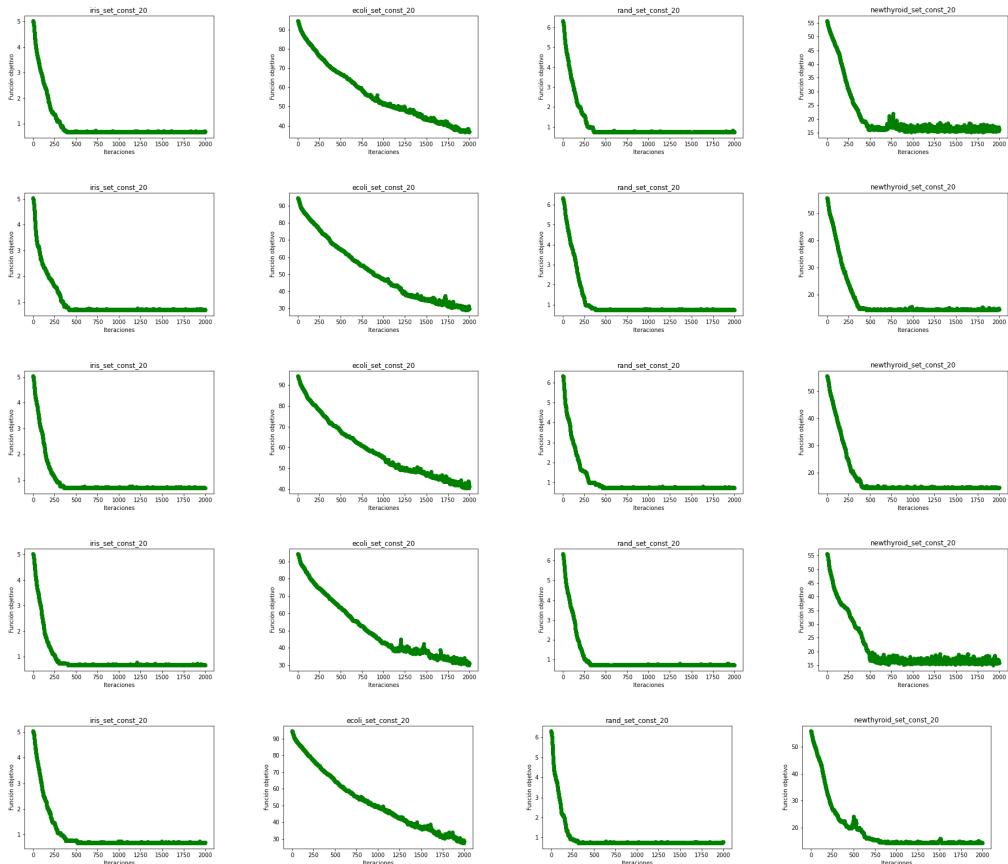


Figura 16: Agregado de AGG-UN para 20 % de restricciones

3.5. Algoritmo Generacional Elitista por Segmento Fijo

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.67	0.00	0.67	118.33	28.03	413.00	39.11	455.72	0.72	0.00	0.72	121.30	12.63	99.00	16.25	206.03
Ejecución 2	0.67	0.00	0.67	114.34	27.58	570.00	42.88	445.23	0.72	0.00	0.72	113.95	13.84	6.00	14.06	200.39
Ejecución 3	0.67	0.00	0.67	115.48	26.16	191.00	31.28	453.80	0.72	0.00	0.72	115.11	10.76	111.00	14.82	201.28
Ejecución 4	0.67	0.00	0.67	114.14	31.57	346.00	40.85	449.53	0.72	0.00	0.72	114.36	13.84	6.00	14.06	200.08
Ejecución 5	0.67	0.00	0.67	114.61	28.18	381.00	38.40	447.98	0.72	0.00	0.72	114.77	13.83	6.00	14.06	200.22
Media	0.67	0.00	0.67	115.38	28.30	380.20	38.50	450.45	0.72	0.00	0.72	115.90	12.98	45.60	14.65	201.60

Cuadro 12: Resultados AGG-SF para 10 % de restricciones

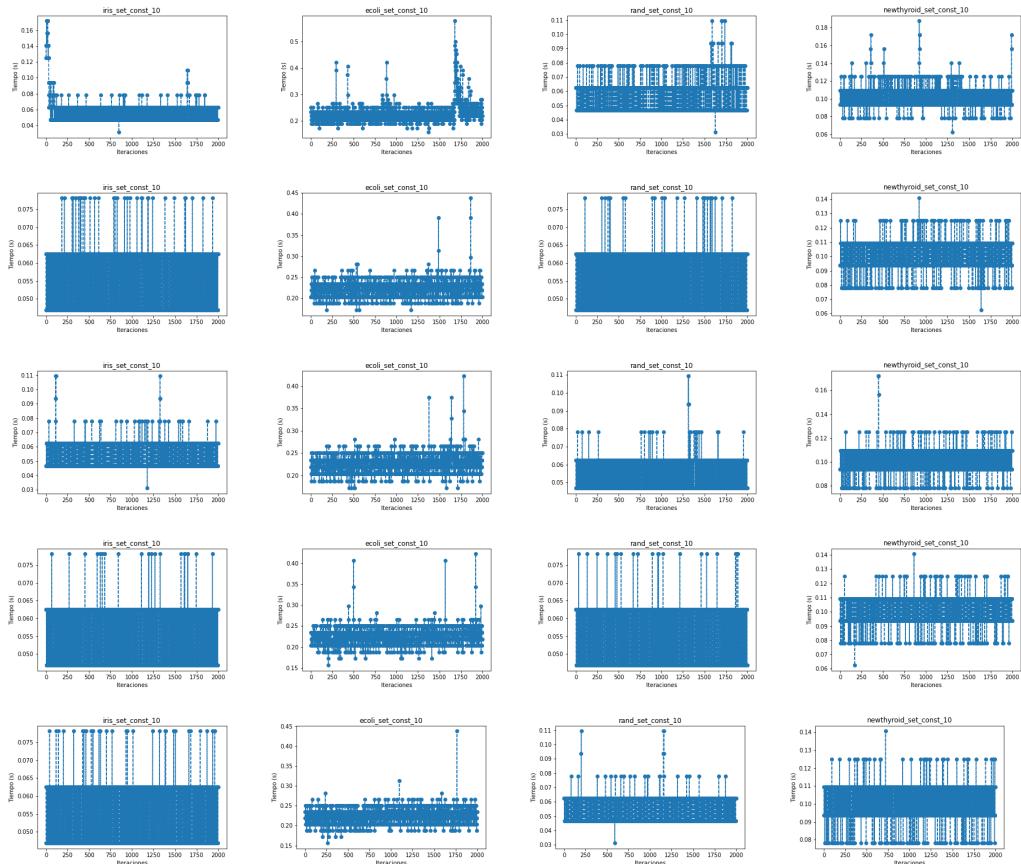


Figura 17: Tiempos de AGG-SF para 10 % de restricciones

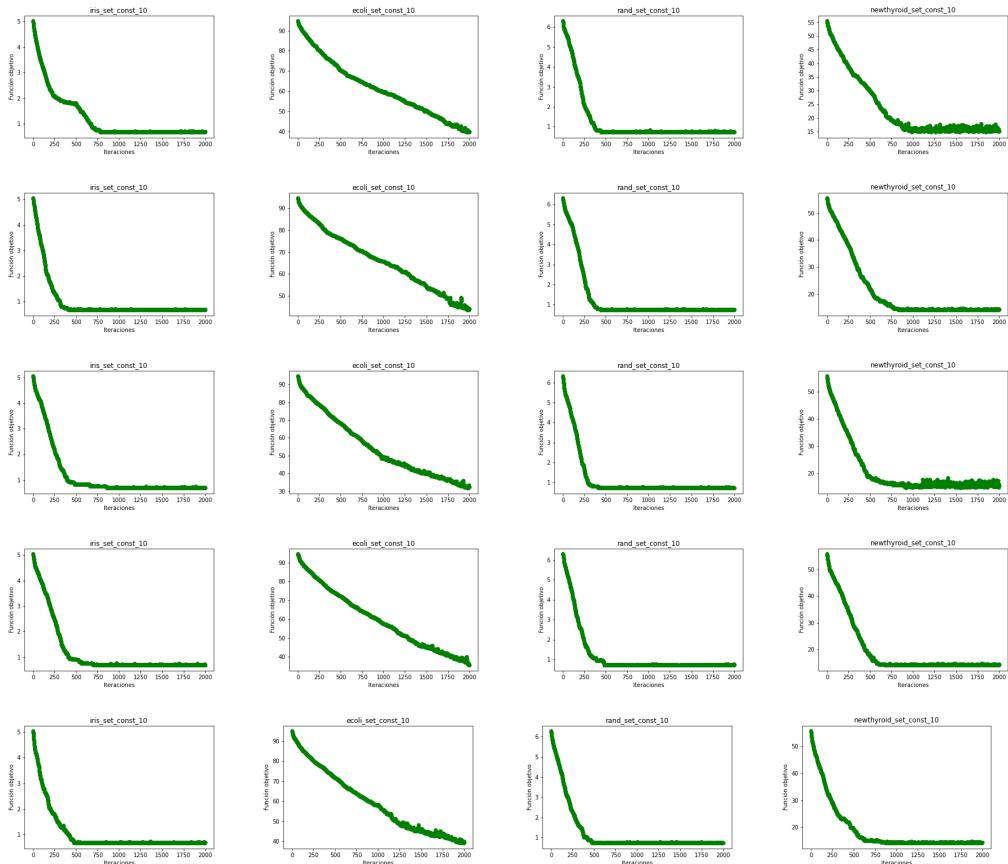


Figura 18: Agregado de AGG-SF para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.67	0.00	0.67	172.59	25.64	367.00	30.56	794.59	0.72	0.00	0.72	178.77	14.29	0.00	14.29	323.89
Ejecución 2	0.67	0.00	0.67	173.44	29.83	374.00	34.84	777.55	0.72	0.00	0.72	180.41	14.29	0.00	14.29	339.16
Ejecución 3	0.67	18.00	0.73	171.86	32.65	498.00	39.33	772.64	0.72	0.00	0.72	178.00	20.51	227.00	24.66	325.42
Ejecución 4	0.67	0.00	0.67	172.69	32.94	877.00	44.71	766.25	0.72	0.00	0.72	179.89	10.66	227.00	14.81	323.64
Ejecución 5	0.67	0.00	0.67	173.09	27.36	438.00	33.24	773.33	0.72	0.00	0.72	178.70	14.29	0.00	14.29	324.61
Media	0.67	3.60	0.68	172.73	29.68	510.80	36.53	776.87	0.72	0.00	0.72	179.15	14.81	90.80	16.47	327.34

Cuadro 13: Resultados AGG-SF para 20 % de restricciones

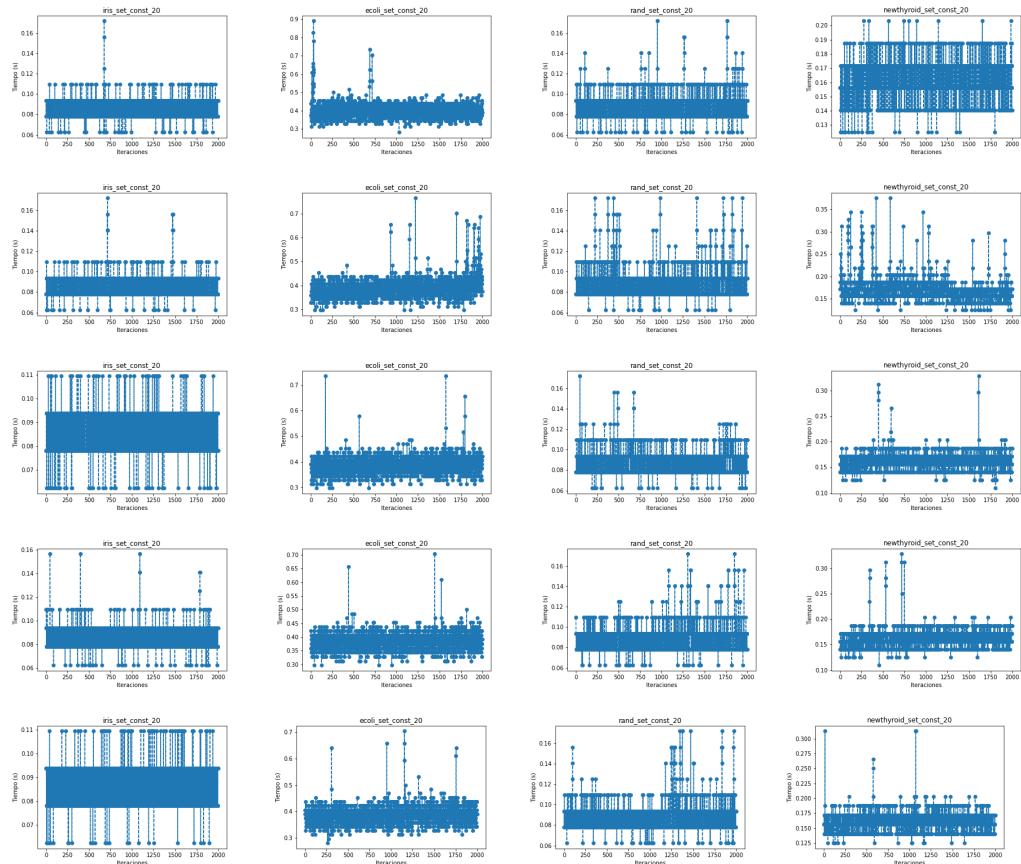


Figura 19: Tiempos de AGG-SF para 20 % de restricciones

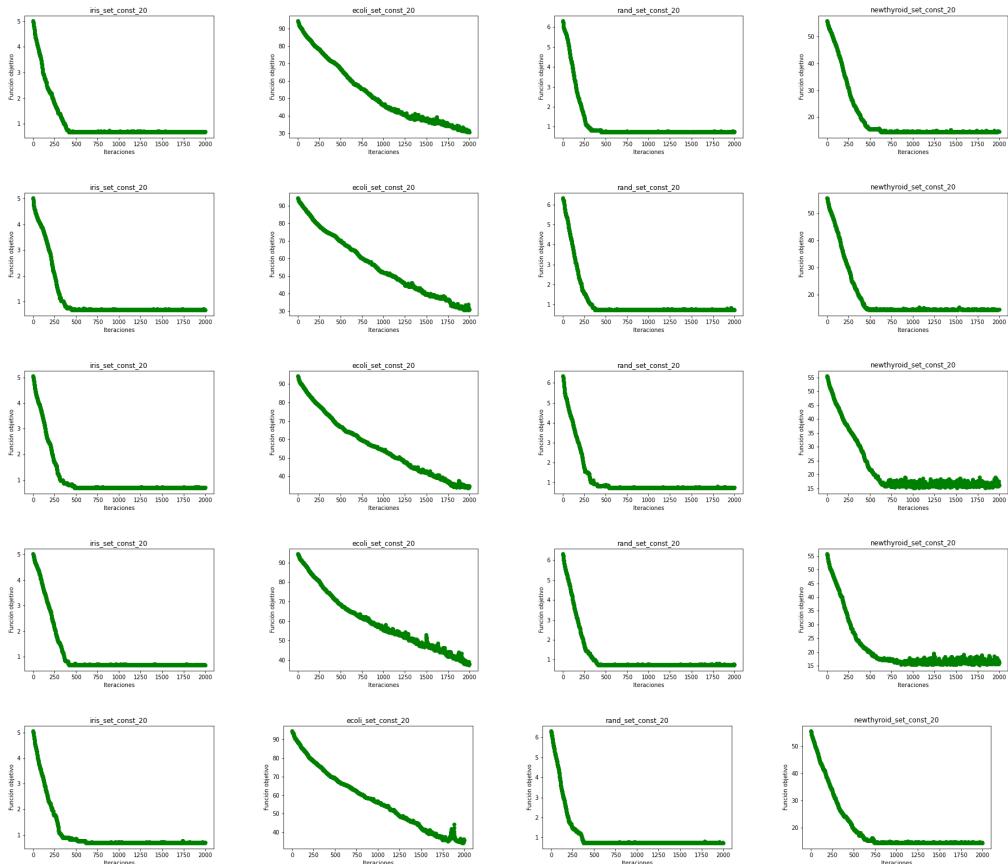


Figura 20: Agregado de AGG-SF para 20 % de restricciones

3.6. Algoritmo Generacional Estacionario Uniforme

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.85	137.00	1.71	12.34	41.23	1352.00	77.50	35.09	1.02	70.00	1.52	12.25	15.46	505.00	33.93	18.52
Ejecución 2	1.00	206.00	2.31	12.61	41.83	1396.00	79.29	34.88	0.91	48.00	1.26	12.36	15.61	15.61	31.67	18.55
Ejecución 3	0.79	78.00	1.29	12.17	41.85	1429.00	80.19	36.03	0.90	67.00	1.38	12.20	14.76	599.00	36.66	18.55
Ejecución 4	0.80	83.00	1.33	12.25	41.32	1309.00	76.44	35.19	1.02	74.00	1.55	12.34	16.48	428.00	32.13	18.48
Ejecución 5	0.78	94.00	1.38	12.39	40.84	1388.00	78.08	35.30	1.01	56.00	1.42	12.38	16.56	321.00	28.30	18.56
Media	0.84	119.60	1.60	12.35	41.41	1374.80	78.30	35.30	0.97	63.00	1.43	12.31	15.77	373.72	32.54	18.53

Cuadro 14: Resultados AGE-UN para 10 % de restricciones

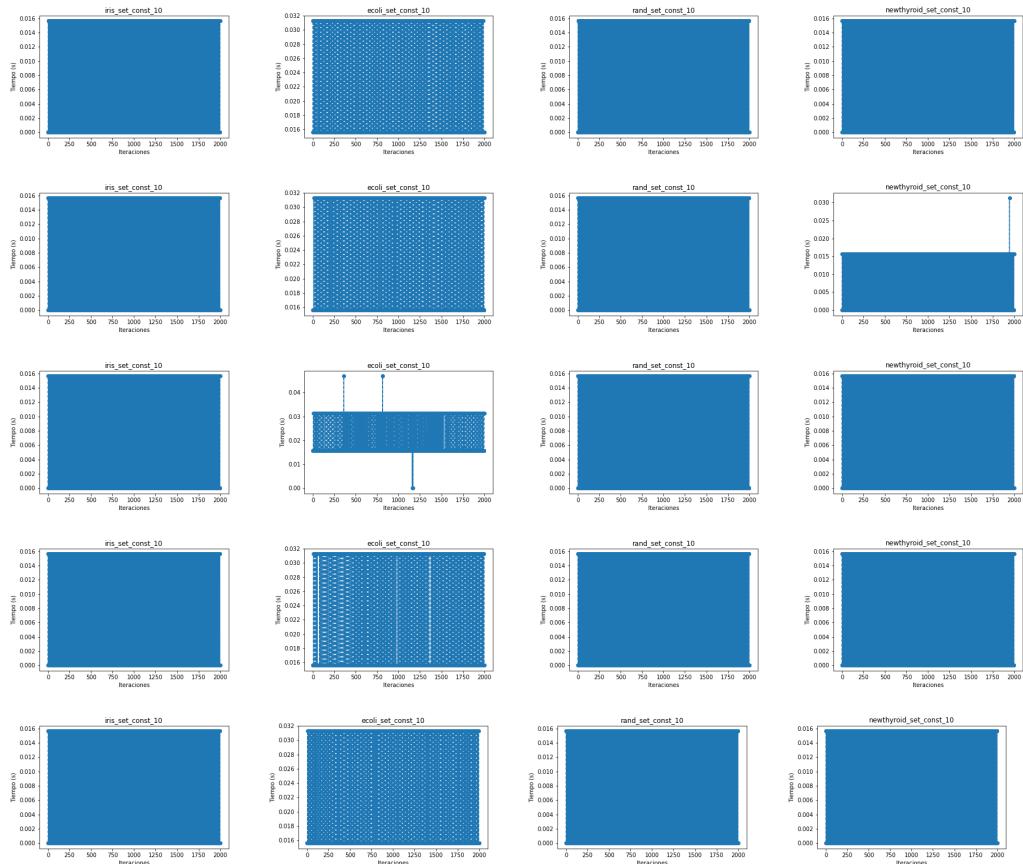


Figura 21: Tiempos de AGE-UN para 10 % de restricciones

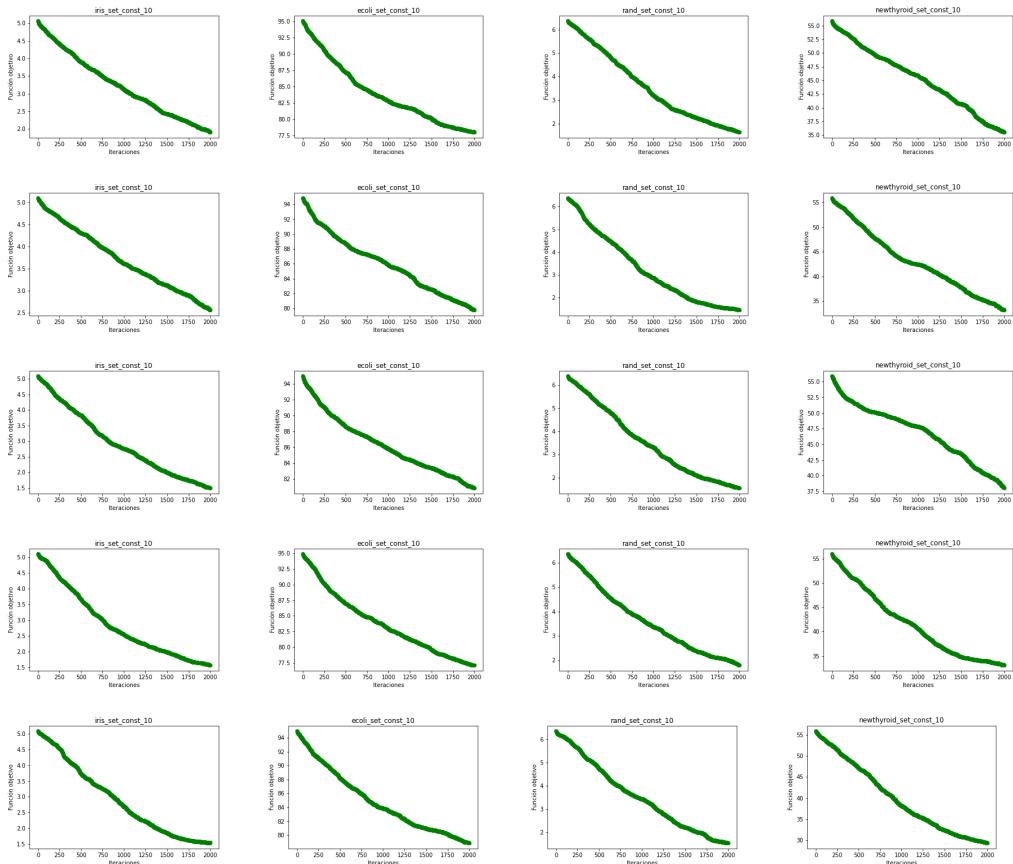


Figura 22: Agregado de AGE-UN para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.83	245.00	1.61	16.45	40.38	2958.00	80.06	55.41	0.82	59.00	1.03	17.33	13.93	1020.00	32.58	26.64
Ejecución 2	0.79	183.00	1.37	16.83	41.87	2715.00	78.29	56.53	0.81	48.00	0.98	17.02	14.66	509.00	23.97	26.86
Ejecución 3	0.83	415.00	2.15	16.48	39.24	2807.00	76.89	55.38	0.87	132.00	1.34	17.47	16.32	774.00	30.47	26.22
Ejecución 4	0.75	96.00	1.05	16.58	40.25	2913.00	79.33	55.00	0.91	86.00	1.22	17.84	15.71	763.00	29.65	26.73
Ejecución 5	0.77	147.00	1.23	16.52	40.73	2720.00	77.22	55.33	0.91	109.00	1.30	17.11	15.94	580.00	26.54	26.81
Media	0.79	217.20	1.48	16.57	40.49	2822.60	78.36	55.53	0.86	86.80	1.17	17.35	15.31	729.20	28.64	26.65

Cuadro 15: Resultados AGE-UN para 20 % de restricciones

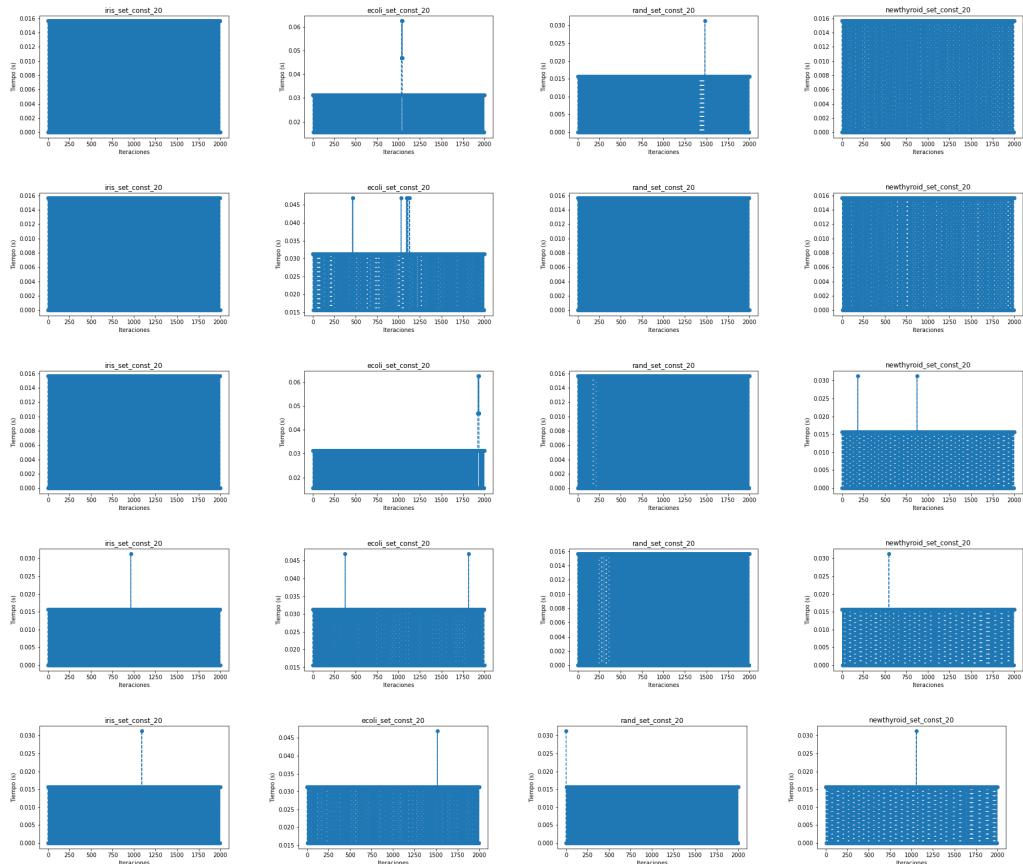


Figura 23: Tiempos de AGE-UN para 20 % de restricciones

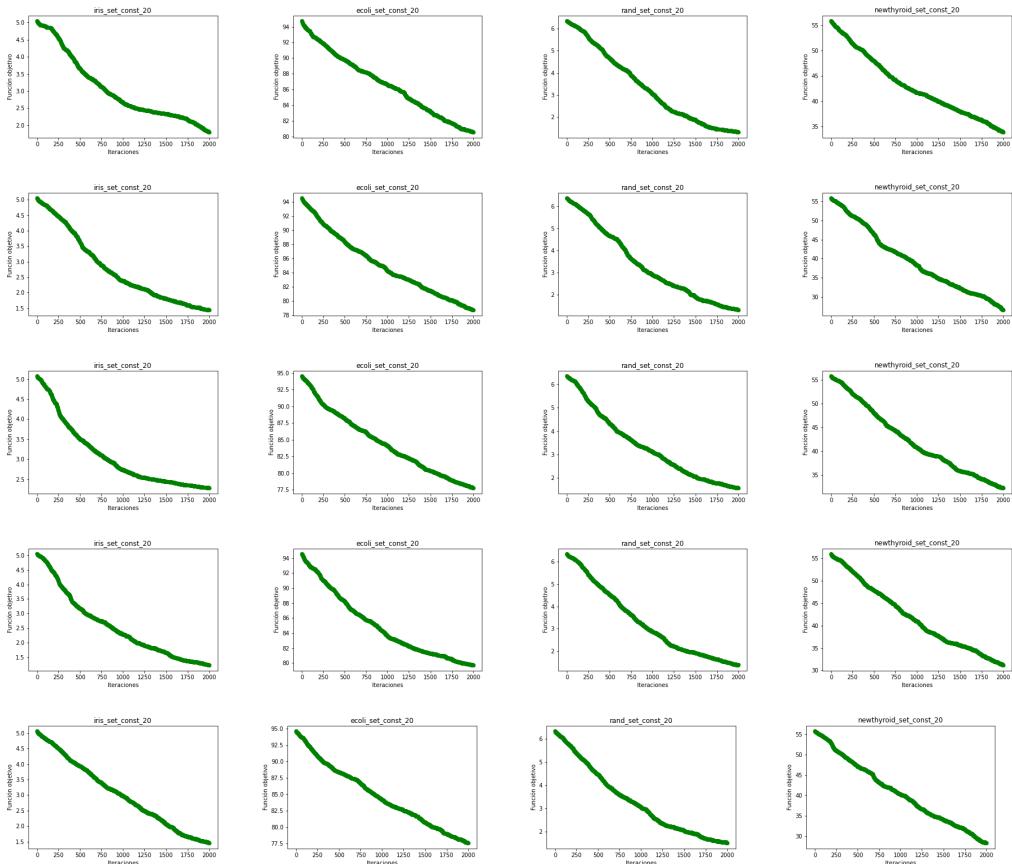


Figura 24: Agregado de AGE-UN para 20 % de restricciones

3.7. Algoritmo Generacional Estacionario por Segmento Fijo

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.95	189.00	2.15	13.45	41.96	1427.00	80.25	36.56	1.17	115.00	1.99	13.58	15.38	419.00	30.70	19.16
Ejecución 2	0.85	120.00	1.61	12.78	42.28	1425.00	80.51	36.17	1.53	178.00	2.81	12.94	15.79	502.00	34.15	19.77
Ejecución 3	0.88	189.00	2.08	12.94	41.00	1430.00	79.37	36.03	1.18	136.00	2.16	13.31	15.32	598.00	37.19	18.97
Ejecución 4	0.82	130.00	1.65	12.92	42.98	1408.00	80.76	37.53	1.18	146.00	2.23	13.03	15.18	549.00	35.25	18.86
Ejecución 5	0.86	113.00	1.58	12.92	43.34	1412.00	81.22	36.08	1.20	130.00	2.14	13.17	13.59	678.00	38.38	19.25
Media	0.87	148.20	1.81	13.00	42.31	1420.40	80.42	36.48	1.25	141.00	2.27	13.21	15.05	549.20	35.13	19.20

Cuadro 16: Resultados AGE-SF para 10 % de restricciones

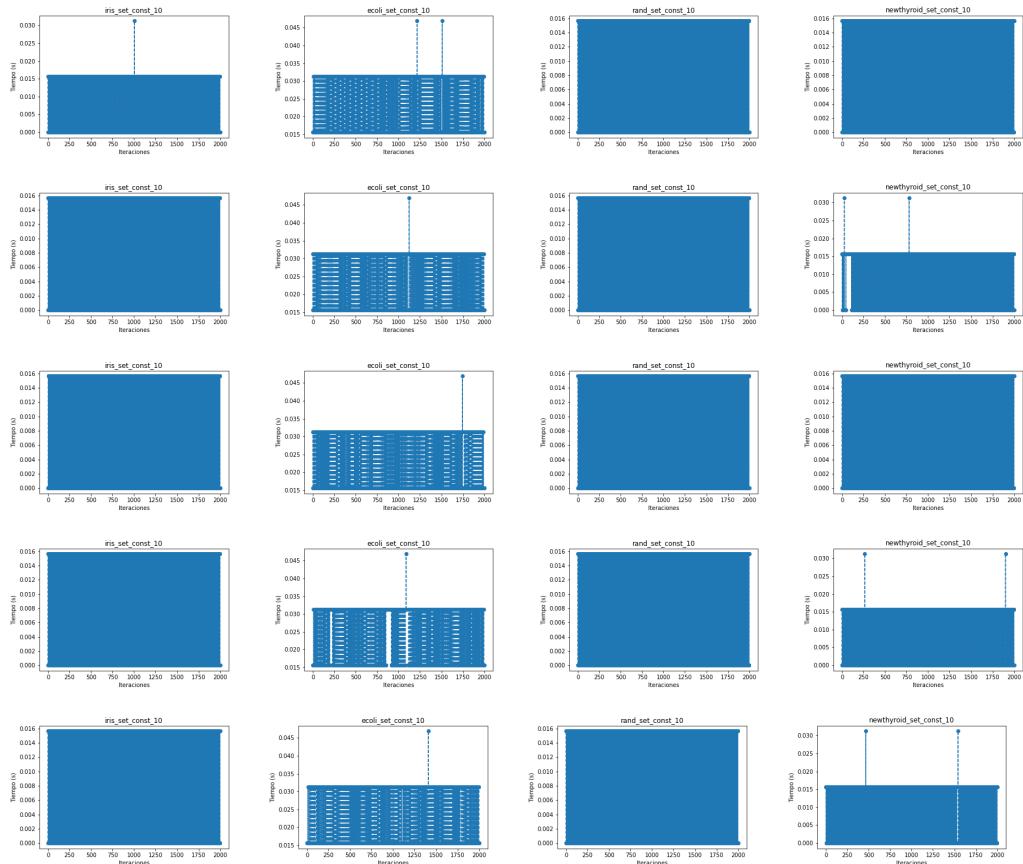


Figura 25: Tiempos de AGE-SF para 10 % de restricciones

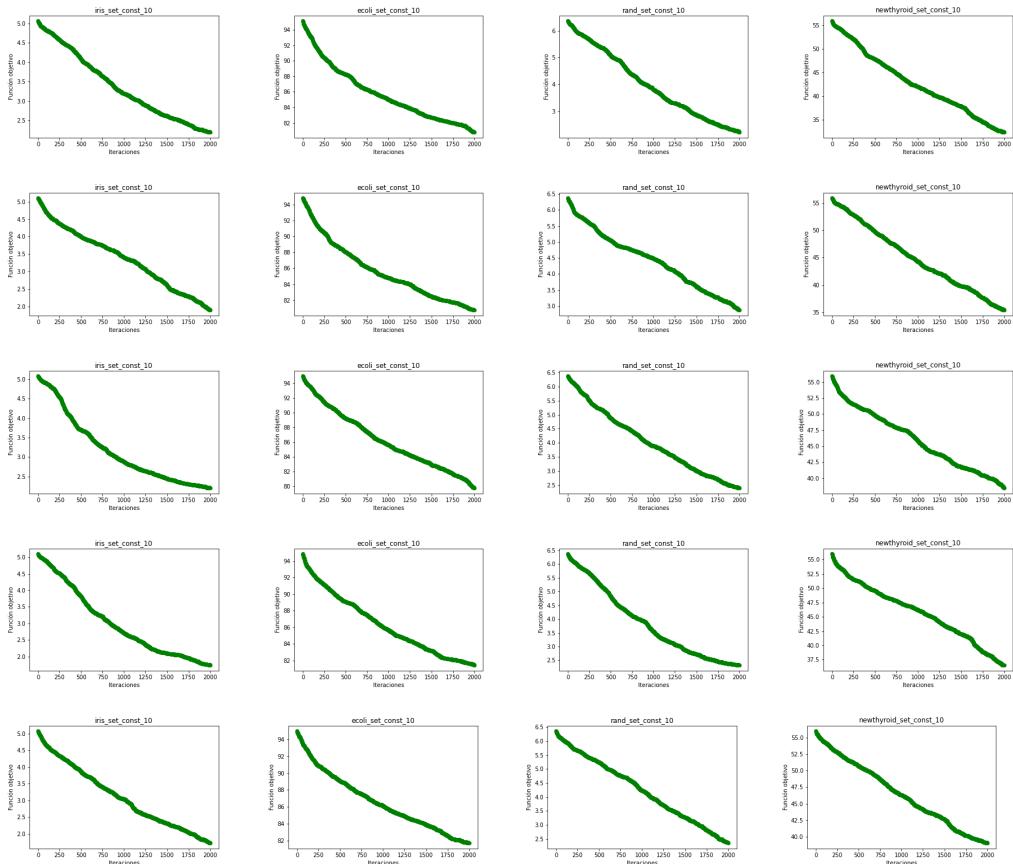


Figura 26: Agregado de AGE-SF para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.85	276.00	1.72	17.13	40.47	2926.00	79.73	56.97	1.10	222.00	1.90	17.41	16.09	690.00	28.71	27.41
Ejecución 2	0.88	169.00	1.41	17.23	40.39	2808.00	78.06	56.17	1.12	235.00	1.96	17.08	17.41	720.00	30.58	27.64
Ejecución 3	0.79	225.00	1.50	17.23	41.34	2840.00	79.44	57.25	0.96	126.00	1.41	17.11	13.69	1387.00	39.05	27.88
Ejecución 4	0.82	373.00	2.00	17.16	40.99	2843.00	79.13	57.06	0.91	127.00	1.37	17.19	16.05	747.00	29.71	27.64
Ejecución 5	0.84	282.00	1.73	17.77	40.44	2926.00	79.69	58.56	0.99	130.00	1.46	17.72	16.34	802.00	31.00	28.22
Media	0.83	265.00	1.67	17.30	40.73	2868.60	79.21	57.20	1.01	168.00	1.62	17.30	15.92	869.20	31.81	27.76

Cuadro 17: Resultados AGE-SF para 20 % de restricciones

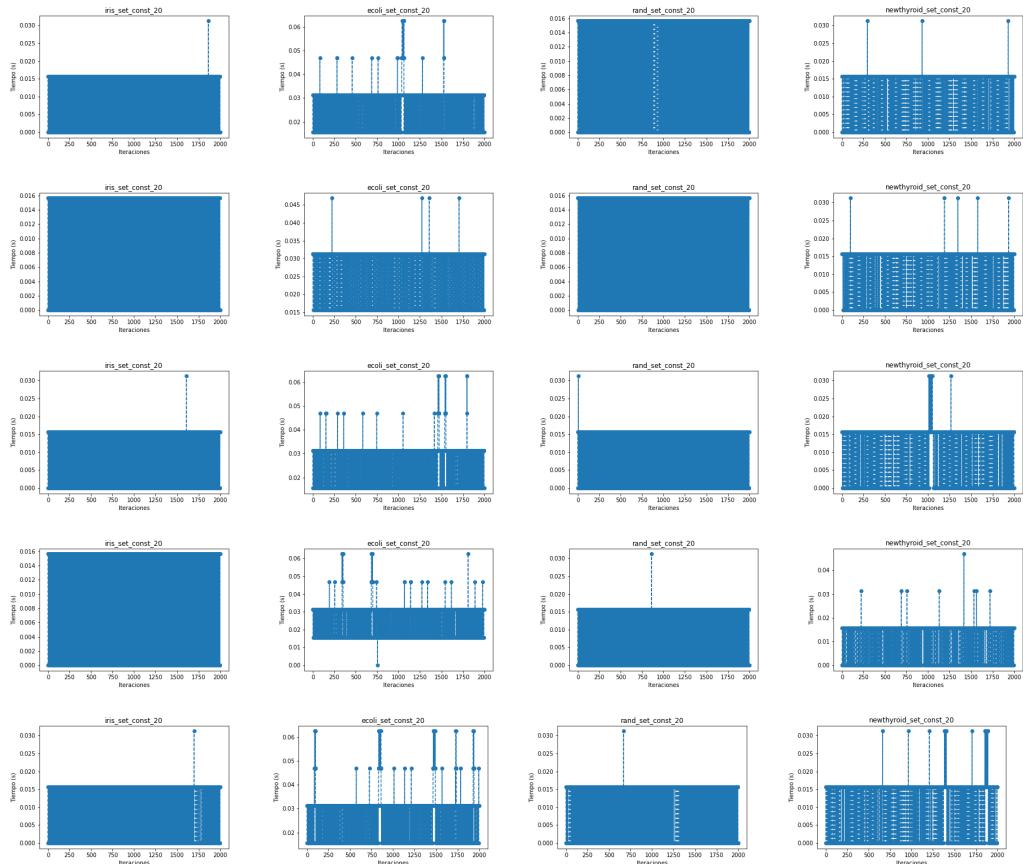


Figura 27: Tiempos de AGE-SF para 20 % de restricciones

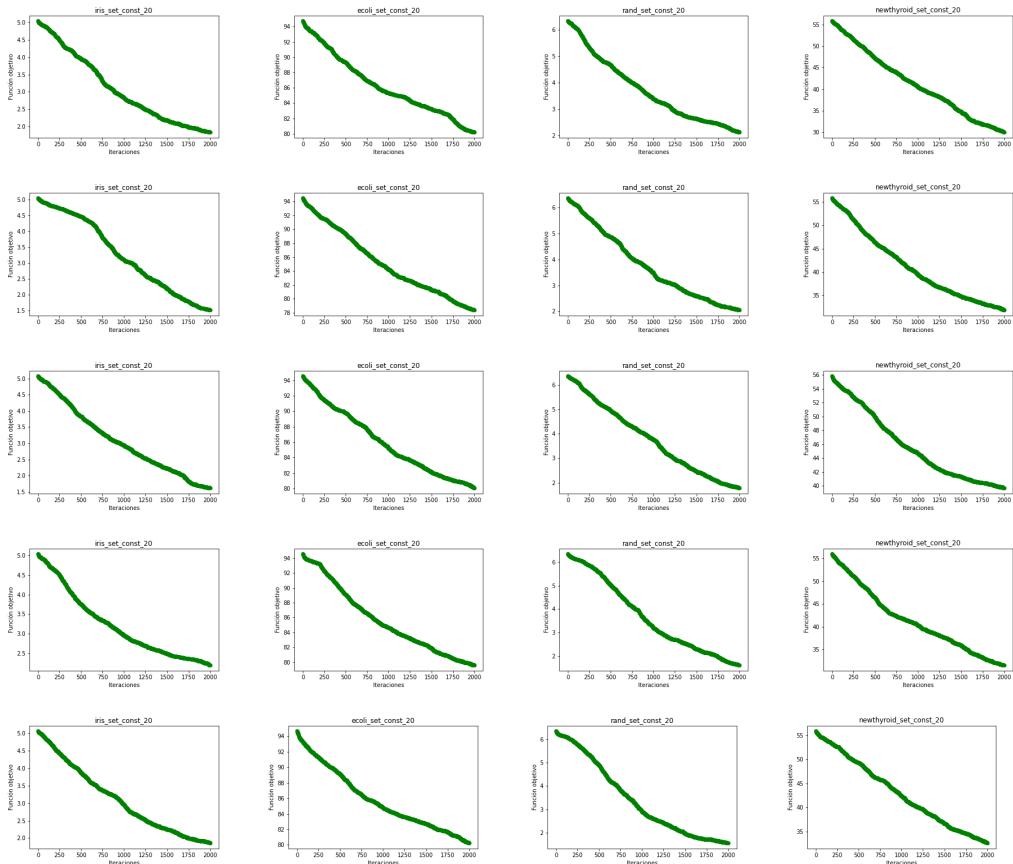


Figura 28: Agregado de AGE-SF para 20 % de restricciones

3.8. Algoritmo Memético (10, 1.0)

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.67	0.00	0.67	187.98	24.54	204.00	30.01	737.92	0.72	0.00	0.72	197.39	14.03	27.00	15.02	327.67
Ejecución 2	0.67	0.00	0.67	178.73	26.40	227.00	32.49	716.20	0.72	0.00	0.72	178.58	13.83	6.00	14.05	310.48
Ejecución 3	0.67	0.00	0.67	178.77	24.20	109.00	27.12	696.34	0.72	0.00	0.72	178.00	12.23	100.00	15.89	310.61
Ejecución 4	0.67	0.00	0.67	178.02	25.66	113.00	28.69	693.14	0.72	13.00	0.84	178.31	10.89	127.00	15.53	310.05
Ejecución 5	0.67	0.00	0.67	177.98	29.60	204.00	35.08	690.05	0.73	9.00	0.80	178.36	14.08	9.00	14.41	308.78
Media	0.67	0.00	0.67	180.30	26.08	171.40	30.68	706.73	0.72	4.40	0.76	182.13	13.01	53.80	14.98	313.52

Cuadro 18: Resultados AM-(10,1.0) para 10 % de restricciones

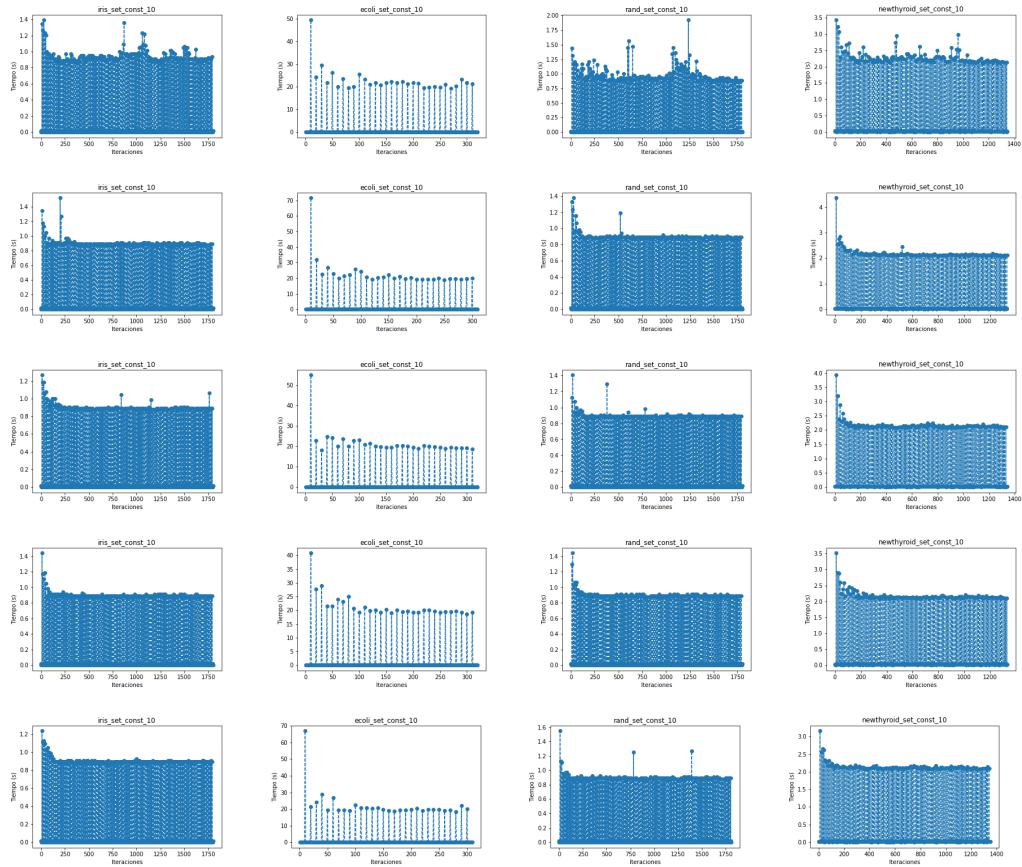


Figura 29: Tiempos de AM-(10,1.0) para 10 % de restricciones

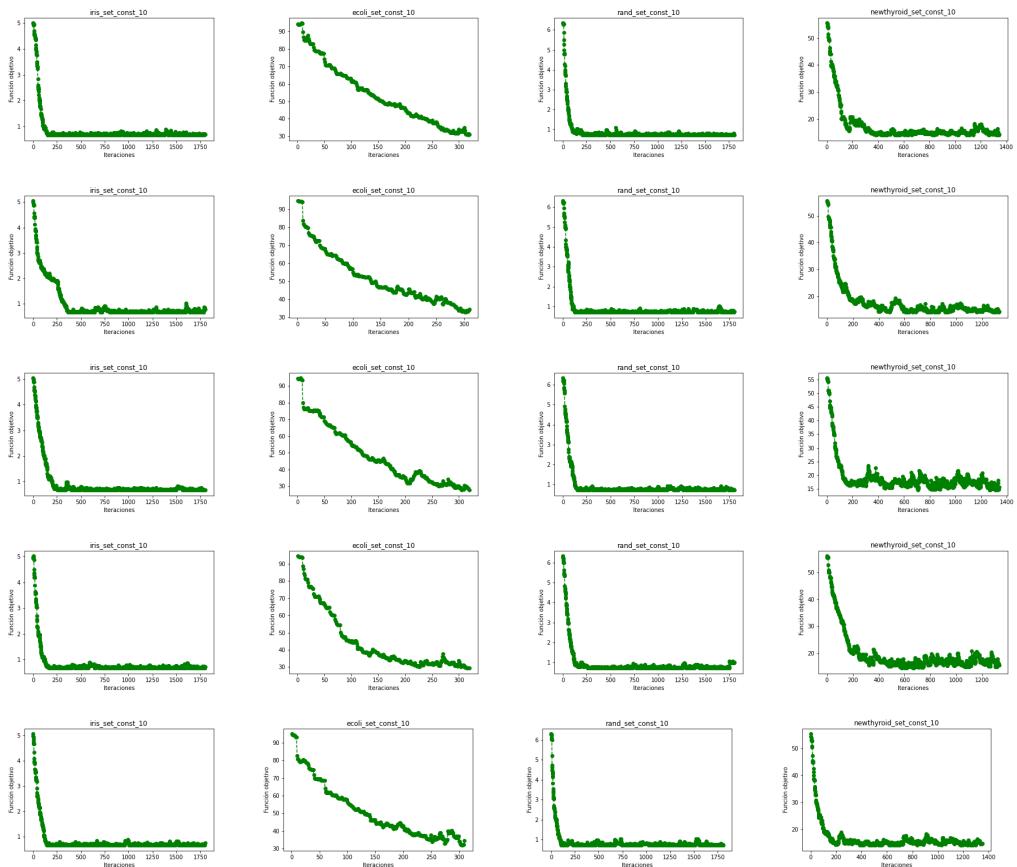


Figura 30: Agregado de AM-(10,1.0) para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.67	0.00	0.67	293.81	34.81	262.00	38.33	1229.59	0.72	0.00	0.72	273.83	13.83	25.00	14.29	509.20
Ejecución 2	0.67	0.00	0.67	272.44	27.50	428.00	33.24	1221.91	0.72	0.00	0.72	274.59	11.96	294.00	17.34	507.50
Ejecución 3	0.67	0.00	0.67	272.13	29.80	442.00	35.73	1210.27	0.72	0.00	0.72	273.27	14.29	0.00	14.29	510.03
Ejecución 4	0.67	0.00	0.67	271.89	27.28	315.00	31.51	1208.44	0.72	0.00	0.72	273.03	14.17	225.00	18.28	506.03
Ejecución 5	0.67	0.00	0.67	272.73	30.51	320.00	34.81	1214.70	0.72	0.00	0.72	272.75	10.68	338.00	16.86	510.45
Media	0.67	0.00	0.67	276.60	29.98	353.40	34.72	1216.98	0.72	0.00	0.72	273.49	12.99	176.40	16.21	508.64

Cuadro 19: Resultados AM-(10,1.0) para 20 % de restricciones

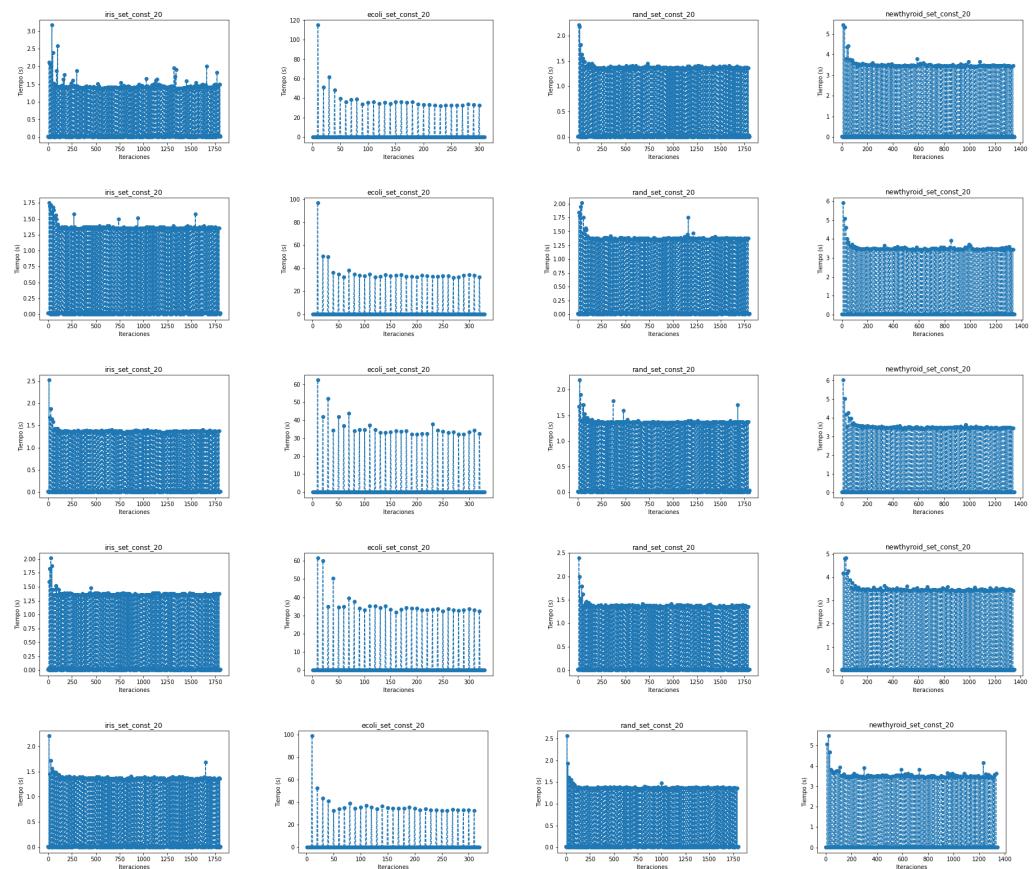


Figura 31: Tiempos de AM-(10,1.0) para 20 % de restricciones

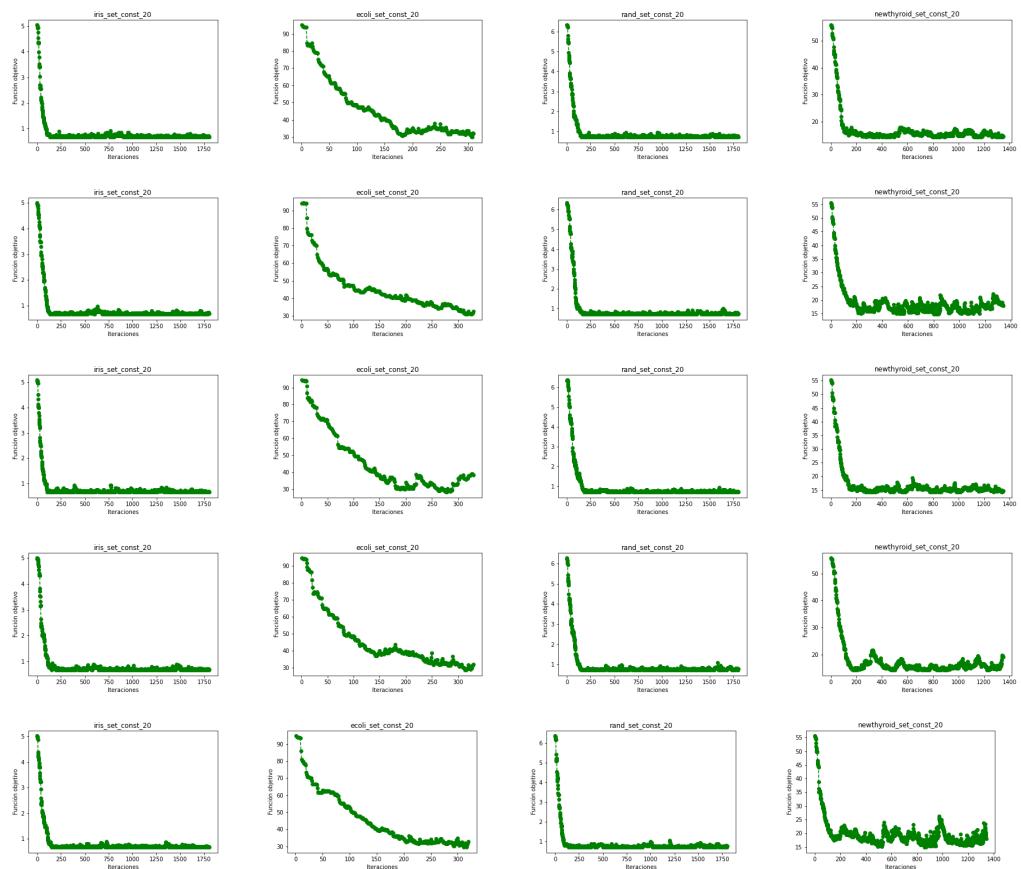


Figura 32: Agregado de AM-(10,1,0) para 20 % de restricciones

3.9. Algoritmo Memético (10, 0.1)

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.67	0.00	0.67	150.59	30.69	233.00	36.94	646.09	0.72	0.00	0.72	136.55	13.76	40.00	15.23	256.55
Ejecución 2	0.67	0.00	0.67	134.27	28.71	265.00	35.82	655.73	0.72	0.00	0.72	139.09	14.06	40.00	15.53	256.02
Ejecución 3	0.67	0.00	0.67	140.34	29.48	344.00	38.71	644.44	0.72	0.00	0.72	137.66	12.95	127.00	17.59	257.89
Ejecución 4	0.67	0.00	0.67	134.64	23.36	233.00	29.62	629.75	0.72	0.00	0.72	139.25	13.24	164.00	19.24	261.72
Ejecución 5	0.67	0.00	0.67	139.61	30.08	272.00	37.37	625.34	0.72	0.00	0.72	140.33	13.89	149.00	19.34	257.17
Media	0.67	0.00	0.67	139.89	28.46	269.40	35.69	640.27	0.72	0.00	0.72	138.58	13.58	104.00	17.38	257.87

Cuadro 20: Resultados AM-(10,0.1) para 10 % de restricciones

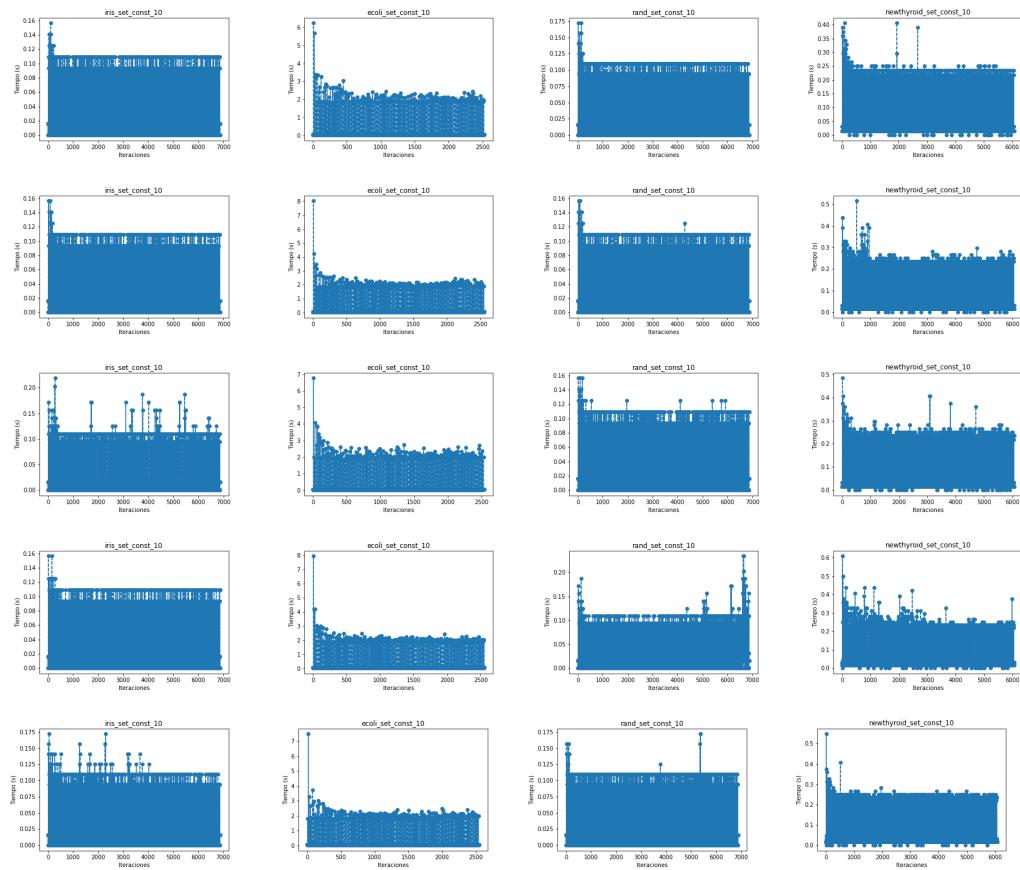


Figura 33: Tiempos de AM-(10,0.1) para 10 % de restricciones

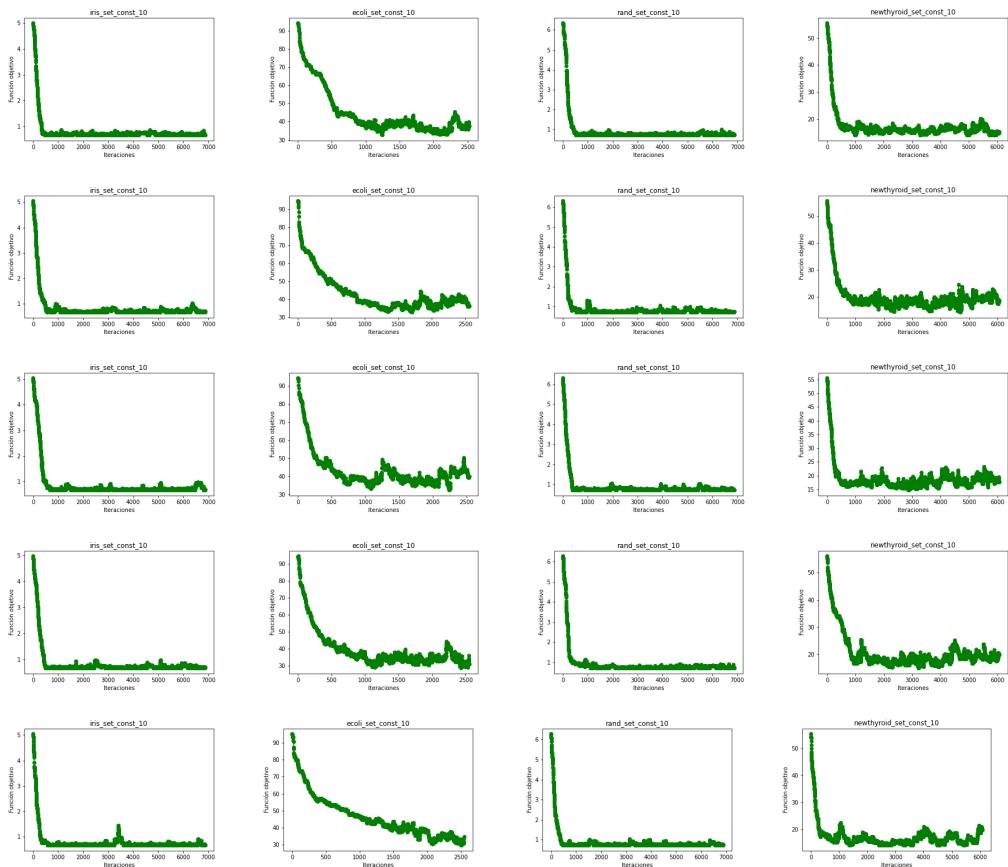


Figura 34: Agregado de AM-(10,0.1) para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.68	27.00	0.77	217.33	27.36	598.00	35.38	1097.81	0.72	0.00	0.72	210.25	12.87	379.00	19.80	407.67
Ejecución 2	0.67	0.00	0.67	213.45	32.81	725.00	42.53	1150.28	0.72	0.00	0.72	212.27	14.37	87.00	15.96	418.59
Ejecución 3	0.67	0.00	0.67	213.19	26.71	453.00	32.79	1099.86	0.72	0.00	0.72	205.31	10.59	369.00	17.33	405.94
Ejecución 4	0.67	0.00	0.67	203.11	27.98	706.00	37.45	1139.64	0.72	0.00	0.72	220.83	17.54	337.00	23.70	446.34
Ejecución 5	0.67	0.00	0.67	209.52	26.14	324.00	30.49	1121.00	0.72	0.00	0.72	213.06	14.24	100.00	16.07	419.89
Media	0.67	5.40	0.69	211.32	28.20	561.20	35.73	1121.72	0.72	0.00	0.72	212.34	13.92	254.40	18.57	419.69

Cuadro 21: Resultados AM-(10,0.1) para 20 % de restricciones

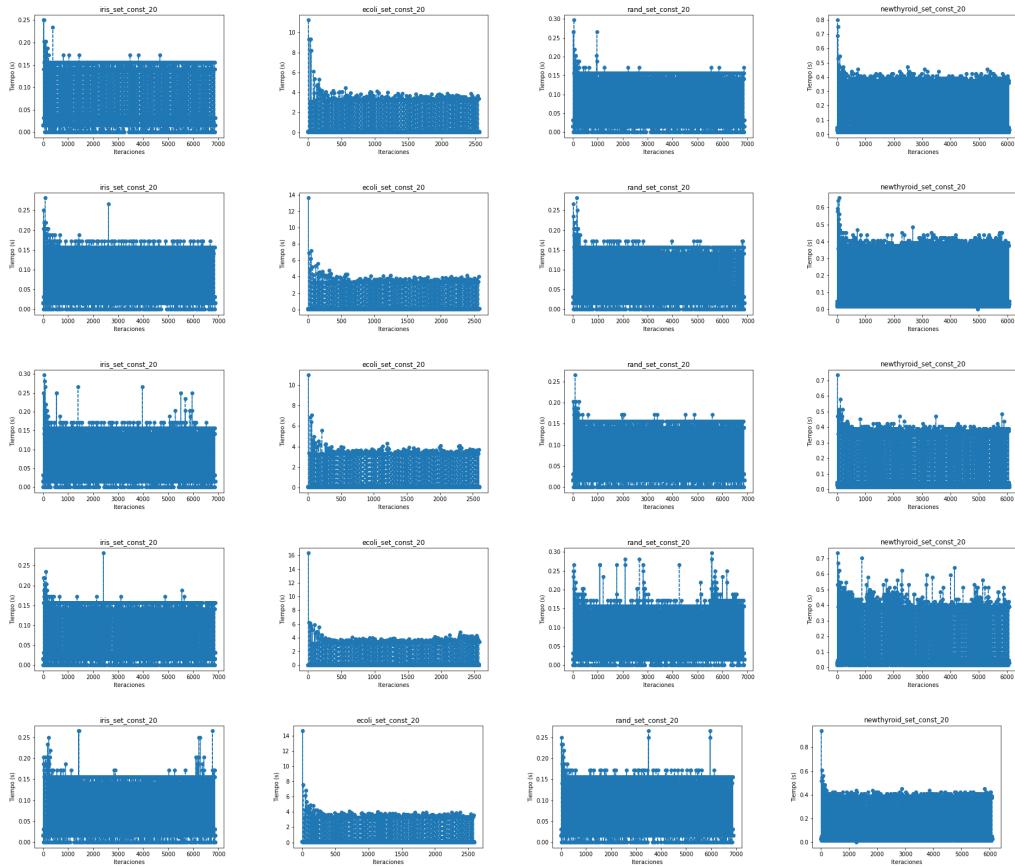


Figura 35: Tiempos de AM-(10,0.1) para 20 % de restricciones

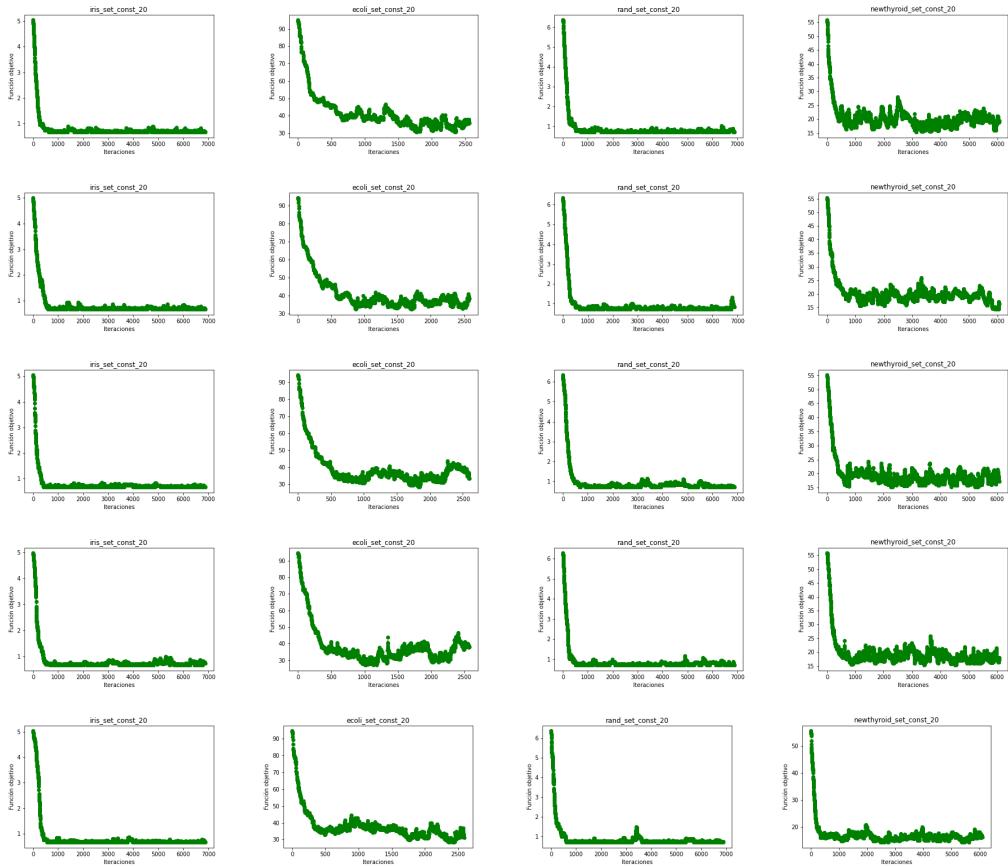


Figura 36: Agregado de AM-(10,0,1) para 20 % de restricciones

3.10. Algoritmo Memético (10, 0.1mej)

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.67	0.00	0.67	143.30	24.71	213.00	30.42	658.52	0.72	0.00	0.72	141.73	13.86	20.00	14.59	256.98
Ejecución 2	0.67	0.00	0.67	139.34	26.74	254.00	33.56	650.45	0.72	0.00	0.72	138.80	10.78	134.00	15.68	257.41
Ejecución 3	0.67	0.00	0.67	136.14	22.92	282.00	30.48	660.06	0.72	0.00	0.72	142.23	13.97	110.00	18.00	253.83
Ejecución 4	0.67	0.00	0.67	135.45	27.69	308.00	35.96	639.16	0.72	0.00	0.72	135.03	13.85	34.00	15.09	246.92
Ejecución 5	0.67	0.00	0.67	134.31	29.57	204.00	35.04	632.38	0.72	0.00	0.72	134.83	14.35	69.00	16.88	249.17
Media	0.67	0.00	0.67	137.71	26.33	252.20	33.09	648.11	0.72	0.00	0.72	138.53	13.36	73.40	16.05	252.86

Cuadro 22: Resultados AM-(10,0.1mej) para 10 % de restricciones

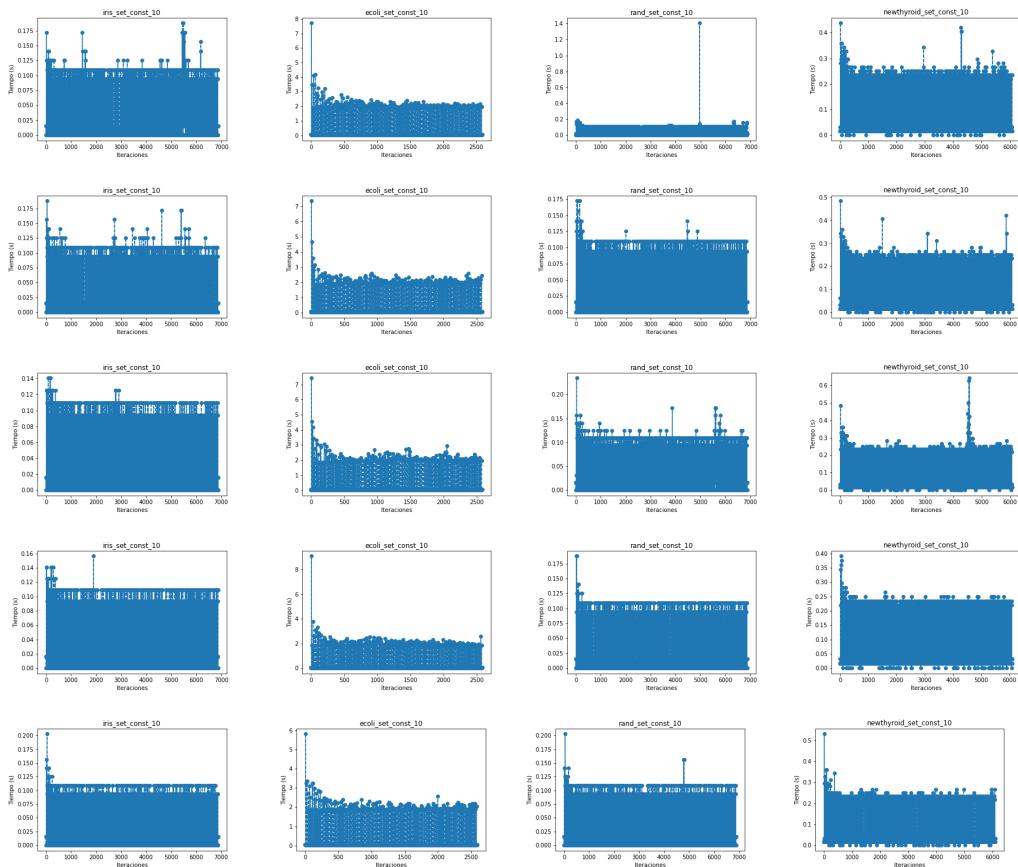


Figura 37: Tiempos de AM-(10,0.1mej) para 10 % de restricciones

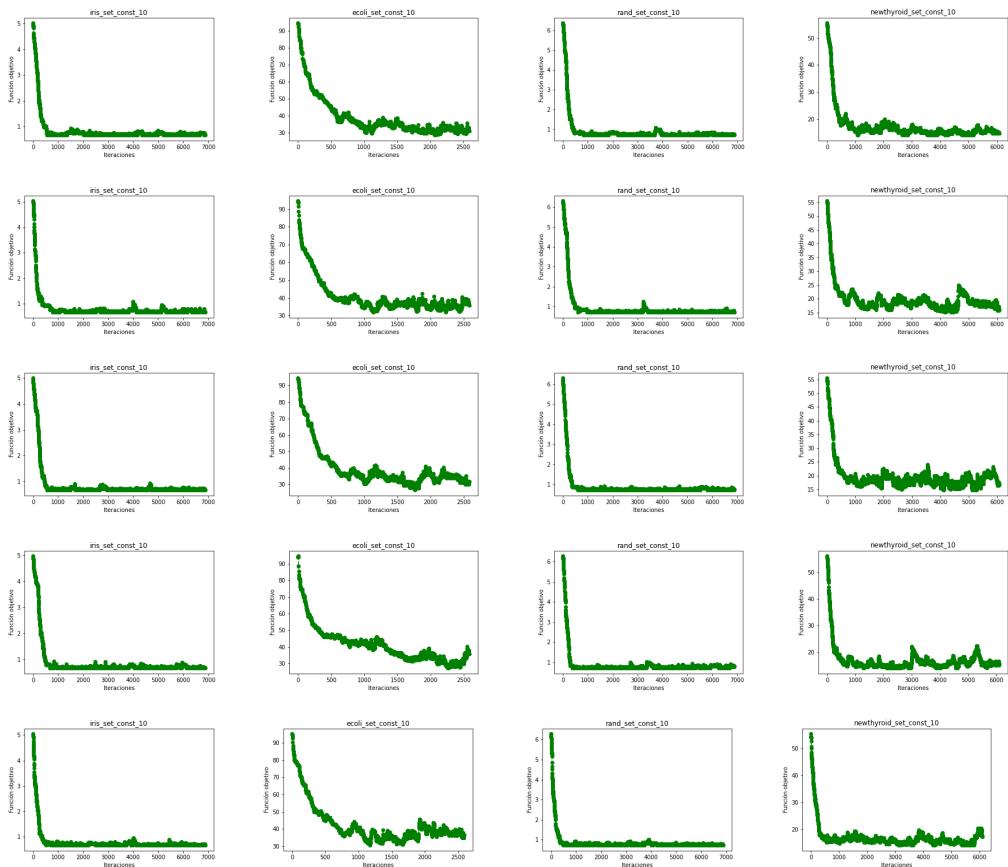


Figura 38: Agregado de AM-(10,0.1mej) para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.67	0.00	0.67	216.19	30.20	868.00	41.84	1146.77	0.72	0.00	0.72	211.19	14.29	0.00	14.29	439.02
Ejecución 2	0.67	0.00	0.67	215.16	25.54	505.00	32.31	1148.09	0.72	0.00	0.72	211.03	15.07	203.00	18.78	424.42
Ejecución 3	0.76	70.00	0.99	209.28	34.56	539.00	41.79	1100.38	0.72	0.00	0.72	206.98	14.16	86.00	15.74	406.59
Ejecución 4	0.67	0.00	0.67	205.77	26.41	585.00	34.25	1094.08	0.74	12.00	0.78	205.72	13.54	229.00	17.73	407.70
Ejecución 5	0.67	0.00	0.67	207.61	26.24	384.00	31.39	1137.09	0.72	0.00	0.72	220.77	14.15	86.00	15.72	416.34
Media	0.69	14.00	0.73	210.80	28.59	576.20	36.32	1125.28	0.72	2.40	0.73	211.14	14.24	120.80	16.45	418.82

Cuadro 23: Resultados AM-(10,0.1mej) para 20 % de restricciones

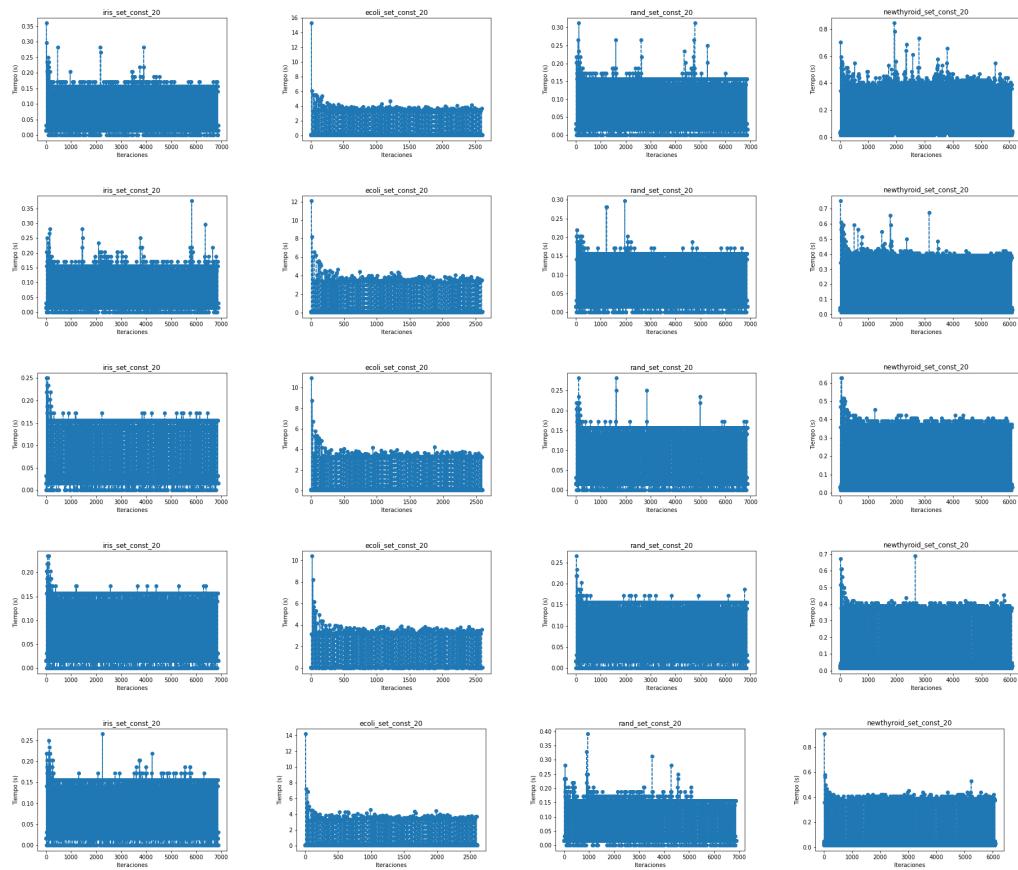


Figura 39: Tiempos de AM-(10,0.1mej) para 20 % de restricciones

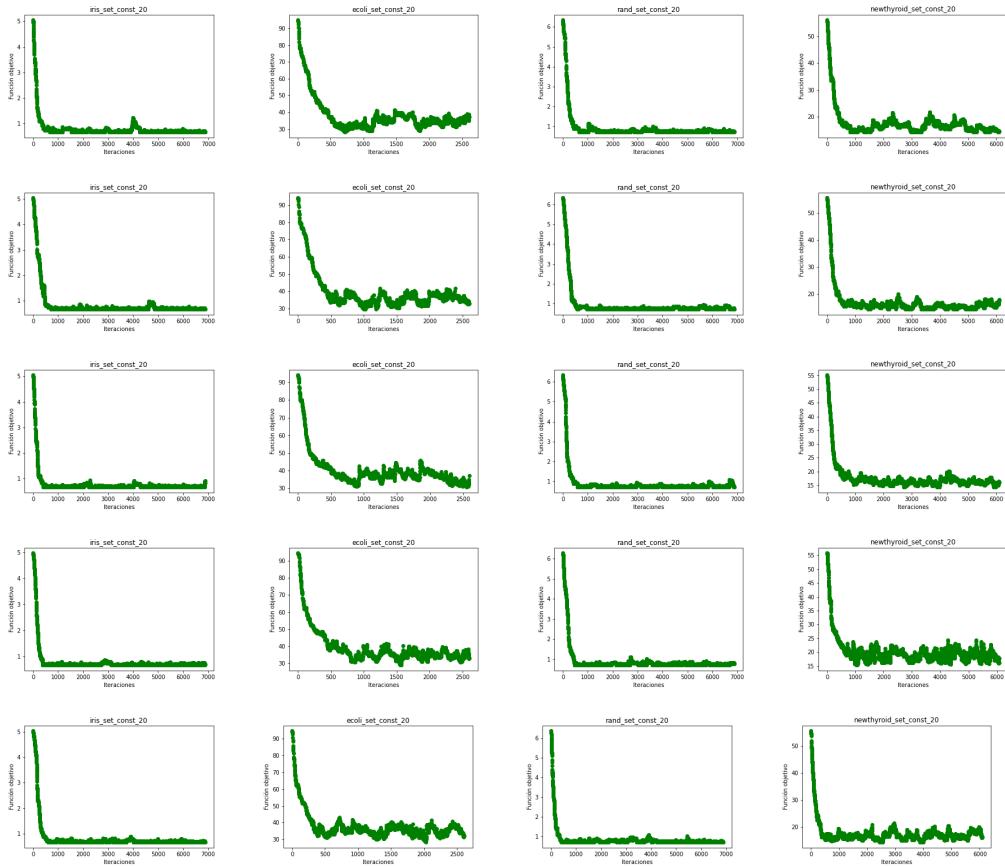


Figura 40: Agregado de AM-(10,0.1mej) para 20 % de restricciones

4. Conclusiones

A partir de las soluciones obtenidas, y con la ayuda de las gráficas del Anexo A, nos damos cuenta que los conjuntos de datos Iris y Rand son extremadamente fáciles. No solo no tienen apenas superposición (en el caso de Rand ninguna), sino que no existe desordenación de los datos y por tanto soluciones del estilo [0..1..2..] son de muy buena calidad. Esto resulta en que las técnicas greedy, las cuales ordenan los clústers posibles a la hora de elegir uno de ellos, encuentren óptimos en una o dos iteraciones (dependiendo de la ordenación que haga en ese caso).

Seguidamente se realiza un análisis de los resultados de cada algoritmo, realizando comparativas entre ellos.

4.1. Greedy COPKM v1

Por un lado vemos que COPKM es un algoritmo bastante rápido y con muy buenas soluciones, donde salvo casos excepcionales encuentra la misma solución para Iris y Rand independientemente de la semilla. En estos conjuntos de datos, puesto que el algoritmo no tiene en cuenta las distancias intra-clúster, resulta obvio que aunque el valor de infeasibility sea cero la calidad de la solución no tiene por qué ser óptima, aunque viendo las gráficas del Anexo A parece que no se aleja mucho.

Además, vemos que la cantidad de iteraciones que hace en los problemas es corta en comparación con los de BL, esto se debe a que en cada iteración reduce la infeasibility de más de un elemento, mientras que en BL solo se cambia un único valor.

Por el comportamiento del algoritmo greedy vemos que se basa en mejorar la posición de los centroides para así fijar con mejor calidad los elementos a sus clústers. Pero la forma de hacerlo acarrea una desventaja, y es que no considera soluciones previas calculadas anteriormente. Esto hace que al seguir el mismo orden de índices en cada iteración los valores de infeasibility varíen menos, puesto que hay que esperar a un elemento que tenga dos o más clústers posibles para poder actualizar en base al centroide.

4.2. Greedy COPKM v2

Añadiéndole la mejora al algoritmo greedy en COPKM v2, hacemos que se reutilice información de iteraciones anteriores, puesto que la infeasibility ya no se calcula únicamente con los elementos recién incluídos sino junto a los que ya estaban puestos (y a la espera de ser actualizados) en la iteración anterior.

Se podría decir que COPKM v2 actúa de manera similar a las mutaciones presentes en los algoritmos genéticos, pero sin permitir empeorar la infeasibility. Esto hace que la convergencia sea aún más rápida y, además, evitemos ciclos (puesto que si ordenamos los clústers posibles siempre cogerá los mismos en caso de empate).

Si nos fijamos en Ecoli, vemos que la calidad de las soluciones es mucho mejor, dónde especialmente supera a COPKM v1 es en tiempos de cómputo. El actualizar centroides y la solución en cada iteración facilita la decisión por una solución y la convergencia hacia un óptimo.

Al igual que en COPKM v1, vemos que los algoritmos greedy se comportan de mejor manera cuanto más aumenta el número de restricciones, esto es porque la selección de un clúster afecta más al valor de infeasibility.

4.3. Búsqueda Local

Observando los resultados de BL vemos que estos son bastante peores a los de los COPKMs. Esto se debe en teoría a dos motivos, la inicialización y al método de búsqueda en el espacio de soluciones.

Respecto al primero, BL busca converger hacia un mínimo a partir de una solución inicial. Esto hace que la selección de esta inicialización sea relevante en la calidad de la solución final, puesto que resulta muy fácil converger hacia mínimos locales. Algoritmos mixtos que sean capaces de empeorar su solución en ciertos momentos pueden ayudar a salir de estos focos locales y dirigirse hacia mejores resultados. Puesto que en el algoritmo COPKM v1 no consideramos soluciones previas, resulta claro que esto pase y pueda encontrar soluciones de mejor calidad.

Por otro lado, la exploración del espacio en BL tiene dos detalles. La selección de primero el mejor reduce el tiempo de cómputo pero también puede ocasionar el problema mencionado en el párrafo anterior. Un posible vecino puede desviar el resultado a un mínimo local de peor calidad.

Además, sabemos que la búsqueda local no se centra únicamente en la mejora de infeasibility, sino también en crear una solución con unos clústers bien diferenciados. Esto vienen dado por el valor λ , que en nuestro caso se ha inicializado a un valor pequeño. Las consecuencias de esta inicialización se aprecia en los valores de tasaC e infeasibility. Mientras que los greedy bajan mucho la infeasibility BL ataca en la tasaC para reducir el agregado (en greedy los valores de tasaC siguen siendo bajos en comparación con BL, pero esto es una consecuencia indirecta de los conjuntos de datos).

Adicionalmente, aunque en el peor de los casos todos los algoritmos exploran el mismo tamaño de vecindario, los greedy ganan en velocidad en gran medida. Esto está causado por el tiempo perdido en la generación y evaluación de un nuevo vecino, además de la necesidad de comprobar las distancias para cada uno de ellos.

Puesto que en BL el vecindario son todos los posibles cambios, a medida que mejora la solución debemos explorarlo más para encontrar uno que mejore (se aprecia en las gráficas de tiempo, donde para los greedy son más inestables pero crecen menos). Mientras exista, el tiempo entre iteraciones aumenta drásticamente para BL. Pero puesto que tratamos con números reales no podemos comparar directamente con $f'(s) < f(s)$. Es necesario que este $<$ lo sea por un margen concreto. Este margen también es influyente en la solución y, aunque por la definición del algoritmo debería ser lo más bajo posible, un valor más pequeño hace que sea más sencillo saltar hacia un vecino y alarga el tiempo de cómputo.

Como se ha mencionado anteriormente, aunque por la forma de funcionar BL se piensa que es muy dependiente de la solución inicial, pues solo modificamos un elemento de la solución en cada iteración y no elegimos empeoramientos en ningún caso, en Ecoli no se da esta situación. Los valores de agregado acaban resultando muy similares independientemente de la semilla que se utilice. Esto se debe probablemente al pequeño número de evaluaciones en Ecoli. De esto último nos damos cuenta viendo las gráficas del valor de agregado por iteración, que nos indica que BL agota el número de evaluaciones permitido cuando aún podría seguir convergiendo.

Para Iris y Rand si se aprecia diferencia, más en el valor de infeasibility que en el de la tasaC.

4.4. Algoritmos Genéticos

Vemos que la calidad de las soluciones encontradas en media solo es comparable con las versiones COPKM, siendo AGG-UN la única que mejora a COPKM v2. Aún así, en el dataset más exigente (Ecoli), AGG-SF también supera a COPKM v2, debido a que la mejora de infeasibility no es tan relevante (principio por el que se regía el algoritmo greedy) como una buena separación en clústers. Se nota como la selección del valor de λ es muy relevante para el funcionamiento de los algoritmos.

Las variantes elitistas, pese a que pierden bastante en la calidad de los resultados, ganan mucho en velocidad, debido al mecanismo evolutivo sobre dos únicos padres. Observando las gráficas también se aprecia que en la mayoría de los casos los algoritmos AGG convergen bastante rápido, y solo en Ecoli es cuando se quedan cortos de evaluaciones.

La simpleza de los datasets Iris y Rand se sigue apreciando en estos casos. El nuevo conjunto de datos (Newthyroid) tampoco genera muchas complicaciones a los algoritmos, pero se producen distintas soluciones y merece la pena compararlos en base a él. Este dataset, con el cuál se obtienen muy buenos resultados con COPKM v2, en este caso está más disputado con los algoritmos genéticos.

Las variantes elitistas tienen la esperanza de que la población mejore como conjunto en cada iteración. Las estacionarias solo van modificando los peores cromosomas, y esto hace que la calidad media de la población casi nunca decremente. Por una parte está bien porque evita fluctuaciones en el valor medio de la función objetivo pero se queda bajo la suposición de que a partir de un cromosoma malo no puede producirse uno de muy buena calidad. Por otro lado estos dos últimos cromosomas en AGE pueden ser mejores que los nuevos, haciendo que empeore la calidad del conjunto. Aunque se puede suponer que la probabilidad de que esto ocurra repetidamente es baja.

En general vemos que los AGE mantienen malas poblaciones durante más iteraciones, pues se debe iterar una mayor cantidad de veces para ir sacando los malos cromosomas de la población.

Por otra parte, la diferencia entre aplicar un cruce uniforme y otro por segmento fijo es notoria. Independientemente de la variante genética se aprecia una mejora importante del cruce uniforme frente al del segmento fijo, aunque cuando tratamos con el 20 % de restricciones esta diferencia se reduce.

Por el funcionamiento de ambos operadores no da la sensación de que uno sea definitivamente mejor que otro, parece que está más relacionado con el tipo de problema con el que estamos tratando. Al estar particionando en clústers con restricciones existe mucha dependencia entre los elementos de la solución, pero no entre posiciones ordenadas sino “aleatorias” (un elemento puede tener restricciones con otro en cualquier posición, no necesariamente con los anteriores/posteriores). Esto aún así no aclara por qué existe tanta diferencia entre ambos operadores, puesto que bajo esa suposición el coger índices aleatorios de los cromosomas debería poder empeorar/mejorar lo mismo que seleccionando un segmento fijo de ellos.

También se ve que los algoritmos de AGE tienen unos valores de infeasibility descomunales. No se encuentra ningún motivo claro para esto, aunque es posible que se deba a que la selección de dos padres, junto al bajo peso que tiene la infeasibility en la función objetivo, haga que se enfoque repetidamente en separar los elementos y en el número de evaluaciones establecido no de tiempo a converger (viendo las gráficas se nota que más evaluaciones deberían mejorar los resultados).

4.5. Algoritmos Meméticos

Sabemos que los algoritmos evolutivos son buenos exploradores, pues los operadores de cruce permiten moverse fácilmente en el espacio de búsqueda. También sabemos que son malos explotadores, puesto que la mutación ocurre con baja probabilidad y realiza cambios muy pequeños en la población. Es por esto por lo que una hibridación con técnicas de búsqueda local pueden mejorar la calidad de los resultados, ya que estas son muy buenas a la hora de explotar una solución.

Aquí la clave reside en conseguir un buen balance entre evolucionar y optimizar la población, puesto que no queremos acabar en mínimos locales de baja calidad, pero tampoco queremos estar continuamente haciendo cambios drásticos en el espacio de búsqueda. Esto se regula con la frecuencia a la que se aplica el algoritmo de BL y modificando el conjunto de la población sobre el que se optimiza.

Trabajar con la población entera implica mejorarla como conjunto, y esto tiene consecuencias mencionadas en el apartado anterior (la suposición de que a partir de una solución mala se pueden generar buenas). Coger un 10 % de la población indica que no desperdiciamos muchas evaluaciones en la optimización, y fomentamos más la exploración. Por otro lado, la selección de los mejores cromosomas pretende mejorar el elitismo y converger más rápido, puesto que estos cromosomas se acercan más rápidamente a sus mínimos.

A partir de los resultados nos damos cuenta de dos detalles:

- Los algoritmos meméticos funcionan en general mejor que los genéticos, pero no consiguen superar al mejor de ellos (AGG-UN). Esto se puede deber a que los hiperparámetros elegidos para los algoritmos meméticos (población de 10 y probabilidades de cruce y mutación 0,7 y 0,001) no sean los óptimos, y exista un desbalanceo entre exploración y optimización.
- Por otro lado, vemos que optimizar la población al completo acaba obteniendo mejores resultados que aplicándola sobre una parte de ella. Esto se contrapone con la idea expuesta anteriormente de que malos elementos en la población pueden ayudar a generar soluciones de mejor calidad.

Una vez más quizás se deba a los hiperparámetros de los algoritmos, una población de 10 cromosomas es pequeña y un mal elemento influye en mayor medida. También es posible que esté influenciado por los conjuntos de datos. Ecoli y Newthyroid, los que mejor representan la calidad del algoritmo, nos muestran en la mayoría de casos de que, aunque peores, los valores de agregado son similares.

A. Representación visual de primeras dimensiones

A.1. Greedy COPKM

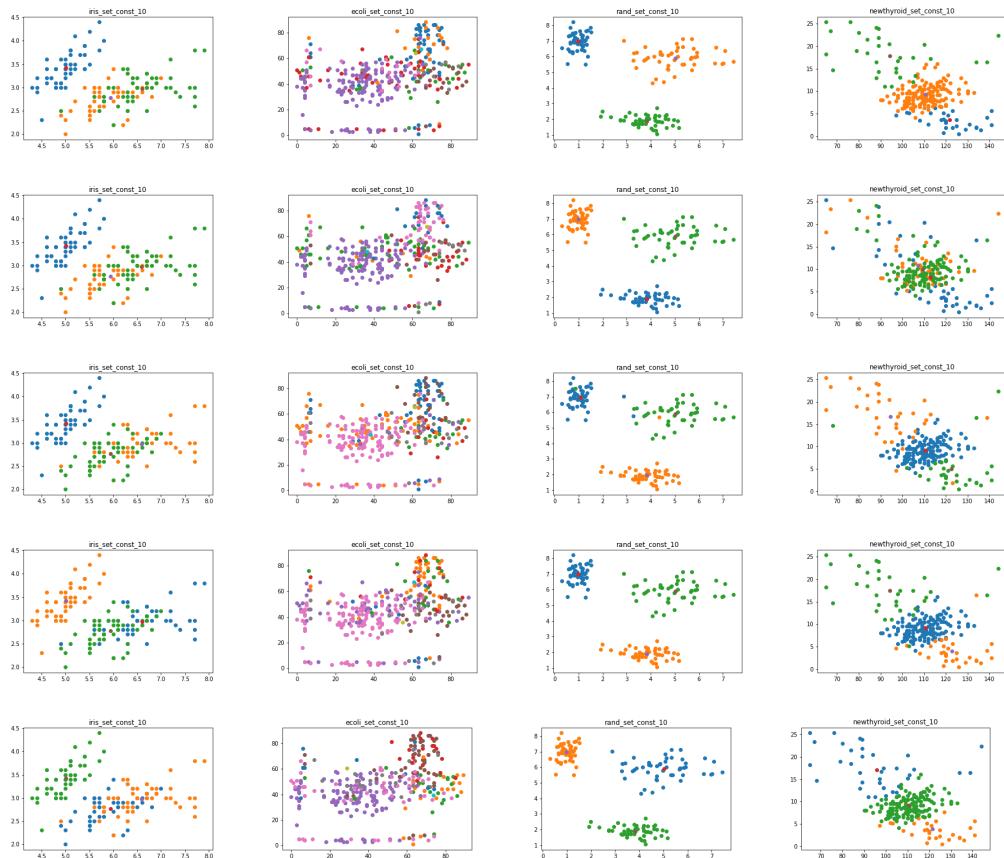


Figura 41: Primeras dimensiones de COPKM para 10 % de restricciones

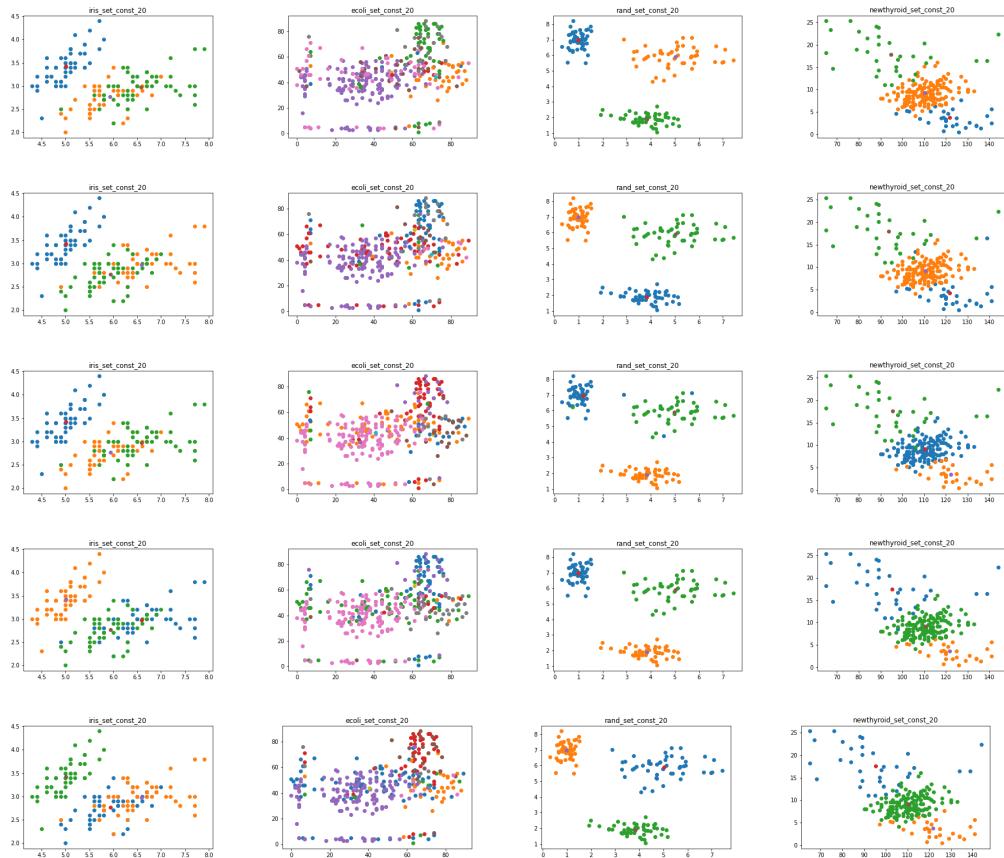


Figura 42: Primeras dimensiones de COPKM para 20 % de restricciones

A.2. Búsqueda Local

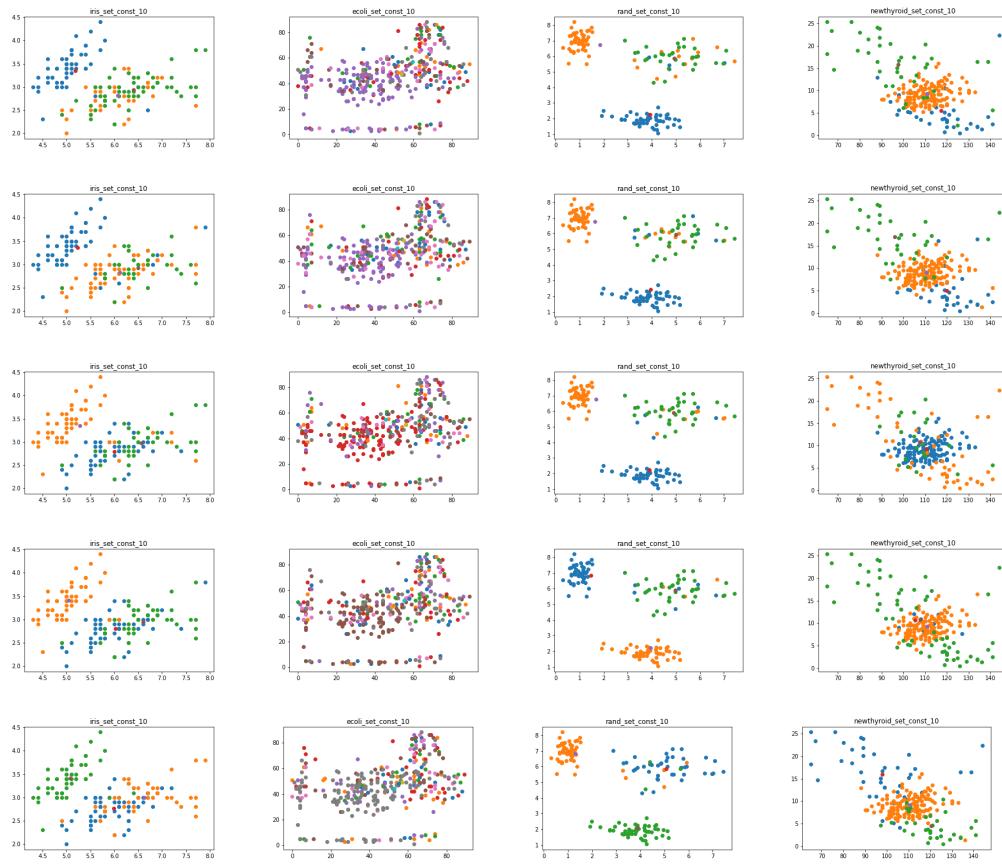


Figura 43: Primeras dimensiones de BL para 10 % de restricciones

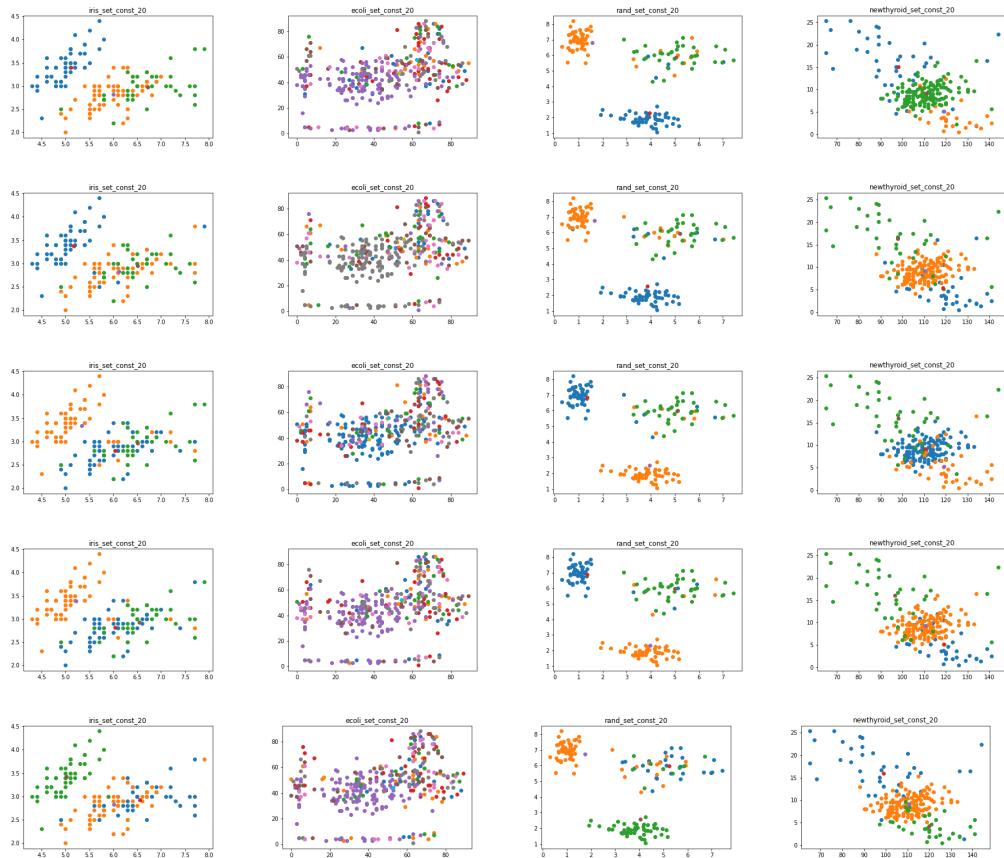


Figura 44: Primeras dimensiones de BL para 20 % de restricciones

A.3. Greedy COPKM v2

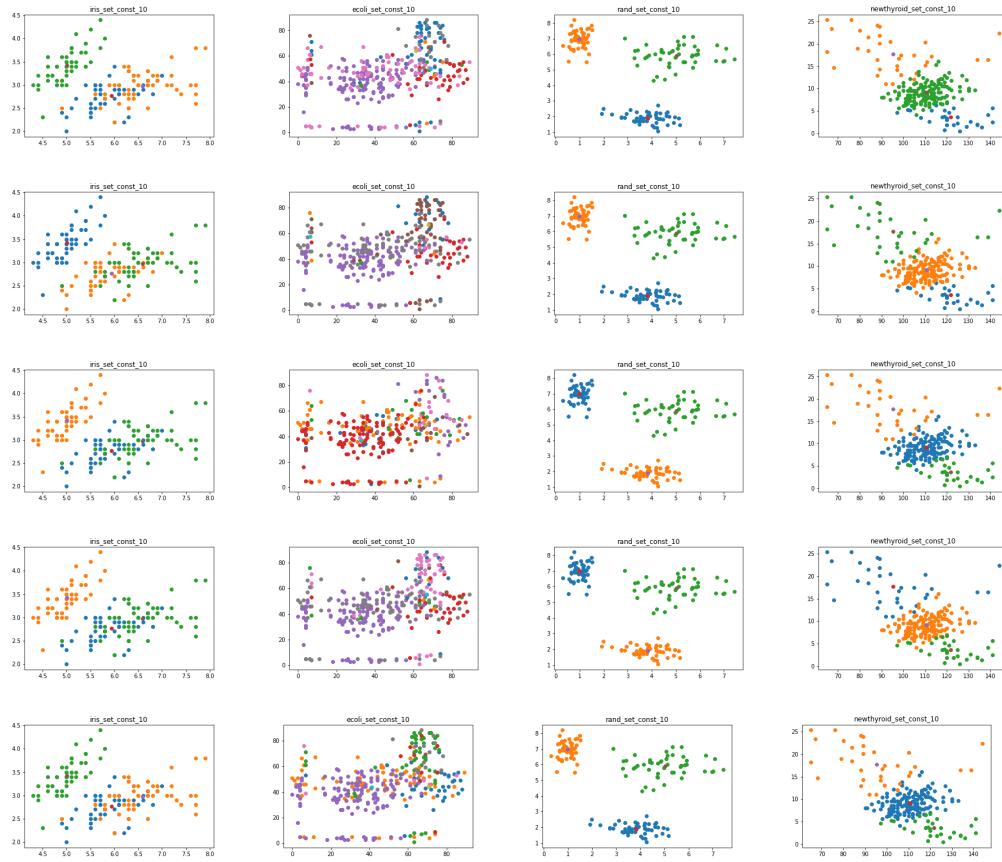


Figura 45: Primeras dimensiones de COPKM v2 para 10 % de restricciones

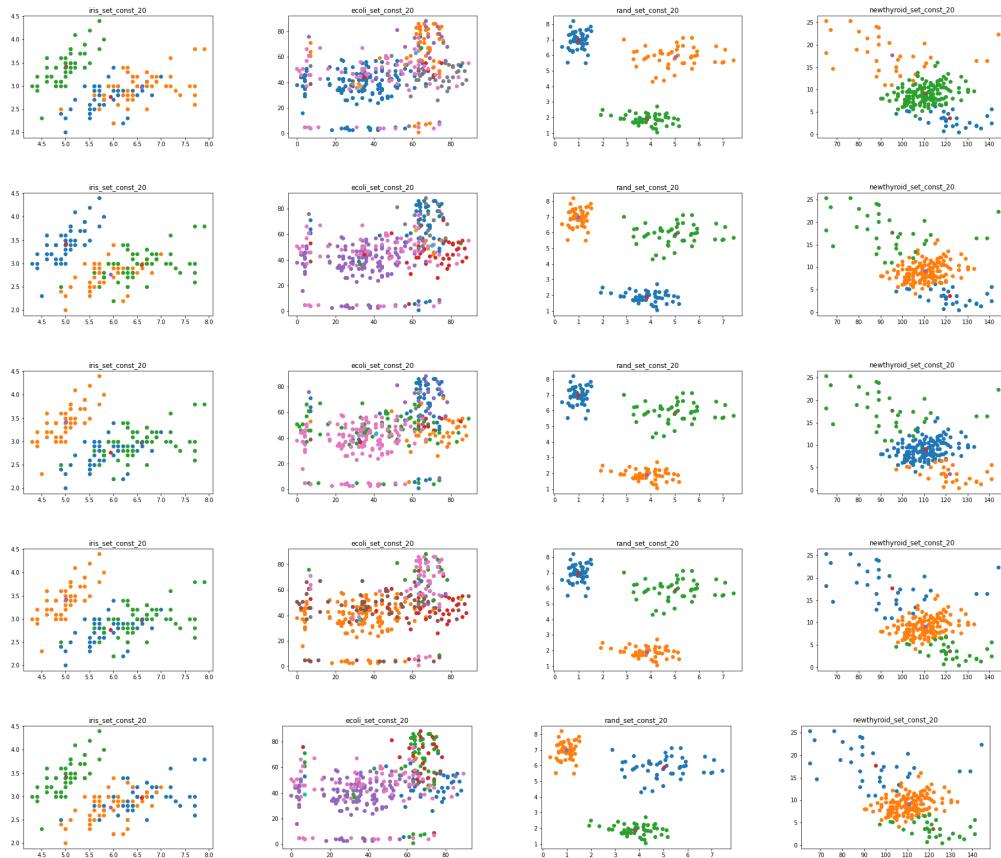


Figura 46: Primeras dimensiones de COPKM v2 para 20 % de restricciones

A.4. Algoritmo Generacional Elitista Uniforme

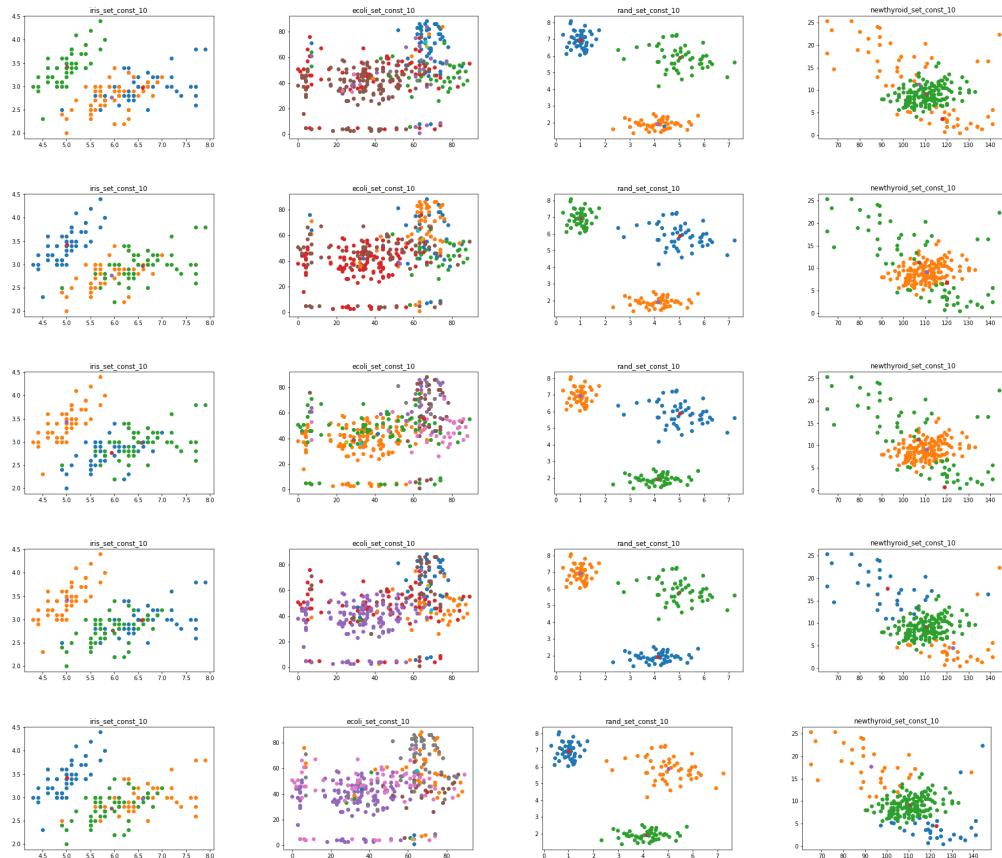


Figura 47: Primeras dimensiones de COPKM para 10 % de restricciones

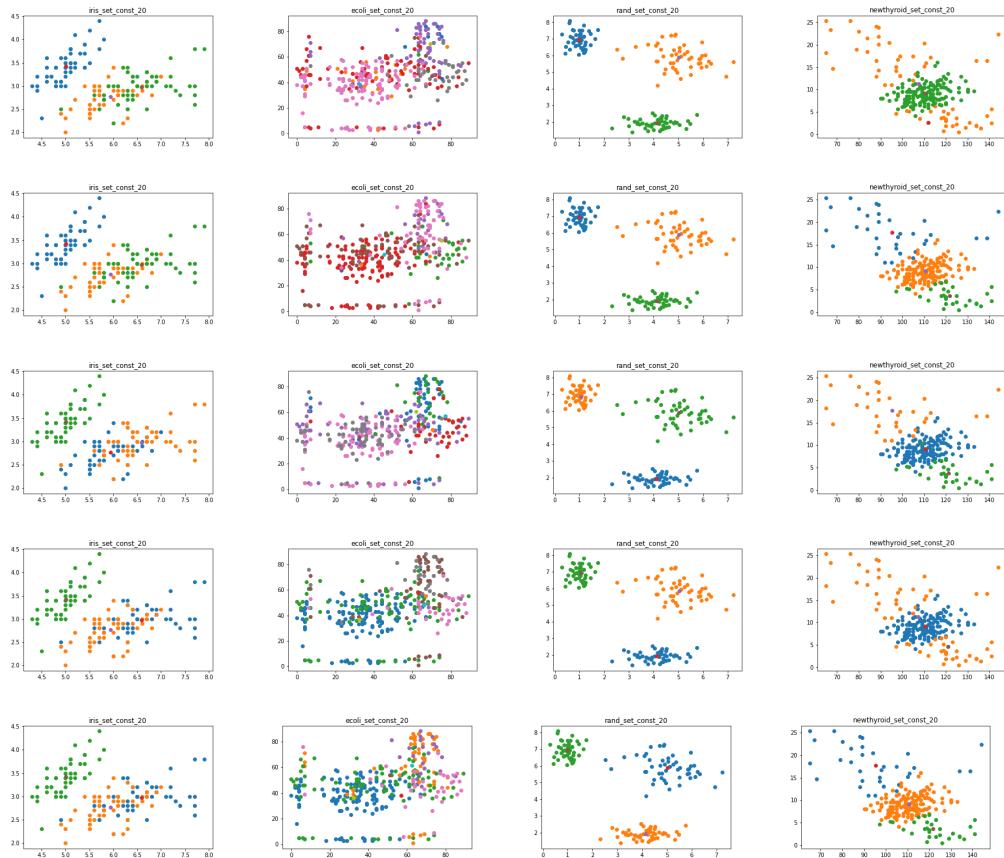


Figura 48: Primeras dimensiones de COPKM para 20 % de restricciones

A.5. Algoritmo Generacional Elitista por Segmento Fijo

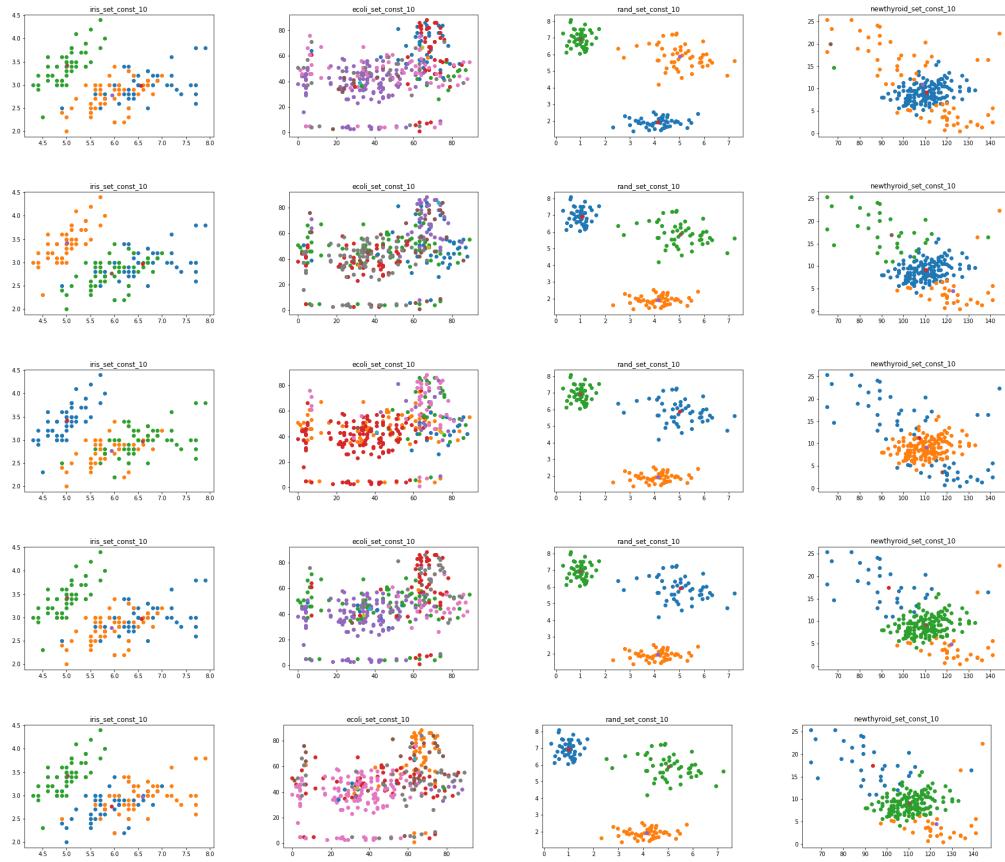


Figura 49: Primeras dimensiones de BL para 10 % de restricciones

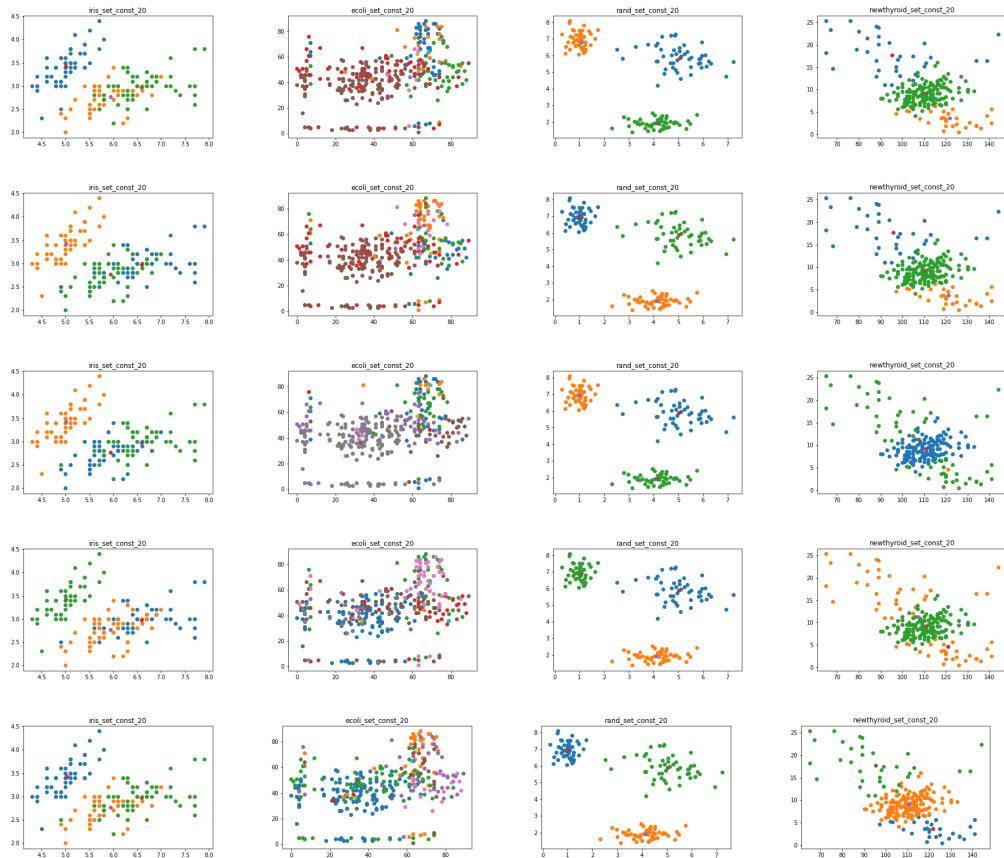


Figura 50: Primeras dimensiones de BL para 20 % de restricciones

A.6. Algoritmo Generacional Estacionario Uniforme

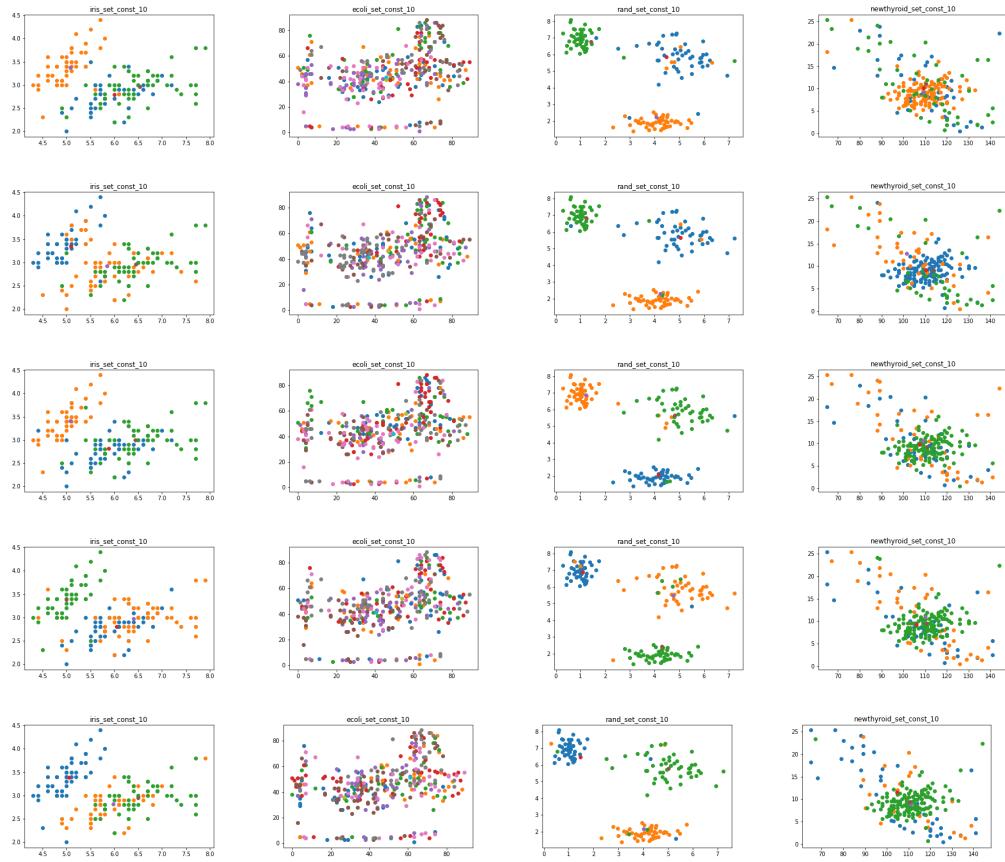


Figura 51: Primeras dimensiones de COPKM v2 para 10 % de restricciones

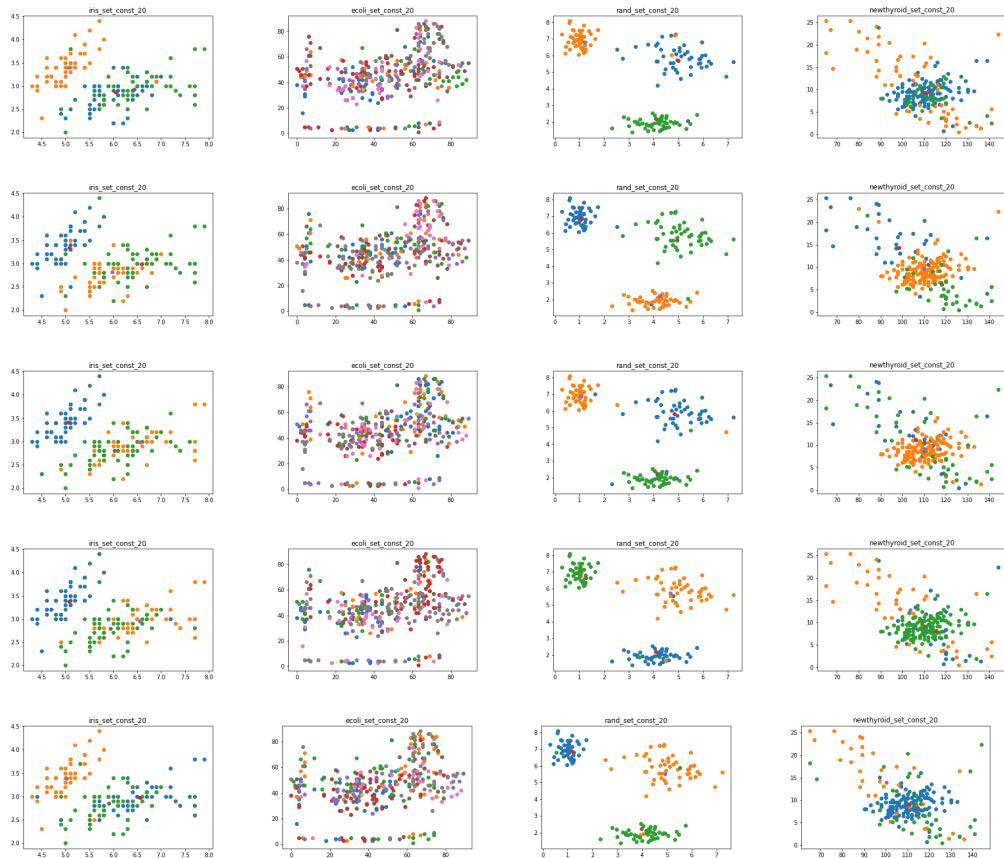


Figura 52: Primeras dimensiones de COPKM v2 para 20 % de restricciones

A.7. Algoritmo Generacional Estacionario por Segmento Fijo

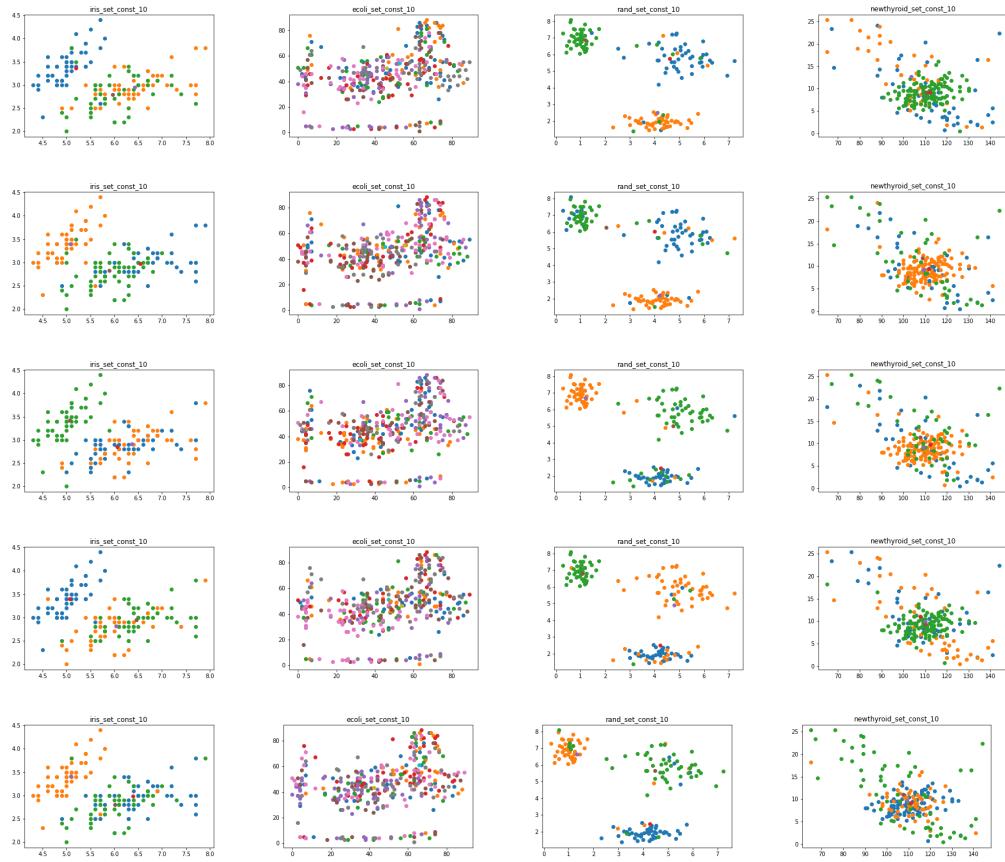


Figura 53: Primeras dimensiones de COPKM v2 para 10 % de restricciones

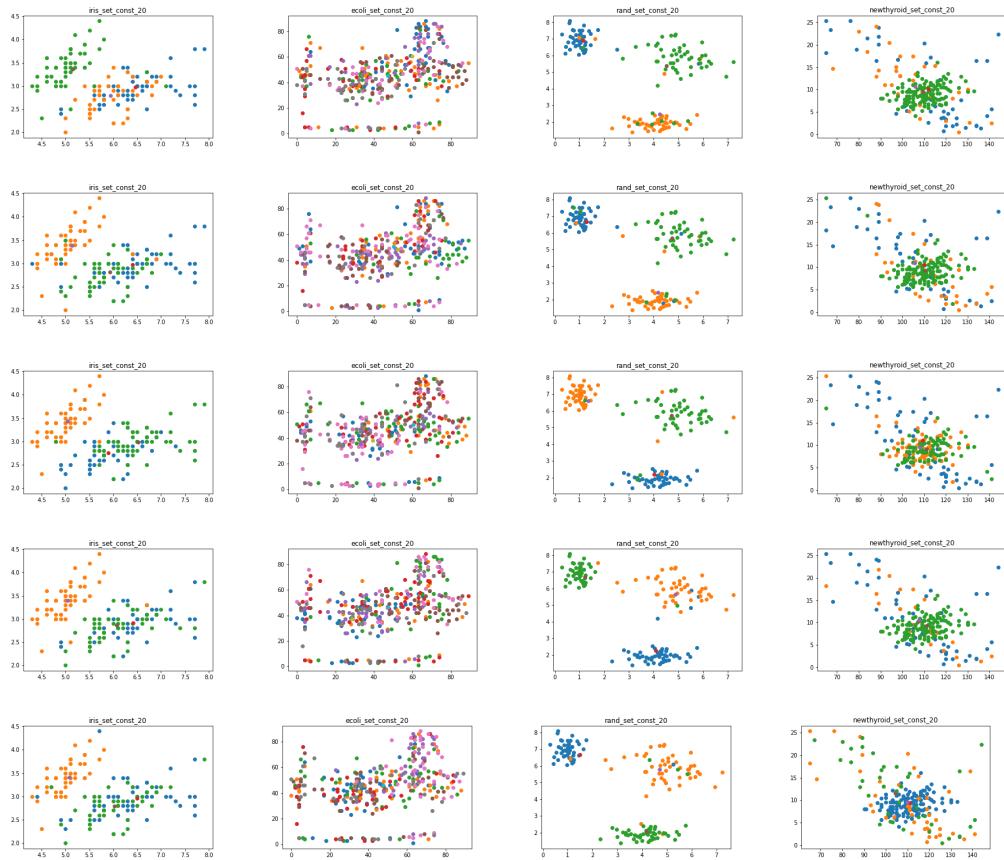


Figura 54: Primeras dimensiones de COPKM v2 para 20 % de restricciones

A.8. Algoritmo Memético (10, 1.0)

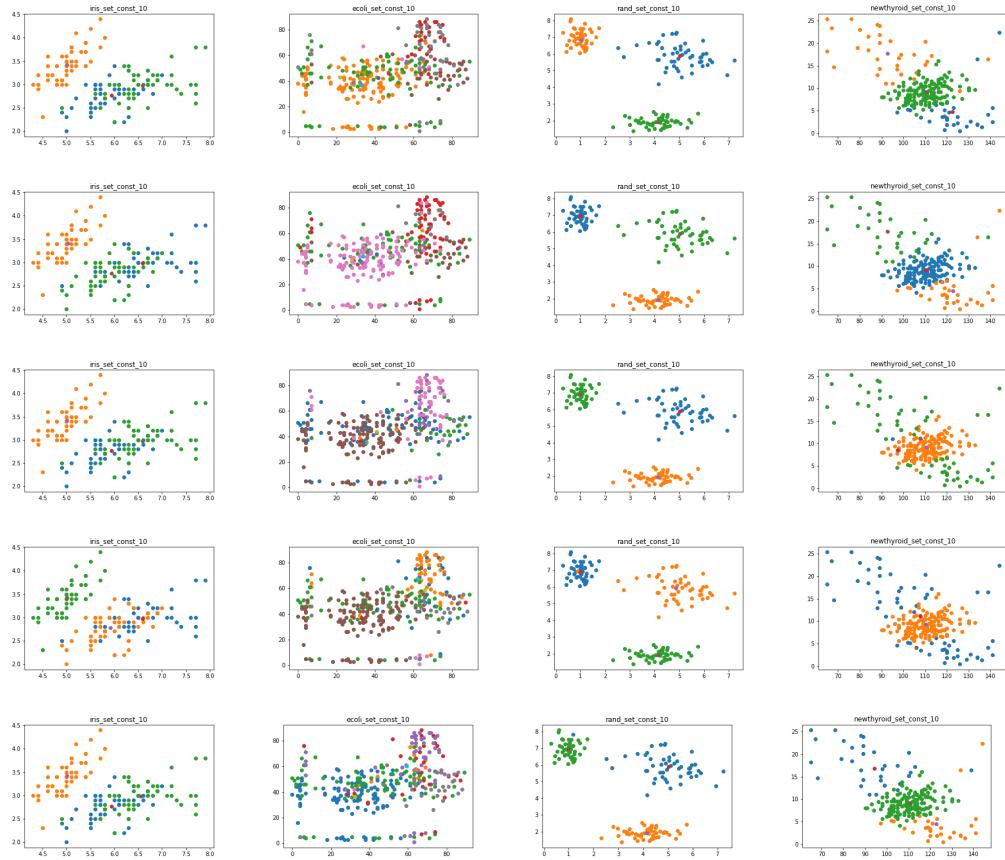


Figura 55: Primeras dimensiones de COPKM v2 para 10 % de restricciones

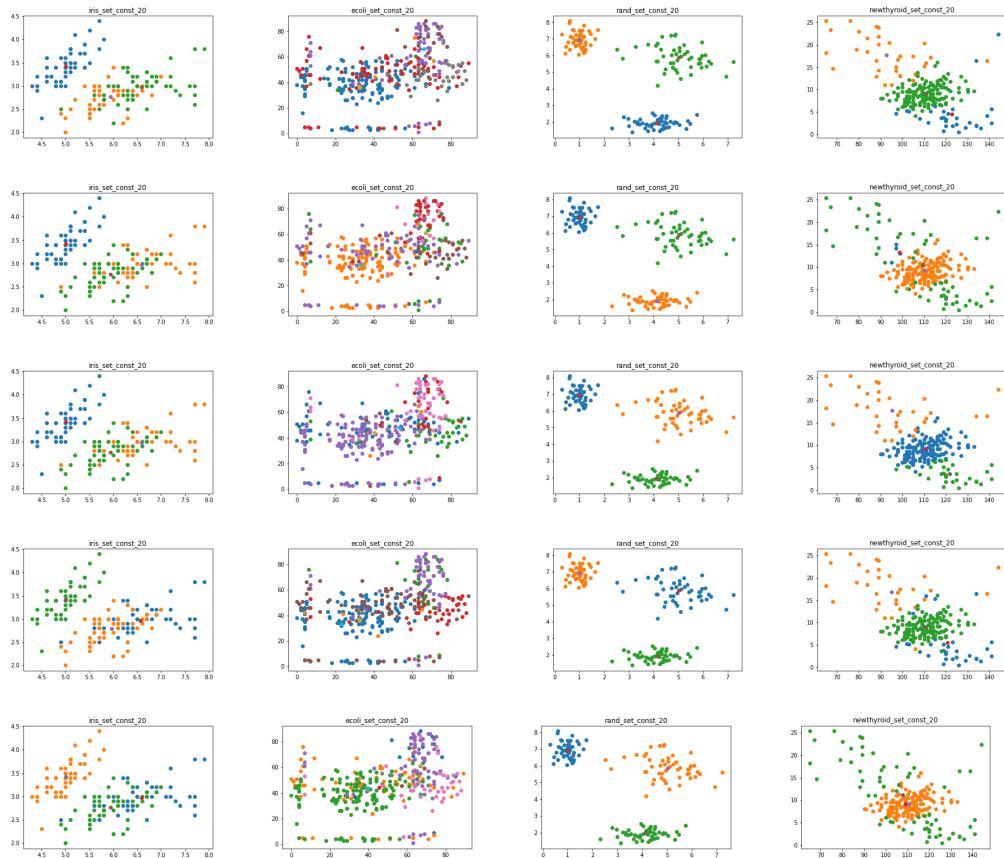


Figura 56: Primeras dimensiones de COPKM v2 para 20 % de restricciones

A.9. Algoritmo Memético (10, 0.1)

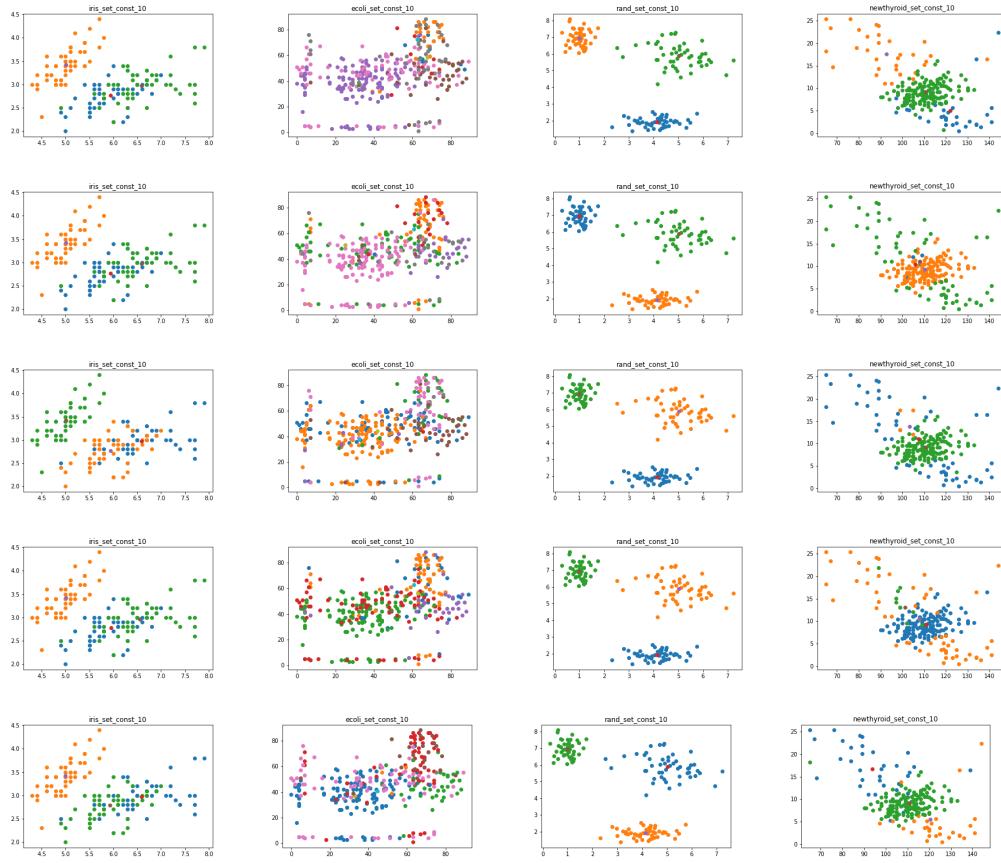


Figura 57: Primeras dimensiones de COPKM v2 para 10 % de restricciones

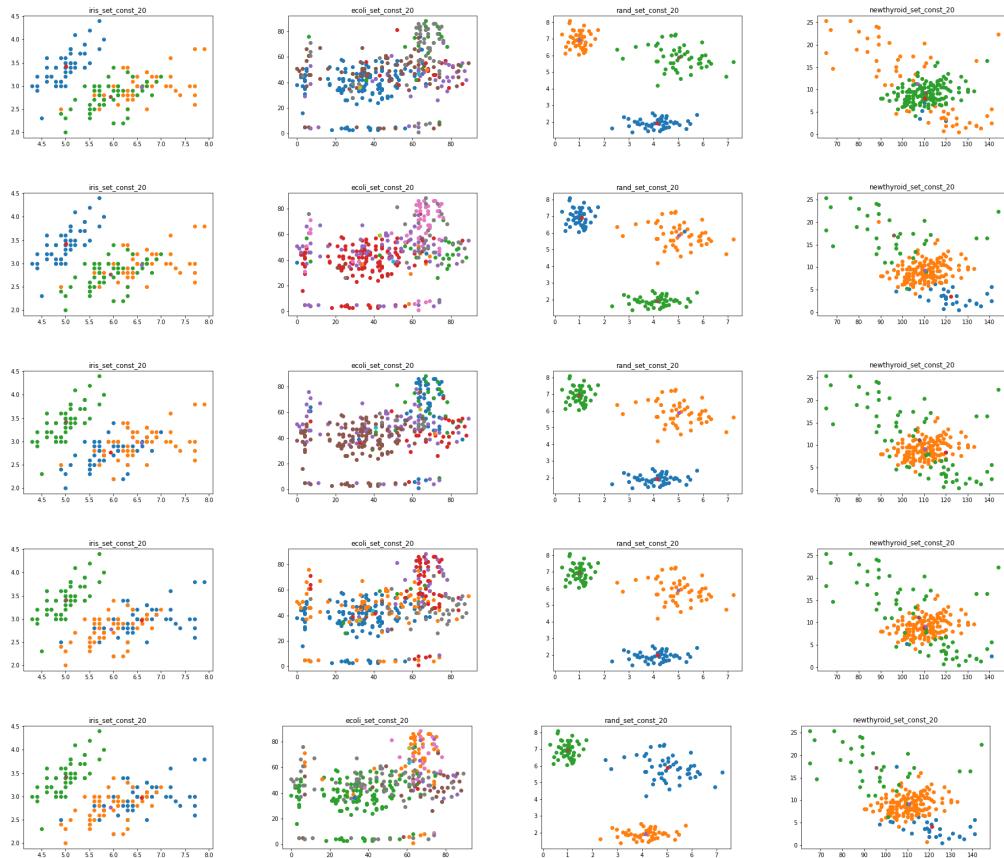


Figura 58: Primeras dimensiones de COPKM v2 para 20 % de restricciones

A.10. Algoritmo Memético (10, 0.1mej)

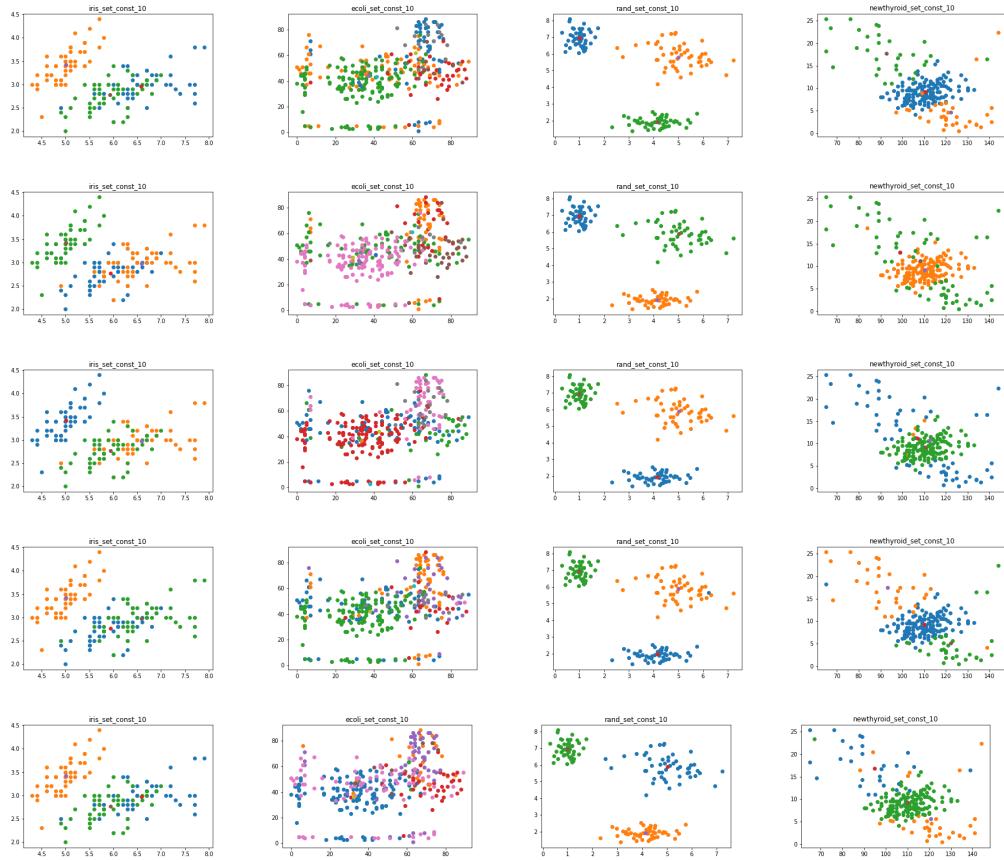


Figura 59: Primeras dimensiones de COPKM v2 para 10 % de restricciones

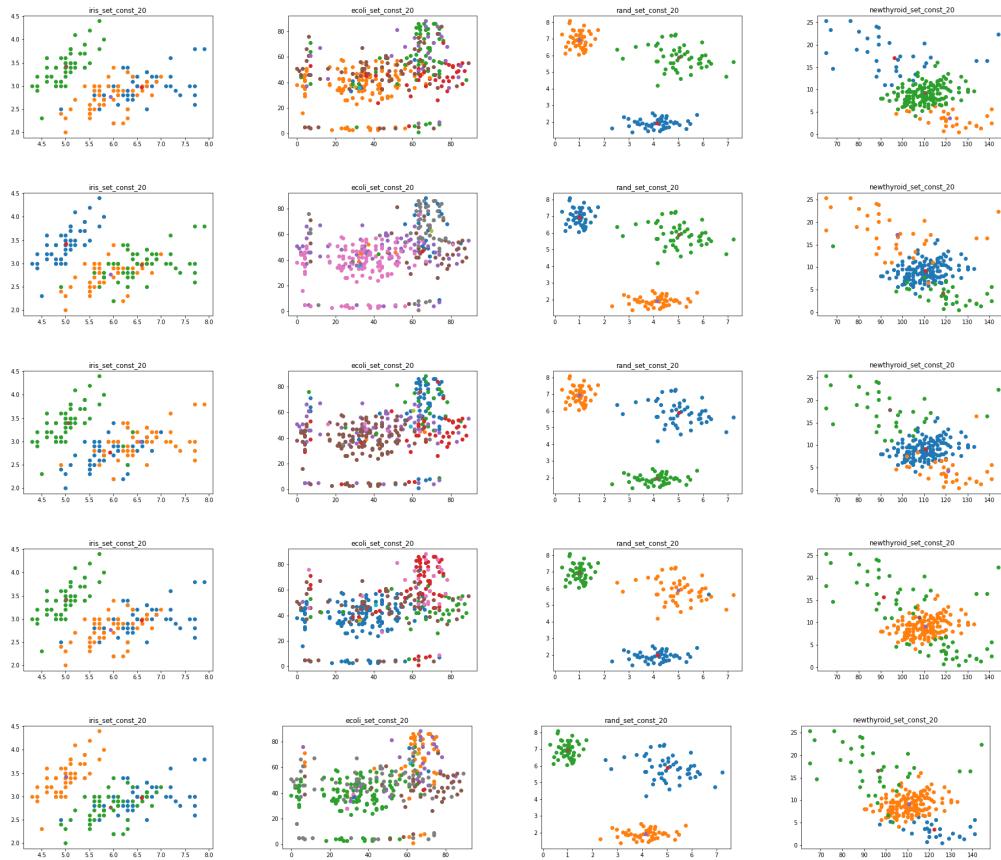


Figura 60: Primeras dimensiones de COPKM v2 para 20 % de restricciones

Referencias

- [1] Numpy. <https://numpy.org/>.
- [2] Scikit-learn. <https://scikit-learn.org/stable/>.