UNIVERSIDAD
DE GRANADA

Metaheurísticas

grado en ingeniería informática

Práctica 4

Harmonic Search para el Problema del Agrupamiento con Restricciones

Autor
Ignacio Vellido Expósito
ignaciove@correo.ugr.es
79056166Z

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2019-2020

Índice

1. Harmonic Search

1.1. Resumen

El algoritmo de búsqueda armónica (HS), originalmente propuesto en [6], toma inspiración del proceso de improvisación que sigue un grupo de músicos en busca de la armonía perfecta. Estos músicos realizan un número de prácticas, improvisando nuevas armonías en cada una de ellas. Cada práctica no comienza desde cero, y es que las improvisaciones pasadas ayudan e influyen en la producción de nuevas armonías de mejor calidad.

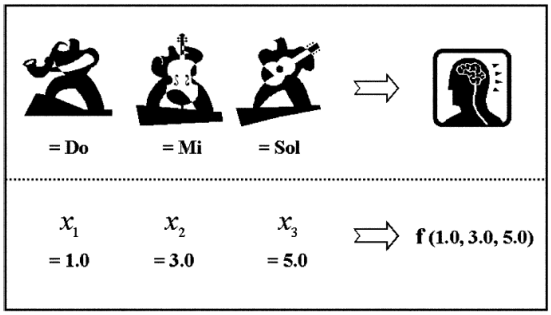

Figura 1: Inspiración de HS. Imagen tomada de [6]

Tal y como indican los autores en [3], los elementos se extrapolan a una metaheurística de la siguiente manera:

- Una armonía está formada por notas, e igualmente una solución candidata está formada a base de variables.
- La valoración estética de la armonía se transforma en una evaluación de la función objetivo.
- La mejor armonía se convierte en el mejor óptimo.
- El cambio de tono se corresponde con variar un elemento de la solución.
- Cada práctica (producción de una nueva armonía) se representa como una nueva iteración del algoritmo.

Por tanto, denominamos armonía a una solución candidata del problema $x = \{y_1, y_2, ..., y_n\}$, donde cada y_i representa una nota de la armonía.

Esta metaheurística hace uso de una estructura de datos, llamada **harmony memory** (HM), en la cuál se almacenan las armonías ya descubiertas, ordenadas en base a su calidad. De forma similar a un algoritmo generacional elitista, en cada iteración se comparará la nueva solución obtenida con la peor existente en la HM, sustituyéndola en caso de que mejore a la peor existente.

De esta manera el algoritmo pretende paliar los problemas generados por una fuerte convergencia con poca cantidad de exploración. Mediante el uso de la HM permite trabajar

con múltiples soluciones al mismo tiempo y no limitar el espacio de búsqueda por culpa de la solución inicial.

Otro concepto de relevancia para la comprensión del algoritmo es el denominado **ajuste de tono**. Este es un operador que modifica el valor de una nota una distancia por arriba o por abajo.

Los parámetros utilizados en el algoritmo son:

- **hms**: Tamaño de la harmony memory (**HM**).
- **HMCR**: Probabilidad de usar una solución ya existente en la HM.
- **PAR**: Probabilidad de ajustar el tono de un elemento en una nueva armonía. Un PAR de 0.3 indicaría que se elige un vecino con un 30 % * HMCR de probabilidad.
- **bw**: Cantidad de ajuste del tono. Indica cuánto puede mutar un elemento de la solución. Este valor es muy dependiente del problema y de si se están tratando con valores continuos o discretos.

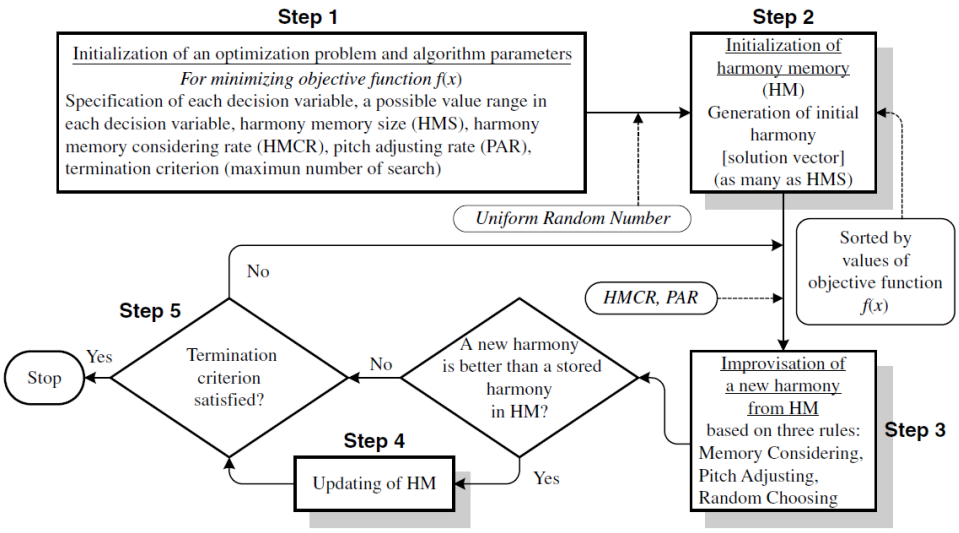

Figura 2: Proceso de búsqueda en HS. Imagen tomada de [6]

El proceso de búsqueda, representado en la figura 2, es el siguiente:

1. Inicialización de parámetros y del proceso de búsqueda.
2. Crear una memoria armónica **HM** inicial.
3. Mientras no se cumpla el criterio de parada, repetir:
   a) Improvisar una nueva armonía. Para ello repetimos para cada nota:

1) Seleccionar una nota de la HM con probabilidad HMCR, o generar una completamente nueva con probabilidad (1 - HMCR).
2) Si se ha seleccionado una nota de la HM, aplicar ajuste de tono con probabilidad PAR, y modificado en base al parámetro *bw*.
El ajuste de tono sigue la fórmula: $x_i' = x_i + bw * u(-1, 1)$.
Donde $u(-1, 1)$ es una distribución uniforme entre -1 y 1.

b) Actualizar la HM de forma elitista sustituyendo el peor elemento de la tabla por la nueva armonía en caso de que esta lo mejore.

4. Devolver la mejor armonía de la HM.

Queda claro que HMCR y PAR son los parámetros más influyentes en este algoritmo. El primero nos indica la cantidad de exploración en el espacio de búsqueda y el segundo la optimización que se realiza a las soluciones ya encontradas.

Puesto que el algoritmo no especifica ningún criterio de parada predeterminado, en nuestro caso se adoptará la condición de alcanzar un número de iteraciones fijo.

1.2. Mejoras

El mayor problema que se encuentra es la poca convergencia con la que cuenta. Para solucionarlo se proponen las siguientes alternativas:

- Cortar la ejecución en las últimas etapas del algoritmo y aplicar una búsqueda local sobre la mejor solución em la HM. Esta idea nos permite optimizar la solución una vez estamos convencidos de que es prometedora.
- Cortar la ejecución en una etapa aún más temprana y aplicar BL sobre las N mejores soluciones en la HM. De esta manera intentamos reducir el impacto de reducir el espacio de búsqueda al no convertirlo en una única solución a optimizar. Para mantener unos costes computacionales comparables, debemos reducir la cantidad de optimización aplicada, tomando el riesgo de no alcanzar completamente el óptimo.
- Reducir *hms* (el tamaño de la HM) y aplicar una búsqueda multiarranque del algoritmo.
- Modificar los parámetros del algoritmo durante su ejecución. Si reducimos el valor de HMCR durante el proceso de búsqueda, incrementamos la improvisación sobre armonías pasadas reduciendo la exploración en las últimas etapas del proceso de búsqueda. Similarmente modificando PAR de forma variable reducimos la cantidad de optimización introducida.

2. HS para el problema PAR

Todos los algoritmos se han implementado a mano en el lenguaje Python con la ayuda de los frameworks Scikit-Learn y Numpy. 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 descripcion
    -a A        Algoritmo a utilizar (hs, hs-ls, hs-ls-v2)
    -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.

2.1. 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

2.2. Consideraciones previas

2.2.1. Función objetivo

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

$$f = \overline{C} + (infeasibility * \lambda) \tag{1}$$

Siendo:

$$\lambda = \frac{\text{máx}\{d_i \in D\}}{|R|} \quad tal\ que\ D = Distancias \tag{2}$$

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

$$\overline{C} = \frac{1}{k} \sum_{c_i \in C} ||\overrightarrow{x_j} - \overrightarrow{u_j}||_2 \tag{4}$$

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, solucion, 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*** $s[r[0]] \neq s[r[1]]$ **then**
    inf++ ;
  else if $r = CL$ ***and*** $s[r[0]] = s[r[1]]$ **then**
    inf++ ;
end
return *inf*

2.2.2. 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.

2.2.3. Generación de soluciones aleatorias

La generación de soluciones aleatorias, común para todos los algoritmos, hace uso de la librería Numpy y genera un vector aleatorio relleno con posibles clústers. 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 solucion, numero de clusters
do
  solution = X donde $\bigtriangledown x_i \in X, 0 \leq x_i < numClust$;
while *solution no es válida*;
return *solution*

2.3. Adaptación de HS al problema

Como para nuestro problema estamos trabajando con variables discretas (identificadores de clústers), necesitamos adaptar el proceso de refinamiento de tono.

El algoritmo define el parámetro *bw* como la cantidad de ajuste a aplicar a una nota, tanto por encima como por debajo. Esto es así ya que se asume que si una nota aporta positivamente al coste de la solución, valores cercanos tendrán un aporte similar. Puesto que en el problema PAR no se define relación alguna entre los clústers, no importa cuál cojamos (ya sea el anterior, el siguiente, u otro cualquiera) que todos podrían alterar la solución de la misma manera.

Tras unas pequeñas pruebas, decidimos para nuestro problema los siguientes valores de los parámetros:

- HMCR = 0.75

  Recomendado por la literatura. Queremos que sea lo suficientemente alto para que optimize las armonías de la HM, pero sin olvidarse de explorar el espacio de búsqueda.

- hms = 12

  Recomendado por la literatura. Este valor parece lo suficientemente grande para considerar un buen número de armonías.

- PAR = 0.7

  Recomendado por la literatura. Es importante optimizar las armonías que se van obteniendo, pero un valor muy alto generará cambios demasiados bruscos en la solución.

- bw = Cualquier clúster diferente del actual

  Como se ha indicado previamente, este parámetro está más enfocado a problemas con variables continuas. En nuestro caso, al tratar con discretas, cogeremos todo el rango posible de clústers sin contar el actual.

- maxIteraciones = 50 000

  Se ha comprobado que es un valor más que suficiente para que el algoritmo se estabilice en una solución. Para ahorrar costes computacionales, el algoritmo puede acabar antes si en 3000 iteraciones no ha entrado ninguna armonía nueva en la HM.

De esta manera, el pseudocódigo que sigue el algoritmo es:

Algorithm 5: Algoritmo HS

```
Input: Conjunto de datos, lista de restricciones
Iniciar parámetros ;
Inicializar HM con armonías aleatorias ;
Ordenar HM en base al coste ;
while no se haya alcanzado el número máximo de iteraciones do
    /* Improvisar nueva armonía - - - - - - - - - - - - - - - - - - - - */
    newHarmony = armonía vacía ;
    for cada elemento i de la solución do
        rand = valor aleatorio en rango [0,1] ;
        if rand < HMCR then
            note = Nota i de una armonía aleatoria en la HM ;
            rand = valor aleatorio en rango [0,1] ;
            if rand < PAR then
                note = ajusteDeTono(note) ;
            end
        else
            note = Elegir un clúster aleatorio ;
        Añadir note a newHarmony ;
    end
    if newHarmony no es válida (algún cluster vacío) then
        Reparar newHarmony ;
    end
    /* Evaluar - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
    cost = coste de newHarmony ;
    if coste peor armonía en la HM > cost then
        Sustituir pero armonía por newHarmony junto a su coste ;
        Ordenar HM ;
    end
end
solution = Mejor armonía en la HM ;
return solution
```

Algorithm 6: Ajuste de tono

```
Input: oldNote, numClust
newNote = valor aleatorio en el rango [0,numClust] distinto de oldNote;
return newNote
```

2.4. Hibridización memética HS-LS

Esperamos que al tener predefinidos unos parámetros fijos la optimización no sea suficiente para converger hacia un óptimo (más tarde corroborado en el apartado *Experimentación*), por lo que se decide reducir el número de ejecuciones que realiza HS a 10000 y a aplicar una búsqueda local a la salida obtenida. El resto de hiperparámetros se mantienen a los mismos valores.

El algoritmo quedaría de la siguiente manera:

Algorithm 7: Algoritmo memético HS-LS

```
Input: Conjunto de datos, lista de restricciones
Iniciar parámetros ;
Inicializar HM con armonías aleatorias ;
Ordenar HM en base al coste ;
while no se haya alcanzado el número máximo de iteraciones do
    /* Improvisar nueva armonía - - - - - - - - - - - - - - - - - -     */
    newHarmony = armonía vacía ;
    for cada elemento i de la solución do
        rand = valor aleatorio en rango [0,1] ;
        if rand < HMCR then
            note = Nota i de una armonía aleatoria en la HM ;
            rand = valor aleatorio en rango [0,1] ;
            if rand < PAR then
                note = ajusteDeTono(note) ;
            end
        else
            note = Elegir un clúster aleatorio ;
        Añadir note a newHarmony ;
    end
    if newHarmony no es válida (algún cluster vacío) then
        Reparar newHarmony ;
    end
    /* Evaluar - - - - - - - - - - - - - - - - - - - - - - - - - -     */
    cost = coste de newHarmony ;
    if coste peor armonía en la HM > cost then
        Sustituir pero armonía por newHarmony junto a su coste ;
        Ordenar HM ;
    end
end
solution = Mejor armonía en la HM ;
/* Optimizar la solución con LS - - - - - - - - - - - - - - - - -     */
solution = localSearch() ;
return solution
```

El algoritmo de Búsqueda Local utilizado es el de la primera práctica adaptando sus parámetros, y siguiendo el pseudocódigo:

Algorithm 8: 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
```

Algorithm 9: Algoritmo de Búsqueda Local

```
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 < 10 000 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
```

2.5. Mejoras a la hibridización memética

De entre todas las mejoras propuestas en la sección anterior, se opta por incluir un modelo de parámetros adaptativos dentro de la hibridización memética. Esperamos así que haya una mayor exploración inicial con HS y la solución encontrada sea más prometedora.

La fórmulas de adaptación de parámetros son las recomendadas en [5], que siguen las fórmulas:

$$PAR_i = PAR_{min} + \frac{PAR_{max} - PAR_{min}}{maxIteraciones} * (maxIteraciones - i) \tag{5}$$

$$HMCR_i = HMCR_{min} + \frac{HMCR_{max} - HMCR_{min}}{maxIteraciones} * i \tag{6}$$

El objetivo es reducir PAR al ritmo que se incrementa HMCR. Con esto se pretende que la cantidad de exploración sea mayor en las primeras etapas del algoritmo y decremente progresivamente hacia sus iteraciones finales, concentrándose en explotar las armonías existentes en la HM.

En base a los resultados obtenidos anteriormente, los hiperparámetros se modifican de la siguiente forma:

- PAR = valores en el rango [0.01, 0.99]
- HMCR = valores en el rango [0.5, 0.95]
- maxIteraciones de la búsqueda local = 30 000

El algoritmo queda de la siguiente manera:

Algorithm 10: Algoritmo memético HS-LS con parámetros adaptativos

```
Input: Conjunto de datos, lista de restricciones
Iniciar parámetros ;
Inicializar HM con armonías aleatorias ;
Ordenar HM en base al coste ;
while no se haya alcanzado el número máximo de iteraciones do
    /* Improvisar nueva armonía - - - - - - - - - - - - - - - - - - - - */
    newHarmony = armonía vacía ;
    for cada elemento i de la solución do
        rand = valor aleatorio en rango [0,1] ;
        if rand < HMCR then
            note = Nota i de una armonía aleatoria en la HM ;
            rand = valor aleatorio en rango [0,1] ;
            if rand < PAR then
                note = ajusteDeTono(note) ;
            end
        else
            note = Elegir un clúster aleatorio ;
        Añadir note a newHarmony ;
    end
    if newHarmony no es válida (algún cluster vacío) then
        Reparar newHarmony ;
    end
    /* Evaluar - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
    cost = coste de newHarmony ;
    if coste peor armonía en la HM > cost then
        Sustituir pero armonía por newHarmony junto a su coste ;
        Ordenar HM ;
    end
    /* Ajustar hiperparámetros - - - - - - - - - - - - - - - - - - - - */
    PAR = adaptarPAR(PAR) ;
    HMCR = adaptarHMCR(HMCR) ;
end
solution = Mejor armonía en la HM ;
/* Optimizar la solución con LS - - - - - - - - - - - - - - - - - - */
solution = localSearch() ;
return solution
```

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
ES	12.20
BMB	35.11
ILS	28.13
ILS-ES	13.62
HS	38.90
HS-LS	34.75
HS-LS-v2	27.95

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
ES	0.67	0.00	0.67	22.15	21.67	82.80	23.89	304.09	0.72	0.00	0.72	14.65	12.59	43.20	14.17	33.32
BMB	1.10	193.00	2.33	200.80	45.12	1695.20	90.60	772.17	1.36	181.60	2.67	201.54	13.99	839.20	44.68	333.70
ILS	0.74	49.40	1.05	64.54	44.76	1635.80	88.65	750.26	0.84	39.80	1.13	68.12	14.28	210.20	21.96	235.05
ILS-ES	0.67	0.00	0.67	80.88	29.51	356.20	39.07	406.03	0.72	0.00	0.72	78.73	12.03	61.20	14.26	217.90
HS	1.82	443.00	4.63	178.55	45.73	1722.75	91.95	439.25	2.65	439.50	5.82	162.33	13.03	1085.00	52.71	204.76
HS-LS	1.15	229.60	2.61	98.98	45.22	1658.20	89.71	283.45	1.31	169.80	2.53	97.70	13.85	815.00	43.65	150.09
HS-LS-v2	0.82	86.40	1.37	93.81	44.22	1563.80	86.18	416.13	1.00	89.60	1.65	85.38	14.63	345.40	27.26	185.12

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
ES	0.67	0.00	0.67	23.62	21.67	131.60	42.20	530.57	0.72	0.00	0.72	20.39	12.90	92.20	14.58	121.75
BMB	1.08	378.20	2.28	297.57	45.06	3421.40	90.95	1264.94	1.33	371.00	2.67	304.10	14.43	1655.00	44.69	545.88
ILS	0.74	113.60	1.10	94.07	44.39	3332.60	89.09	1259.40	0.86	109.80	1.26	91.90	14.46	346.80	20.80	349.52
ILS-ES	0.67	0.00	0.67	118.69	28.96	718.60	38.60	687.68	0.72	0.00	0.72	127.13	14.29	0.00	14.29	470.47
HS	1.81	897.80	4.66	162.13	45.14	3511.40	92.24	583.18	2.67	896.60	5.89	184.06	13.12	2196.20	53.27	298.81
HS-LS	0.98	315.60	1.98	118.62	44.69	3384.20	90.09	379.78	1.33	377.20	2.69	118.55	13.94	1684.40	44.74	190.87
HS-LS-v2	0.75	139.80	1.19	96.62	41.97	3019.80	82.48	629.57	0.83	83.60	1.13	96.97	14.65	421.80	22.36	277.53

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

3.1. Harmonic Search

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	1.84	434.00	4.59	165.63	46.13	1709.00	91.98	578.14	2.60	437.00	5.75	134.50	12.91	1086.00	52.63	208.56
Ejecución 2	1.78	443.00	4.59	218.66	45.62	1720.00	91.77	430.94	2.58	449.00	5.82	160.83	12.98	1090.00	52.84	134.48
Ejecución 3	1.77	457.00	4.67	184.88	45.55	1736.00	92.13	404.66	2.72	427.00	5.80	264.75	13.15	1082.00	52.72	250.91
Ejecución 4	1.87	438.00	4.65	145.03	45.61	1726.00	91.92	343.28	2.70	445.00	5.90	89.23	13.08	1082.00	52.65	225.08
Ejecución 5	1.80	438.00	4.58	149.75	45.35	1744.00	92.14	395.39	2.72	441.00	5.89	162.33	13.13	1083.00	52.73	269.25
Media	1.82	443.00	4.63	178.55	45.73	1722.75	91.95	439.25	2.65	439.50	5.82	162.33	13.03	1085.00	52.71	204.76

Cuadro 4: Resultados HS para 10 % de restricciones

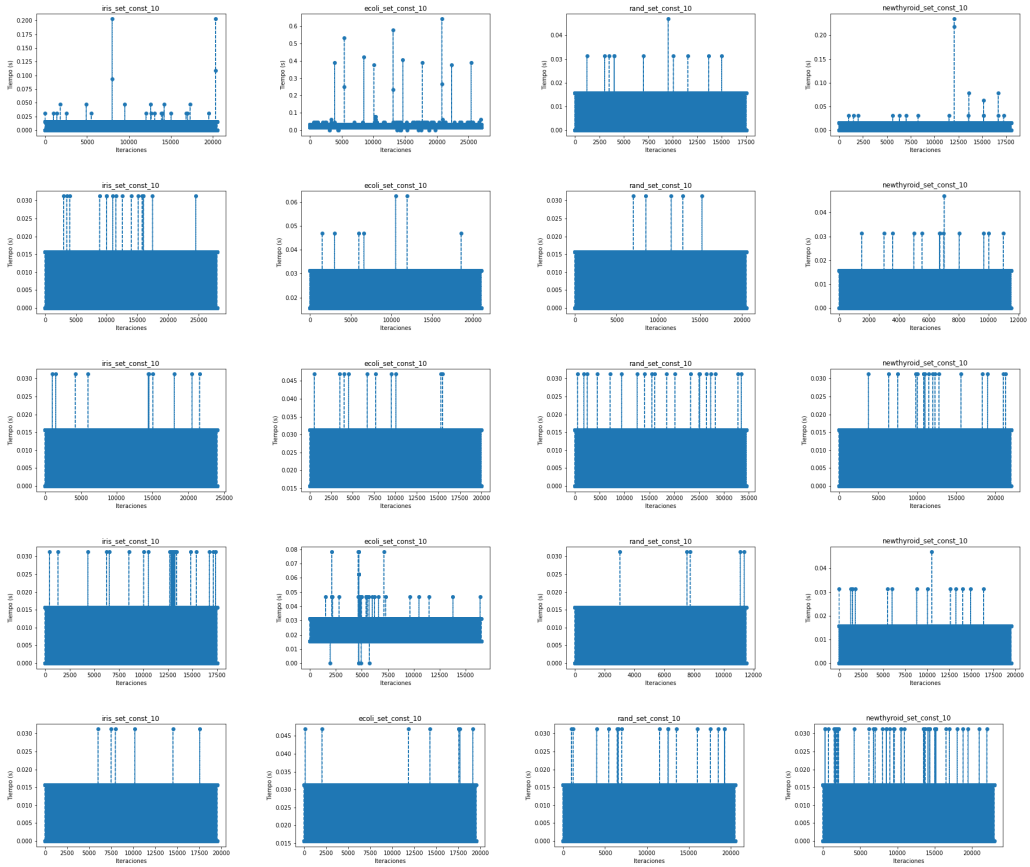

Figura 3: Tiempos de hs para 10 % de restricciones

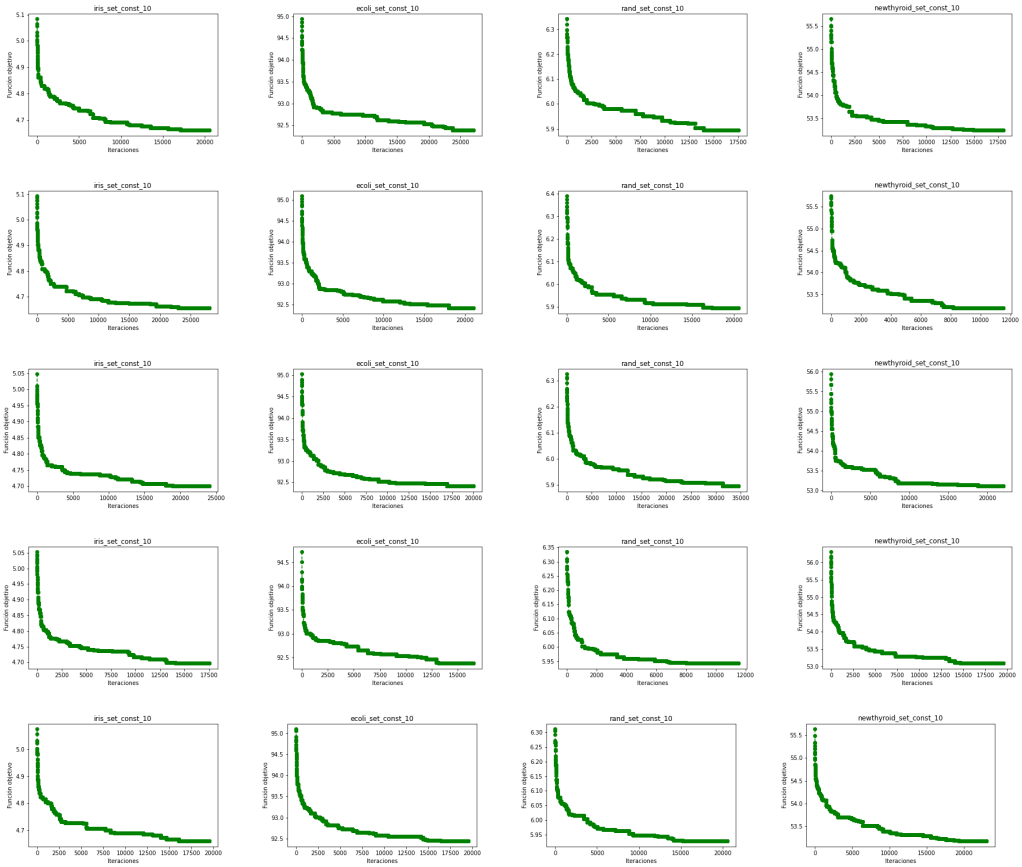

Figura 4: Agregado de hs para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	1.75	910.00	4.63	152.88	44.77	3538.00	92.23	691.66	2.70	895.00	5.92	222.00	12.86	2217.00	53.39	240.78
Ejecución 2	1.85	897.00	4.69	110.63	45.07	3513.00	92.20	817.97	2.63	897.00	5.86	177.34	13.03	2211.00	53.45	223.33
Ejecución 3	1.84	900.00	4.69	156.75	45.49	3492.00	92.34	595.61	2.66	905.00	5.92	174.86	13.17	2185.00	53.12	416.55
Ejecución 4	1.83	896.00	4.67	215.70	45.06	3491.00	91.89	441.19	2.69	895.00	5.92	199.89	13.17	2201.00	53.41	407.69
Ejecución 5	1.80	886.00	4.60	174.72	45.29	3523.00	92.55	369.47	2.65	891.00	5.85	146.22	13.36	2167.00	52.97	205.69
Media	1.81	897.80	4.66	162.13	45.14	3511.40	92.24	583.18	2.67	896.60	5.89	184.06	13.12	2196.20	53.27	298.81

Cuadro 5: Resultados HS para 20 % de restricciones

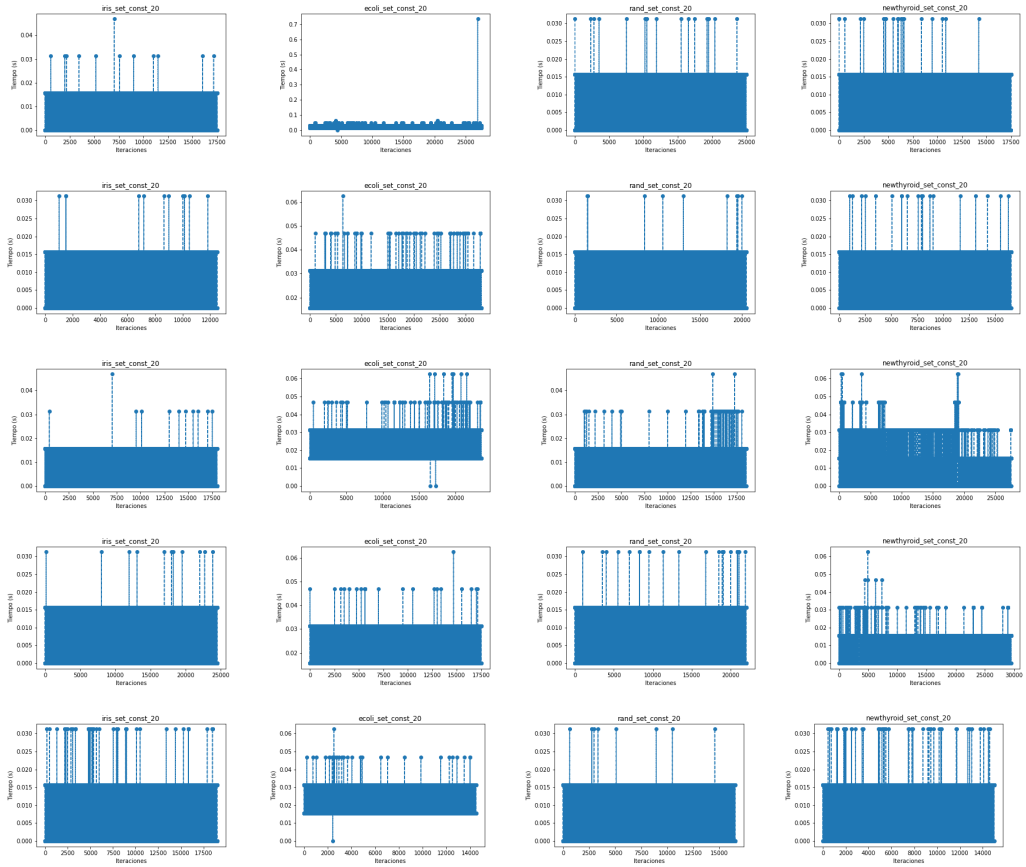

Figura 5: Tiempos de HS para 20 % de restricciones

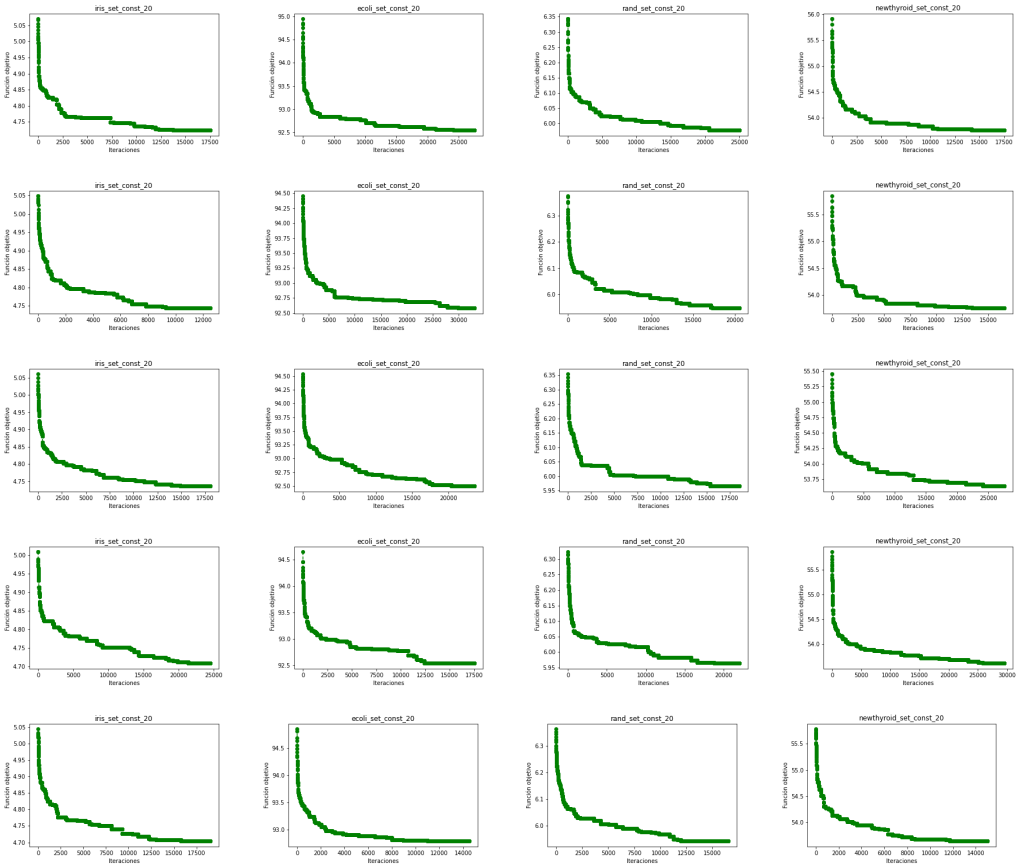

Figura 6: Agregado de HS para 20 % de restricciones

3.2. HS-LS

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	1.06	181.00	2.21	99.22	44.69	1673.00	89.57	286.69	1.40	207.00	2.89	97.42	13.26	756.00	40.91	149.02
Ejecución 2	1.23	256.00	2.85	100.92	45.68	1647.00	89.86	283.44	1.14	123.00	2.02	97.28	13.43	883.00	45.72	154.86
Ejecución 3	1.28	280.00	3.06	96.44	45.25	1646.00	89.41	276.69	1.27	169.00	2.48	97.03	14.35	784.00	43.02	150.48
Ejecución 4	1.07	203.00	2.36	98.39	45.00	1669.00	89.78	282.45	1.35	183.00	2.67	97.70	14.20	812.00	43.90	149.92
Ejecución 5	1.13	228.00	2.58	99.94	45.47	1656.00	89.90	288.00	1.39	167.00	2.60	99.06	13.99	840.00	44.71	146.16
Media	1.15	229.60	2.61	98.98	45.22	1658.20	89.71	283.45	1.31	169.80	2.53	97.70	13.85	815.00	43.65	150.09

Cuadro 6: Resultados HS-LS para 10 % de restricciones

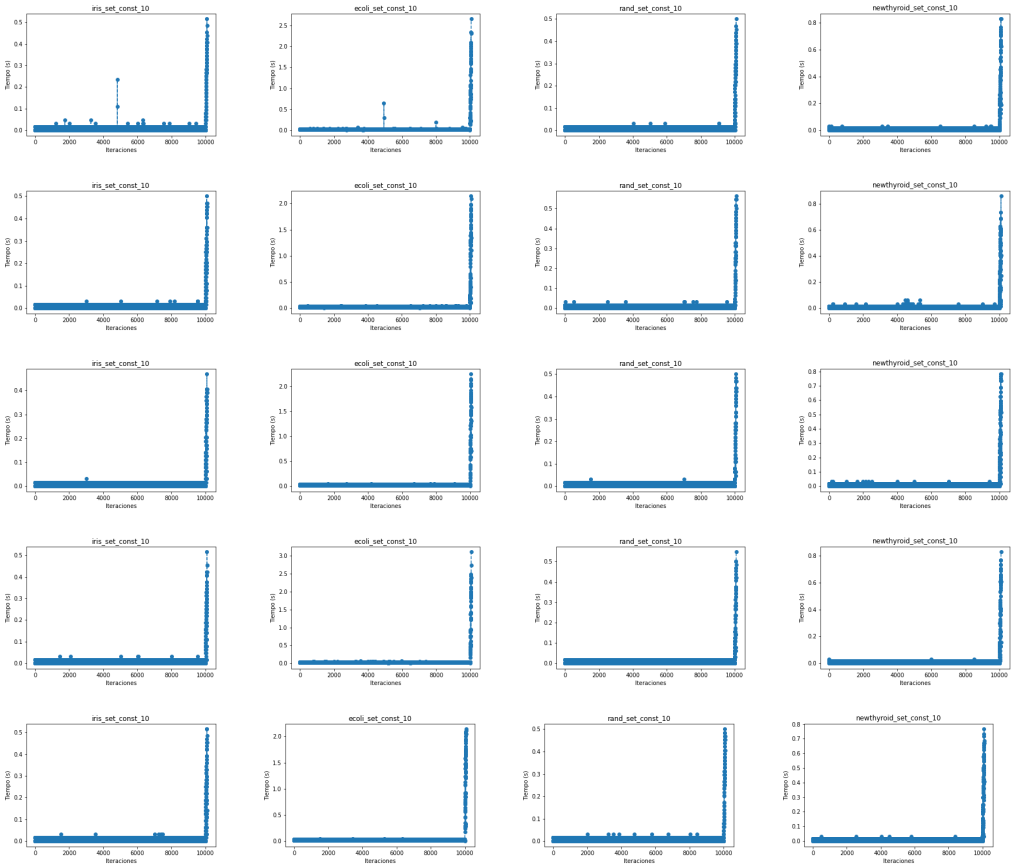

Figura 7: Tiempos de HS-LS para 10 % de restricciones

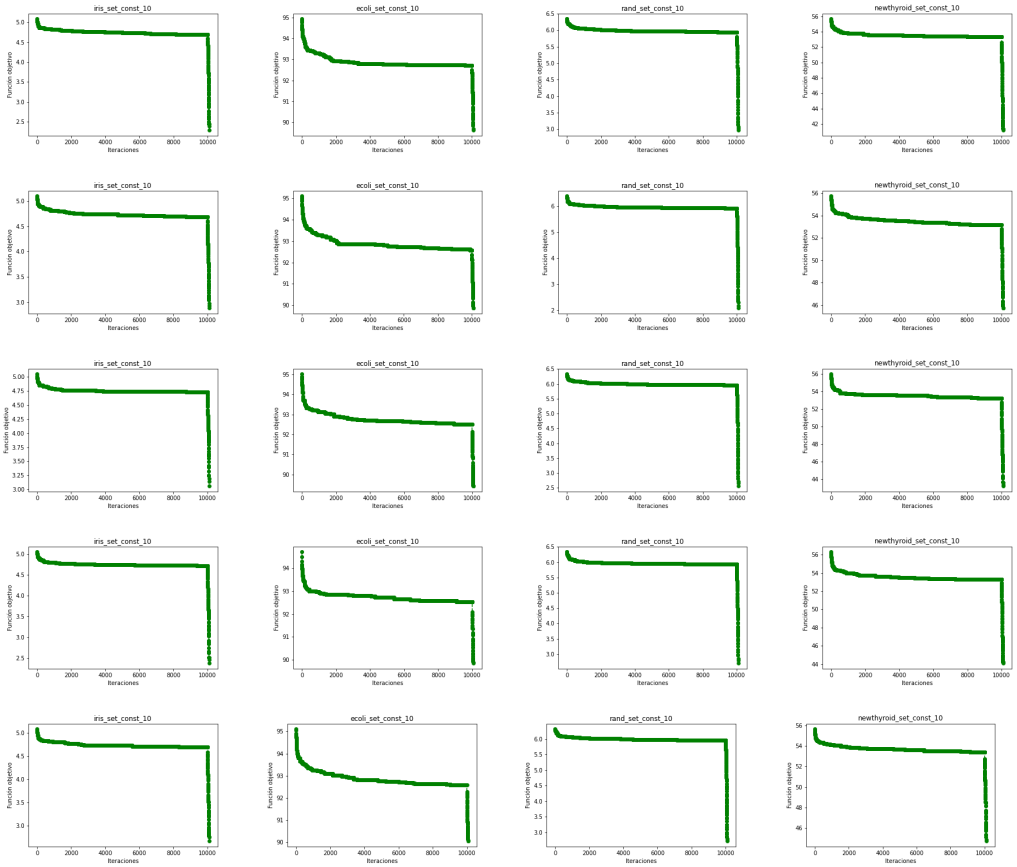

Figura 8: Agregado de HS-LS para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.89	303.00	1.85	117.97	44.84	3365.00	89.98	377.77	1.27	369.00	2.60	118.67	13.60	1675.00	44.23	189.53
Ejecución 2	1.01	325.00	2.04	120.91	44.23	3408.00	89.95	381.22	1.19	317.00	2.33	117.42	13.73	1708.00	44.95	189.78
Ejecución 3	1.06	360.00	2.20	119.00	44.83	3393.00	90.35	381.61	1.45	387.00	2.84	119.64	13.59	1874.00	47.85	191.17
Ejecución 4	1.01	343.00	2.09	118.92	44.92	3361.00	90.01	381.64	1.31	410.00	2.79	119.69	14.38	1563.00	42.96	192.20
Ejecución 5	0.91	247.00	1.70	116.28	44.64	3394.00	90.17	376.69	1.42	403.00	2.87	117.31	14.40	1602.00	43.69	191.66
Media	0.98	315.60	1.98	118.62	44.69	3384.20	90.09	379.78	1.33	377.20	2.69	118.55	13.94	1684.40	44.74	190.87

Cuadro 7: Resultados HS-LS para 20 % de restricciones

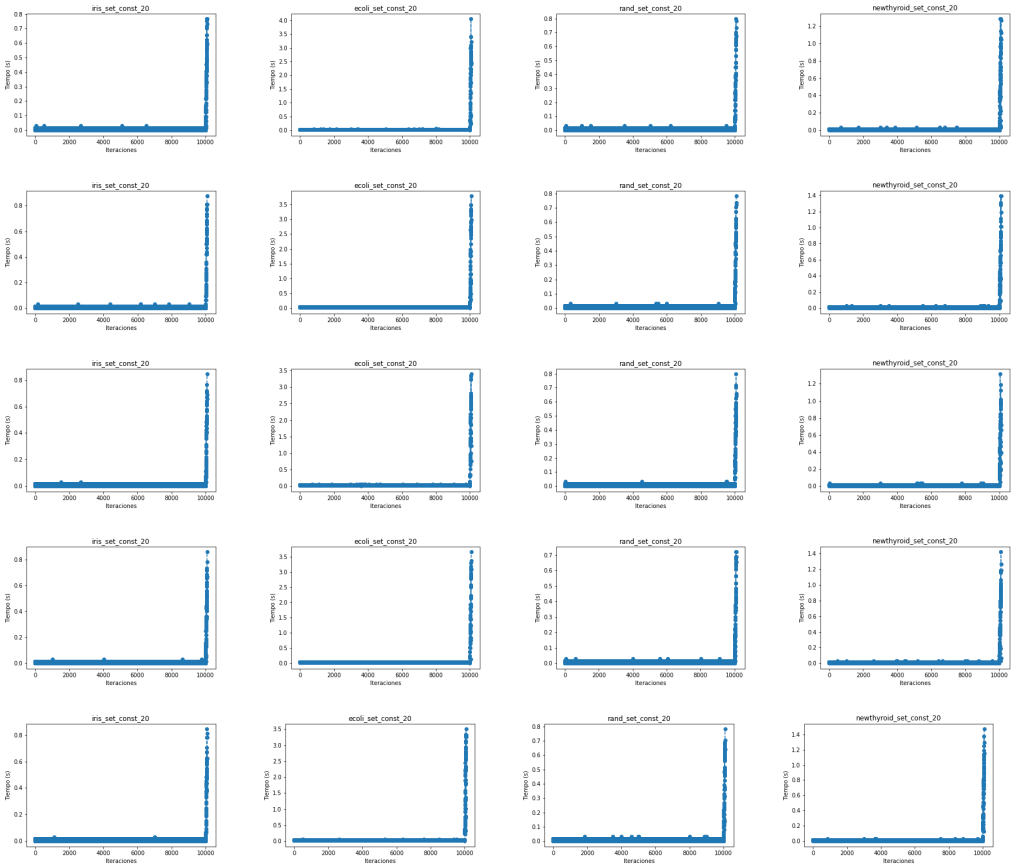

Figura 9: Tiempos de HS-LS para 20 % de restricciones

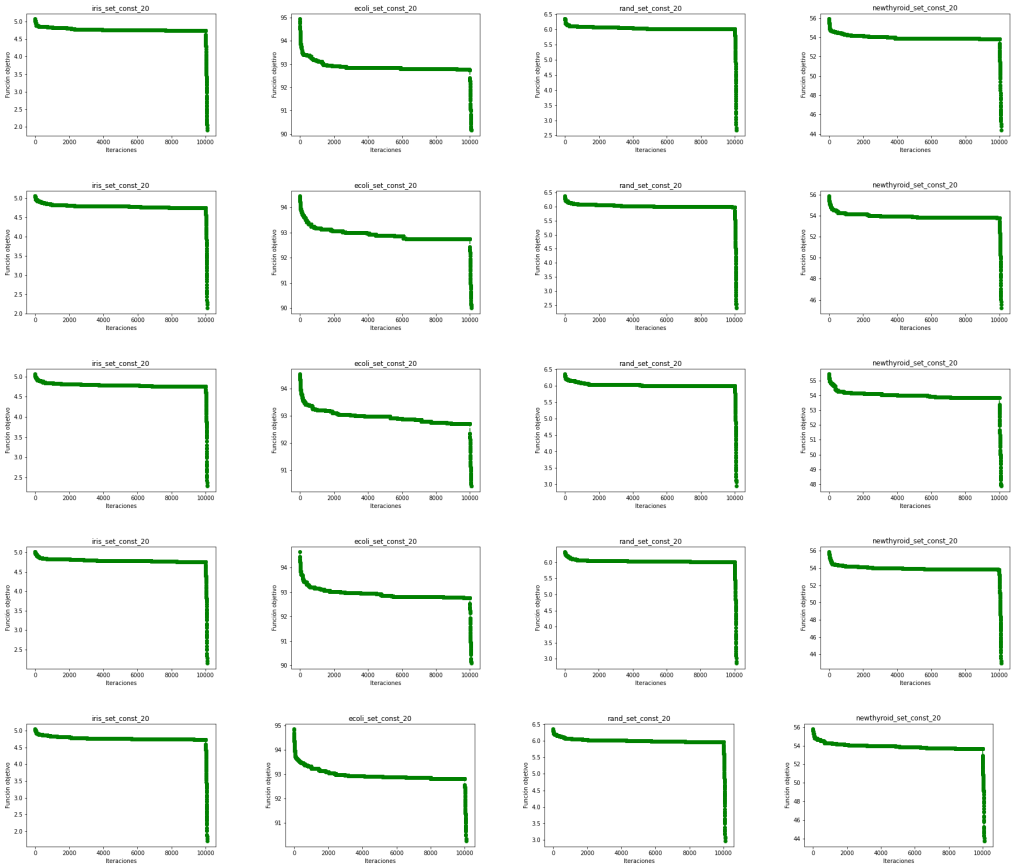

Figura 10: Agregado de HS-LS para 20 % de restricciones

3.3. HS-LS con parámetros adaptativos

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.85	80.00	1.35	110.08	44.61	1604.00	87.64	412.98	0.93	66.00	1.40	81.63	14.47	273.00	24.46	159.11
Ejecución 2	0.81	64.00	1.22	81.95	43.97	1574.00	86.20	411.55	1.05	117.00	1.89	94.23	13.49	367.00	26.91	184.94
Ejecución 3	0.83	117.00	1.58	85.20	44.21	1590.00	86.87	407.25	1.03	77.00	1.59	83.47	14.15	418.00	29.44	206.19
Ejecución 4	0.85	97.00	1.47	97.78	43.46	1500.00	83.70	409.33	1.02	90.00	1.67	74.83	16.23	228.00	24.57	161.13
Ejecución 5	0.75	74.00	1.22	94.02	44.87	1551.00	86.48	439.55	0.97	98.00	1.67	92.75	14.79	441.00	30.92	214.23
Media	0.82	86.40	1.37	93.81	44.22	1563.80	86.18	416.13	1.00	89.60	1.65	85.38	14.63	345.40	27.26	185.12

Cuadro 8: Resultados HS-LS-v2 para 10 % de restricciones

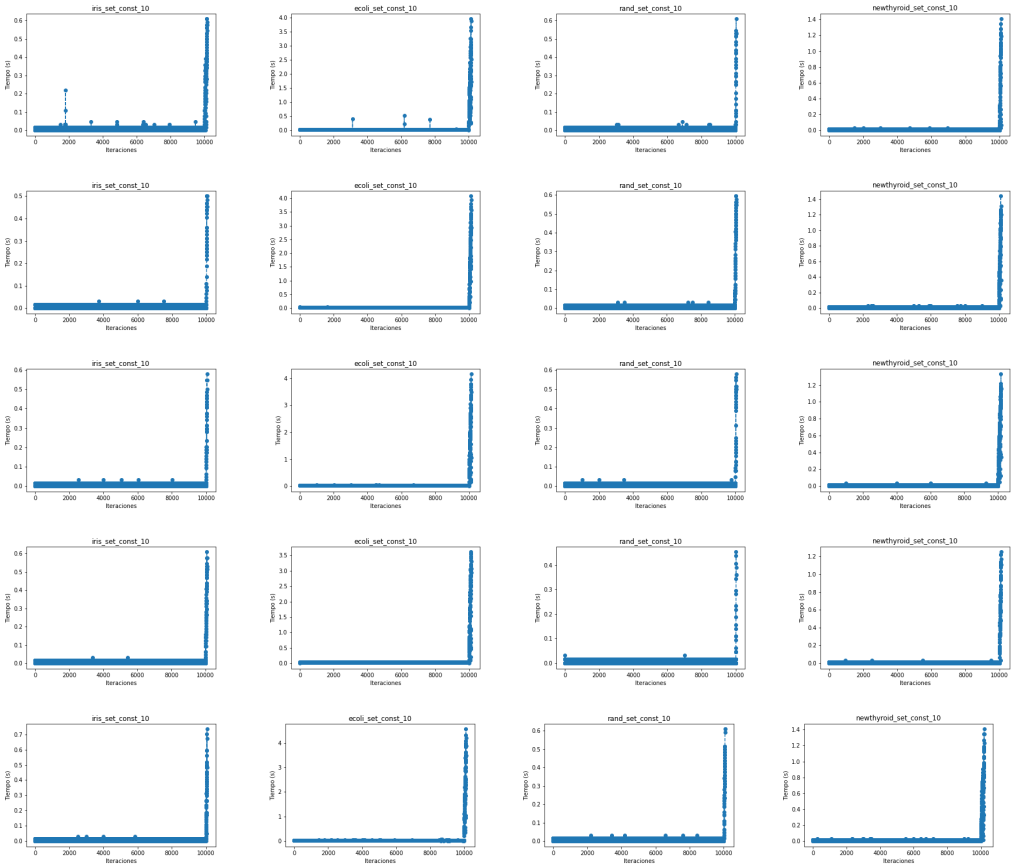

Figura 11: Tiempos de HS-LS-v2 para 10 % de restricciones

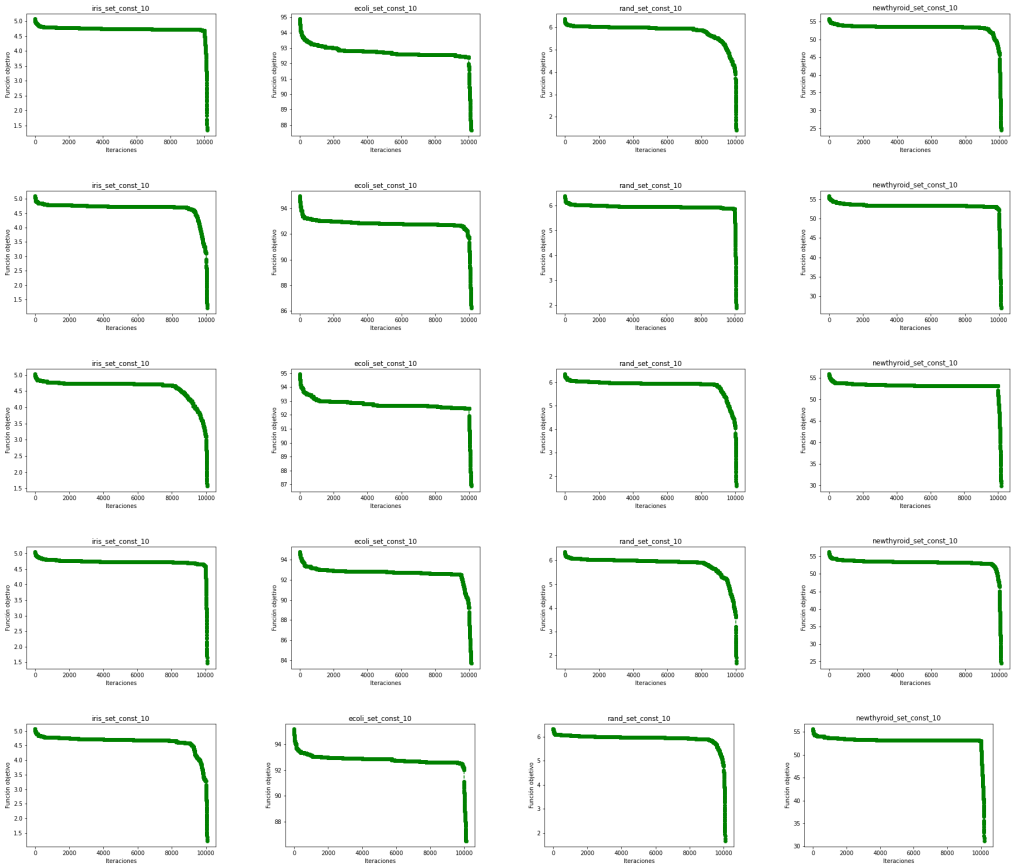

Figura 12: Agregado de HS-LS-v2 para 10 % de restricciones

	Iris				Ecoli				Rand				Newthyroid			
	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T	TasaC	TasaInf	Agr	T
Ejecución 1	0.72	88.00	1.00	98.55	40.95	2913.00	80.02	619.88	0.81	85.00	1.12	94.09	14.44	409.00	21.92	284.08
Ejecución 2	0.81	215.00	1.49	92.19	41.43	3100.00	83.01	625.80	0.78	75.00	1.05	94.94	14.99	344.00	21.27	249.31
Ejecución 3	0.74	107.00	1.08	97.53	42.20	3081.00	83.53	633.17	0.84	81.00	1.13	98.22	14.53	489.00	23.47	303.86
Ejecución 4	0.77	206.00	1.42	88.41	41.73	2754.00	78.67	624.61	0.85	87.00	1.17	97.59	15.12	480.00	23.90	326.47
Ejecución 5	0.69	83.00	0.96	106.44	43.55	3251.00	87.16	644.41	0.86	90.00	1.18	100.02	14.18	387.00	21.26	223.95
Media	0.75	139.80	1.19	96.62	41.97	3019.80	82.48	629.57	0.83	83.60	1.13	96.97	14.65	421.80	22.36	277.53

Cuadro 9: Resultados HS-LS-v2 para 20 % de restricciones

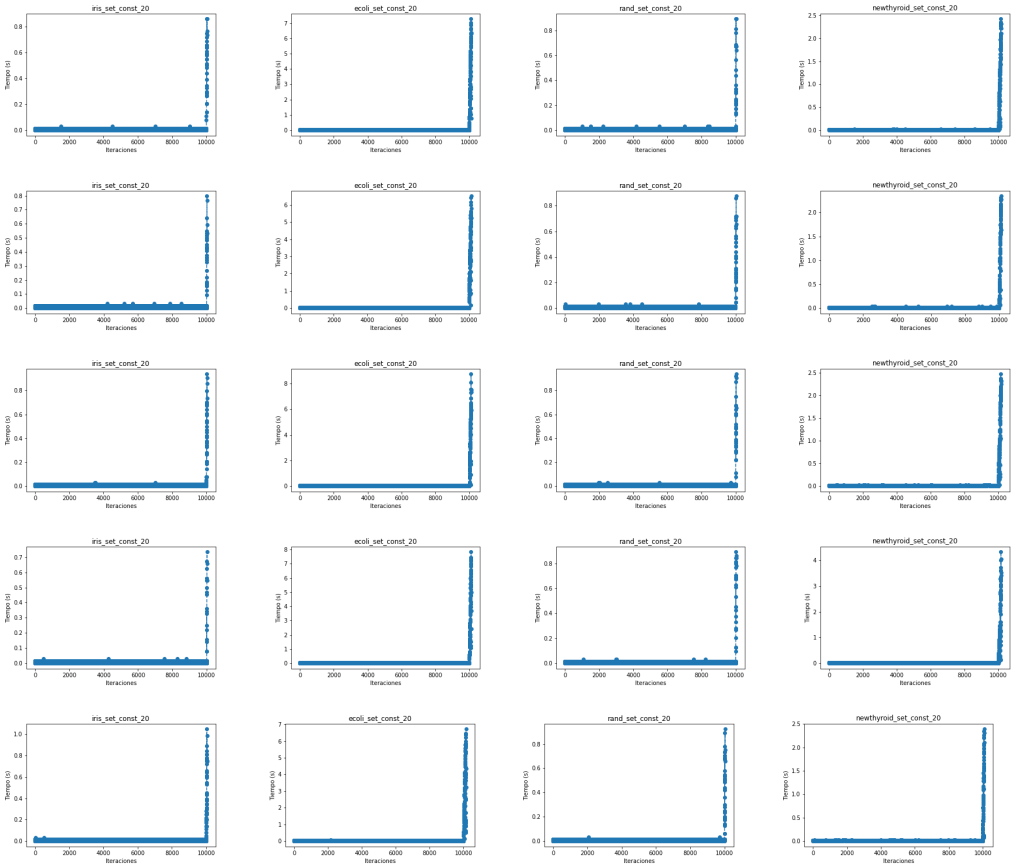

Figura 13: Tiempos de HS-LS-v2 para 20 % de restricciones

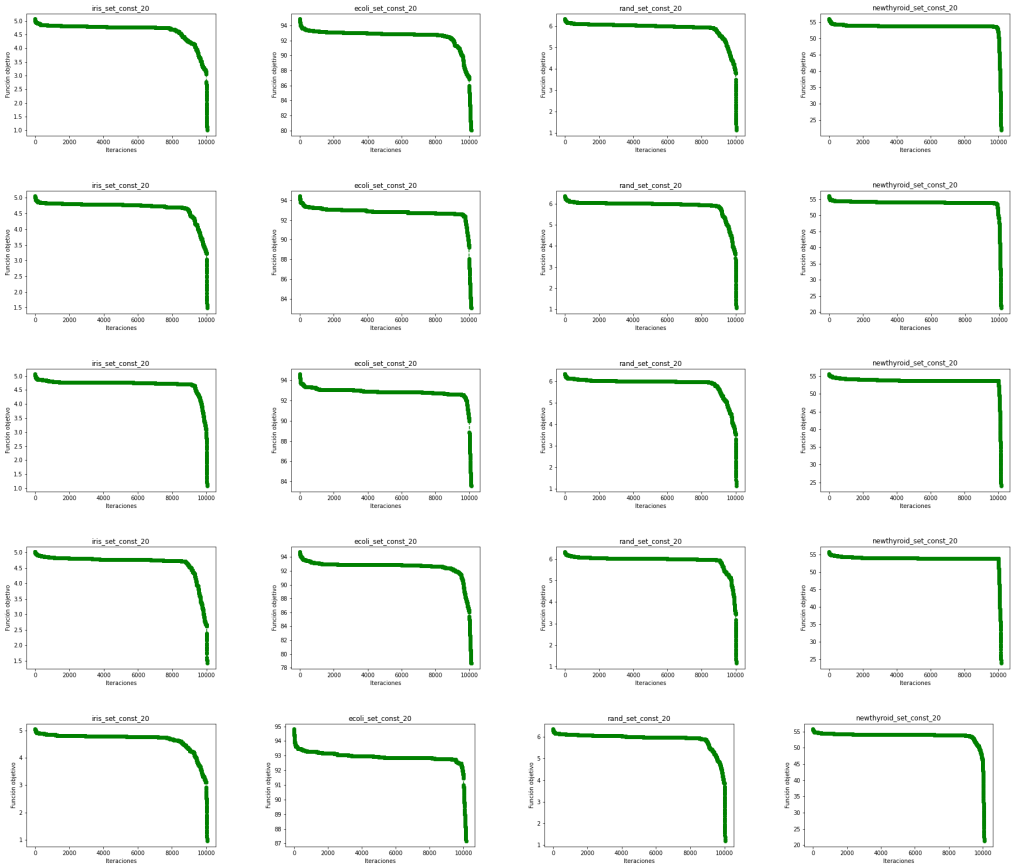

Figura 14: Agregado de HS-LS-v2 para 20 % de restricciones

4. Conclusiones

Se recuerda que 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 extremádamente 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 por ejemplo hace 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. Harmonic Search

Los resultados son bastante pobres comparando con el resto de algoritmos. Viendo las gráficas (de coste medio en la HM) queda claro que se estanca con mucha facilidad en óptimos locales. Una de las posibles causas de esto sea los hiperparámetros obtenidos, pero una exploración de diferentes valores iniciales no da mejores resultados.

Valores altos de PAR hacen que las armonías se modifiquen tanto que tenga el mismo resultado que generar una nueva. En otro tipo de problema donde se trate con valores continuos y se haga el ajuste de tono propuesto debería funcionar mucho mejor.

Como se ha dicho anteriormente, HS con el operador de ajuste de tono asume que si una nota aporta positivamente al coste de la solución, valores cercanos tendrán un aporte similar. Puesto que en el problema PAR no se define relación alguna entre los clústers, no importa cuál cojamos (ya sea el anterior, el siguiente, u otro cualquiera) que todos podrían alterar la solución de la misma manera.

Da la sensación de que aunque sea posible (y la propuesta original lo mantenga), HS no está pensado para variables discretas. Parece que las complicaciones con ajuste de tono debería ocurrir en todos los problemas que tengan estas características.

Por otro lado, puesto que el algoritmo le da importancia a coger notas ya descubiertas en otras armonías, parece que espera que los valores sean bastante diferentes entre los elementos de la HM. Con variables discretas esto no pasa, pues tenemos un grupo limitado de posibilidades. Fácilmente una HM de un tamaño suficiente podría cubrir todas las notas para un elemento de la solución.

Por último, no parece sensato comparar los tiempos puesto que los resultados son mucho peores que con el resto de algoritmos, pero podríamos decir que en condiciones óptimas debería seguir siendo más lento que los genéticos y el enfriamiento simulado puesto que HS trabaja elemento a elemento de la solución.

4.2. HS-LS

Primeramente, las gráficas nos muestran que unas cuantas evaluaciones más habrían dado mejores resultados, pues parece que aún queda margen de convergencia. Aunque, tras aumentarlo en la siguiente mejora, se aprecia que el valor óptimo de iteraciones no era mucho mayor.

Esto nos indica que las soluciones devueltas por HS no son prometedoras, puesto que apuntan hacia malos óptimos locales, y que la exploración del espacio de búsqueda inicial no es suficiente para introducir en la HM armonías de calidad.

El hecho de que independientemente de la semilla hayamos obtenido resultados pésimos nos indica que una metodología multiarranque no habría mejorado mucho los resultados. La cantidad de exploración inicial que se realiza en ILS-ES no se acabaría dando en nuestro caso.

4.3. HS-LS con parámetros adaptativos

Vemos que aunque existe mejoría respecto a los dos anteriores, el uso de parámetros adaptativos sigue sin ser suficiente. Al igual que la versión anterior la mejora es más notoria en los conjuntos de datos sencillos que aquellos difíciles.

HS está inicialmente pensado para trabajar solo y gracias a HSMR y PAR ser capaz de tener un balance en exploración-explotación. Puesto que para nuestro problema no se adapta bien, hemos visto que la introducción de una LS a la salida de HS nos permite optimizar el óptimo lo máximo posible, pero no resulta ser tan prometedor como los obtenidos con otros algoritmos.

Aún así, el ir variando los valores, tal y como se explica en el apartado *Mejoras a la hibridación memética*, conseguimos que la cantidad de exploración sea mayor en las primeras etapas del algoritmo y decremente progresivamente hacia sus iteraciones finales, concentrándose en explotar las armonías existentes en la HM.

De todas maneras vemos en las gráficas que el algoritmo no devuelve una armonía optimizada, por lo que mantener la LS sigue siendo importante.

En resumen, incluso con las mejoras experimentadas HS no llega a ser suficiente para superar al algoritmo de búsqueda local, lo que nos indica que:

1. La forma de adaptar el algoritmo al problema no es idónea. En la literatura otras aproximaciones han usado HS para calcular los centroides y K-Means para posicionar los elementos. Pero lo más importante es no modificar el operador de ajuste de tono y mantenerlo con valores continuos.

2. Los parámetros pueden no estar bien ajustados, pero por las características del algoritmo no resulta viable dejarlos fijos durante toda la ejecución.

3. Es posible que el algoritmo no sea suficiente para este problema, y haya que añadir aún más modificaciones.

A. Representación visual de primeras dimensiones

A.1. Harmonic Search

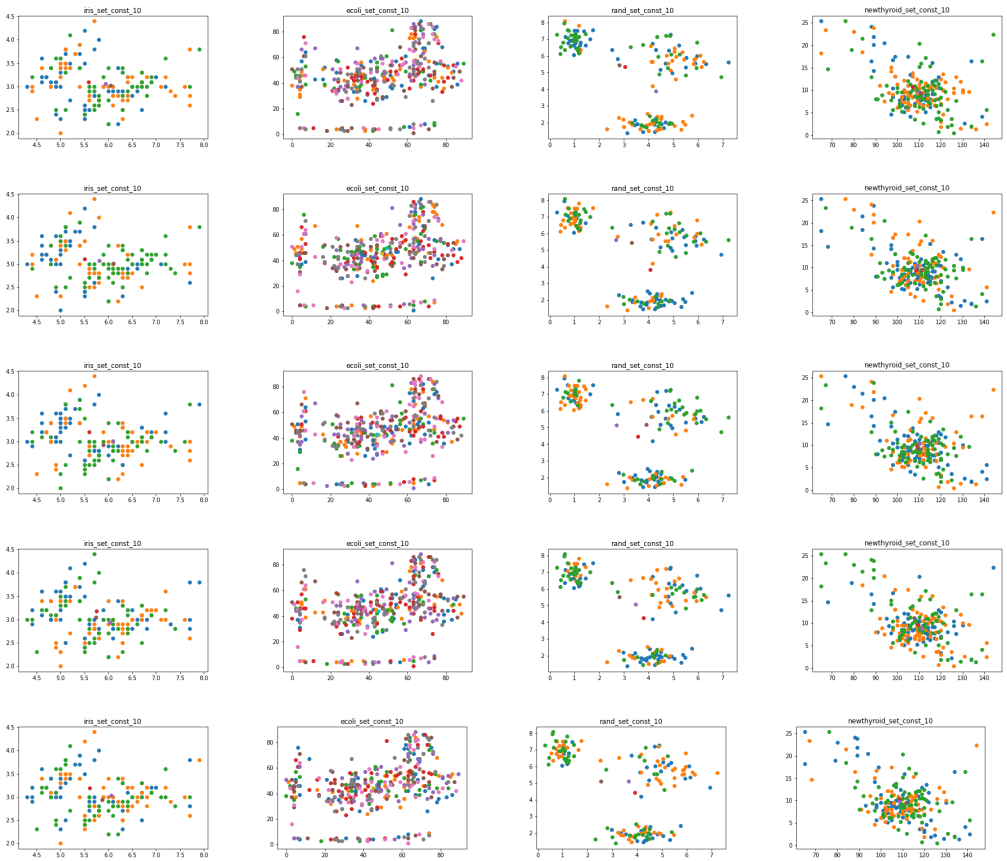

Figura 15: Primeras dimensiones de COPKM para 10 % de restricciones

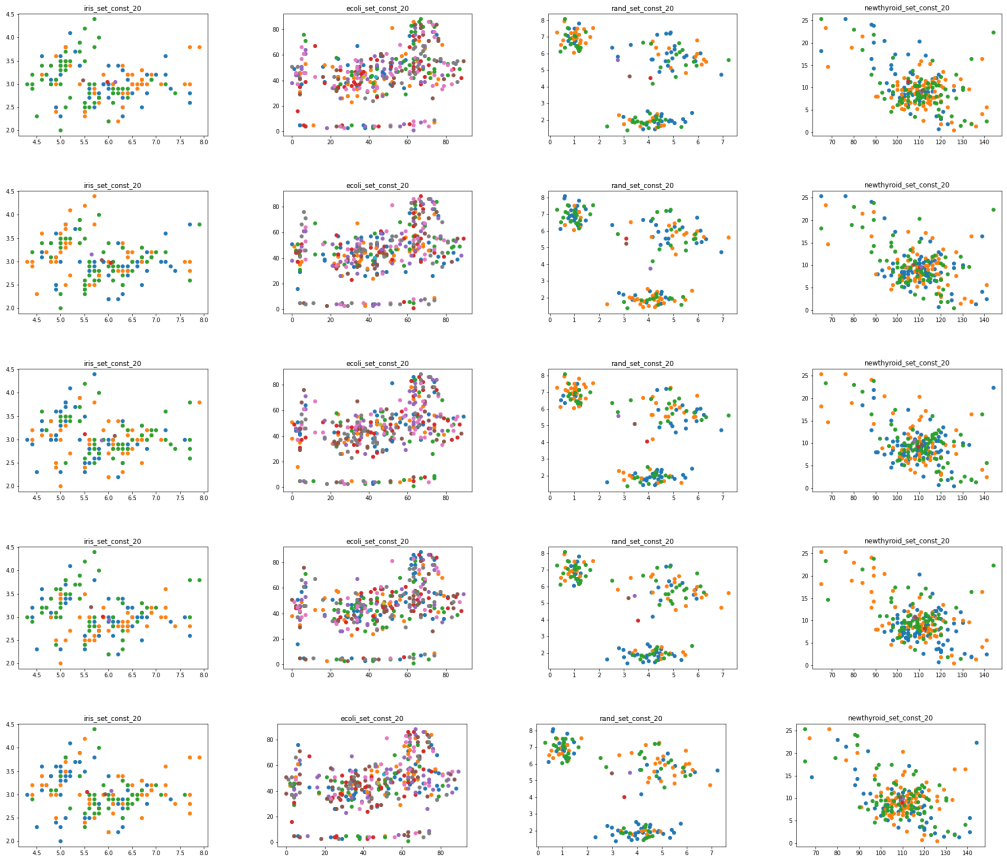

Figura 16: Primeras dimensiones de COPKM para 20 % de restricciones

A.2. HS-LS

Figura 17: Primeras dimensiones de COPKM v2 para 10 % de restricciones

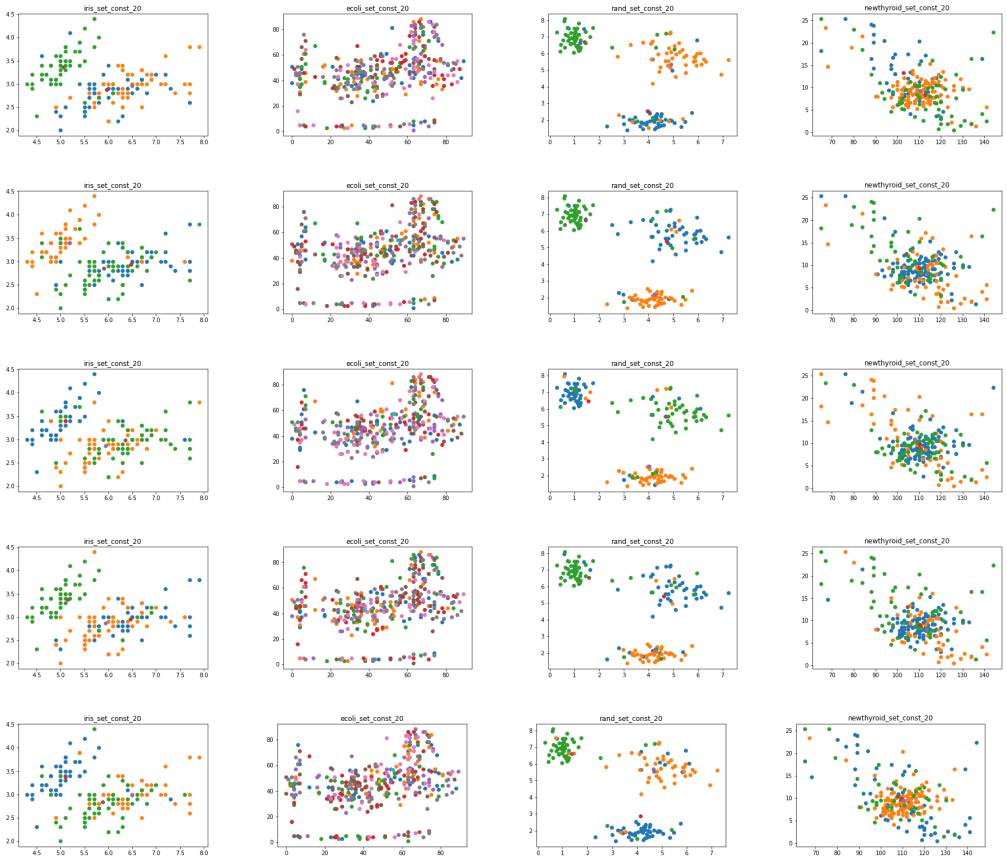

Figura 18: Primeras dimensiones de COPKM v2 para 20 % de restricciones

A.3. HS-LS con parámetros adaptativos

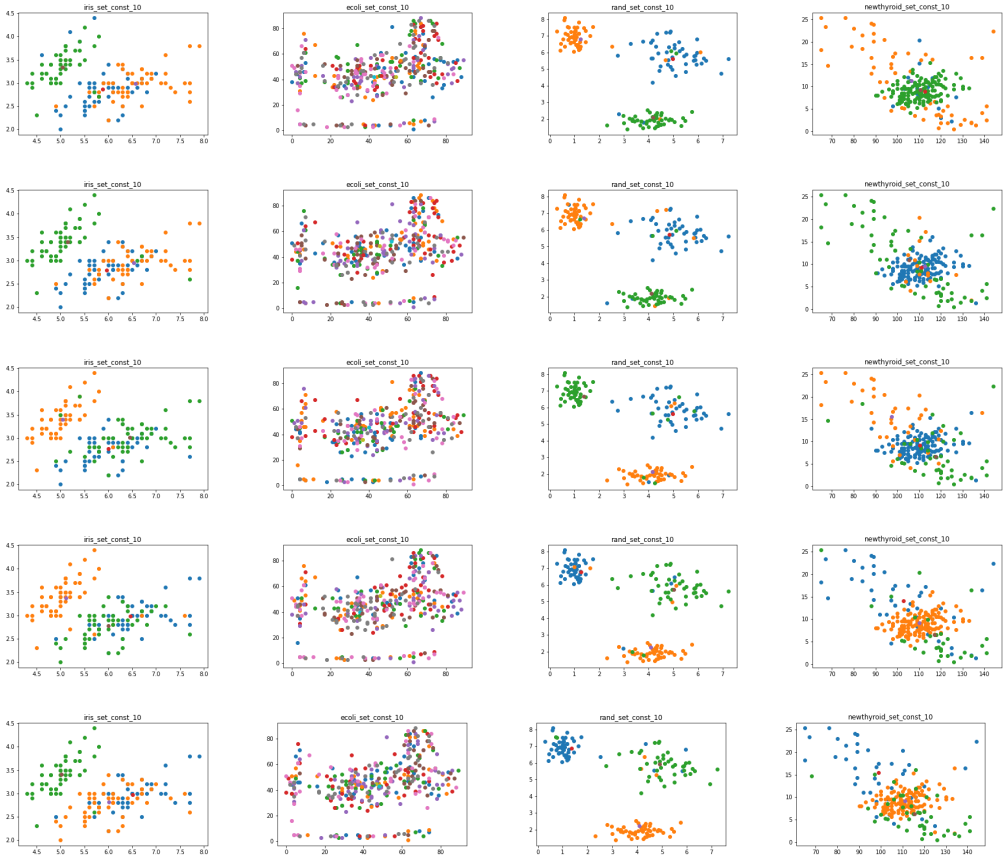

Figura 19: Primeras dimensiones de COPKM v2 para 10 % de restricciones

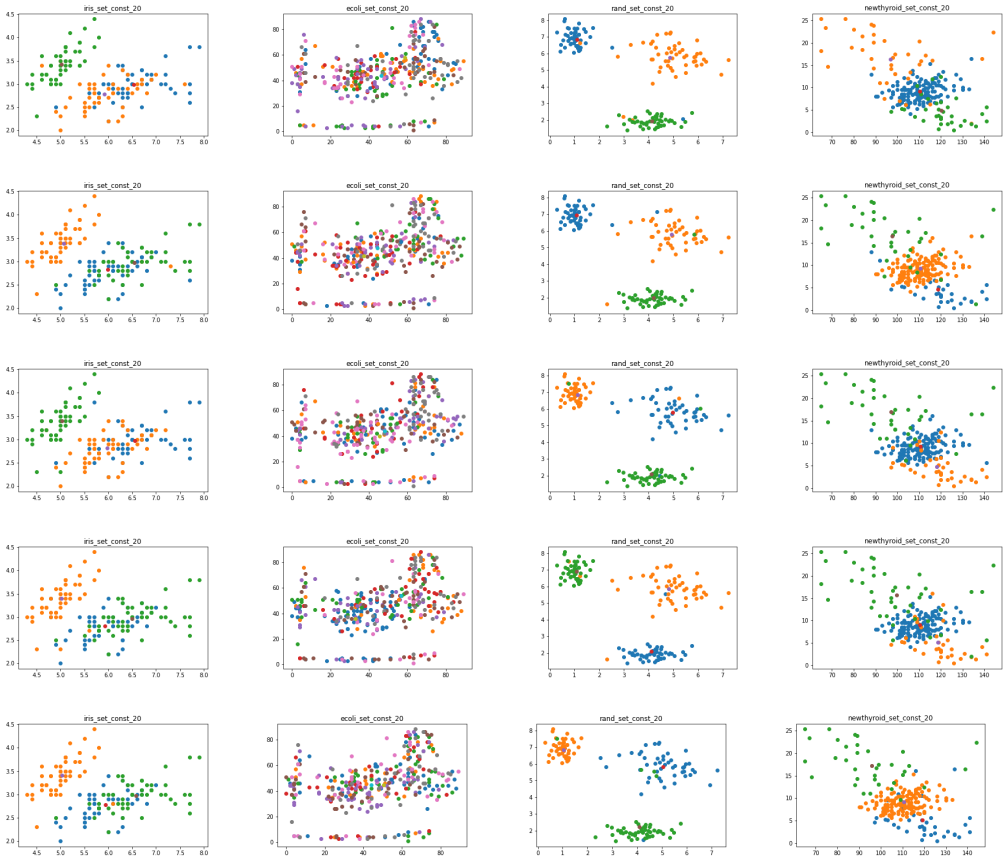

Figura 20: Primeras dimensiones de COPKM v2 para 20 % de restricciones

Referencias

[1] Numpy. `https://numpy.org/`.

[2] Scikit-learn. `https://scikit-learn.org/stable/`.

[3] Alireza Askarzadeh and Esmat Rashedi. *Harmony Search Algorithm*. 03 2017.

[4] Kai Gao, Ling Wang, Jianping Luo, Hua Jiang, Ali Sadollah, and Quan-Ke Pan. Discrete harmony search algorithm for scheduling and rescheduling the reprocessing problems in remanufacturing: a case study. *Engineering Optimization*, pages 1–17, 11 2017.

[5] Vijay Kumar, Jitender Kumar Chhabra, and Dinesh Kumar. Effect of harmony search parameters variation in clustering. *Procedia Technology*, 6:265 – 274, 2012.

[6] Kang Seok Lee and Zong Woo Geem. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 194(36):3902 – 3933, 2005.