IMPLEMENTACIÓN Y RENDIMIENTO DE ESTRATEGIAS EVOLUTIVAS BÁSICAS Y HARMONY SEARCH

APLICACIONES DEL SOFT-COMPUTING EN ENERGÍA, VOZ E IMAGEN

Álvaro Zamorano Ortega, Gabriel López Cuenca

Índice

Enunciado

Funciones a optimizar:

$$f_1(x) = \sum_{i=1}^{30} x_i^2 \quad , x_i \in [-100,100] \quad (f_{\min} = 0)$$

$$f_2(x) = \sum_{i=1}^{30} -x_i \cdot sen(\sqrt{|x_i|}), \; x_i \in [-500,500], \; (f_{\min} = -12569.5)$$

$$f_3(x) = \sum_{i=1}^{29} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2], \; x_i \in [-30,30], \; (f_{\min} = 0)$$

1. Comenzad por la implementación de la estrategia evolutiva más simple, la (1+1)-ES. Implementad a continuación la estrategia $(\lambda + \mu)$-ES.

2. Implementad una búsqueda mediante Harmony Search, con tamaño de HM= λ.

Comparad los resultados en las 3 funciones de prueba.

Estrategia evolutiva (1+1)-ES

Este algoritmo se basa en la **supervivencia** del mejor individuo.

Su funcionamiento parte de una solución a la cual vamos aplicando operadores iterativamente (**mutación** a partir de un operador gaussiano) para generar un individuo **hijo**. En el caso de que el resultado de la función objetivo del hijo sea **menor** que la del padre (para el caso de las funciones del ejercicio), el individuo hijo se cambia por el individuo padre.

```
function [allFitness,fitness,hijoMejor] = algoritmo11(vector,gmax,nFuncion,sigma)
    padre = vector;
    minimoActual = evaluarFuncion(padre, nFuncion);

    allFitness = [];
    varianza = sigma;

    for i=1:gmax
        allFitness = [allFitness; minimoActual];
        hijo = generarHijo(padre,varianza,nFuncion);
        minimoHijo = evaluarFuncion(hijo,nFuncion);
        if(minimoHijo<minimoActual) %Si el hijo es mejor que el padre, se sustituye
            minimoActual = minimoHijo;
            hijoMejor = hijo;
            padre = hijo;
            fitness = minimoActual;
        end
    end
end
```

```
function [hijo] = generarHijo(padre,varianza,funcion)
    ruido = 0 + varianza*randn(1,30);
    hijo = padre + ruido;
    hijo = comprobarHijo(hijo,funcion);
end
```

Estrategia evolutiva (μ + λ)-ES

Este algoritmo sigue una estrategia **elitista**, es decir, no se pierde al mejor de la población.

Su funcionamiento parte de un conjunto de soluciones a las cuales vamos aplicando operadores iterativamente (**mutación** a partir de un operador gaussiano) para generar un conjunto de individuos **hijos**, es decir, cada solución padre genera un conjunto concreto de individuos hijos. Si el mejor (**menor** resultado en la evaluación) de los hijos es mejor que su padre, se sustituirá por él.

```
function [allFitness,mejorFitness,solucion] = algoritmoML(vectores,gmax,nFuncion,sigma,nHijos)
    padres = vectores;

    valoresPadres = evaluarAux(padres,nFuncion);
    minimoPadres = min(valoresPadres);

    allFitness = [];
    varianza = sigma;
    for j=1:gmax
        allFitness = [allFitness;minimoPadres]; %Guardamos el fitness del mejor padre
        padres = obtenerNuevosPadres(padres,nHijos,varianza,nFuncion,valoresPadres);
        valoresPadres = evaluarAux(padres,nFuncion);
        [minimoPadres, posicionMinimo] = min(valoresPadres);
        mejorFitness = minimoPadres;
        solucion = padres{posicionMinimo}; %Mejor padre
    end
end
```

```
function [padres] = obtenerNuevosPadres(padres,nHijos,varianza,nFuncion,valoresPadres)
    nPadres = size(padres);
    for i = 1:nPadres(2)
        hijos = generarHijosAux(padres{i},nHijos,varianza,nFuncion);
        valoresHijos = evaluarAux(hijos,nFuncion);
        [valorHijoMinimo, posicionHijoMinimo] = min(valoresHijos);
        if(valorHijoMinimo<valoresPadres(i))
            padres{i} = hijos{posicionHijoMinimo};
        end
    end
end
```

Búsqueda mediante Harmony Search

Este algoritmo se basa en el cómo una orquesta de jazz **improvisa**.

Los pasos del algoritmo son los siguientes:

- **HMCR**: consiste en la generación de una nueva harmonía (hijo). Cada nota (elemento del hijo) se genera independientemente, de manera que el HMCR es una probabilidad de que dicha nota sea obtenida de la Harmony Memory (matriz de padres). Si es menor la probabilidad aleatoria que la HMCR, entonces la nota se por una nota aleatoria de la misma columna que la Harmony Memory. En caso contrario, se genera una nota aleatoria entre el dominio de la misma columna de la Harmony Memory. En nuestro caso, HMCR es **0.8,**
- **PAR**: consiste en la mutación de la nota en base a una probabilidad. En nuestro caso, la mutación parte de un operador gaussiano, su probabilidad es **0.3.**
- **RSR**: algunas notas son reemplazadas por otras de forma aleatoria con una probabilidad RSR. En nuestro caso, la probabilidad RSR es **0.01.**

Finalmente, la nueva armonía se compara con la **peor** del Harmony Memory, y si es mejor que ésta última la **sustituye**.

```
function [hijo] = generarHijoRSR(padres,varianza,nFuncion)
    hijo = [];
    tam = size(padres{1}); tamColumnas = size(padres); tamColumnas = tamColumnas(2);
    [maximo,minimo] = limitesFunciones(nFuncion);
    for i = 1:tam(2)
        prob = rand;
        if(prob<0.8)
            numAleatorioFila = randi(tamColumnas); = padres{numAleatorioFila};
            valor = fila(i);
        else
            valor = minimo + (maximo-minimo)*rand;
        end
        if(prob<0.2)
            ruido = 0 + varianza*rand; = valor + ruido;
        end
        hijo = [hijo,valor];
    end
    tamHijo = size(hijo); probCambio = rand;
    if(probCambio<0.01)
        for i=1:randi(tamHijo(2))
            posicionAleatoria = randi(tamHijo(2));
            hijo(posicionAleatoria) = rand*(maximo-minimo)+minimo;
        end
    end
end
```

```
function [allFitness,mejorFitness,solucion] = algoritmoHS(vectores,gmax,nFuncion,sigma)
    padres = vectores;
    varianza = sigma;
    valoresPadres = evaluarAux(padres,nFuncion);
    minimoPadres = min(valoresPadres);
    mejorFitness = minimoPadres;
    allFitness = [];
    for j=1:gmax
        allFitness = [allFitness; mejorFitness];
        hijo = generarHijoRSR(padres,varianza,nFuncion);
        valorHijo = evaluarFuncion(hijo, nFuncion);
        [valorPeorPadres, padre]= max(valoresPadres);
        if(valorPeorPadres>valorHijo)
            padres{padre} = hijo;
            valoresPadres = evaluarAux(padres,nFuncion);
            [minimoPadres,posMinimoPadres] = min(valoresPadres);
            mejorFitness = minimoPadres;
            solucion = padres{posMinimoPadres};
        end
    end
end
```

Resultados

El estudio de los algoritmos lo realizaremos ejecutando cada uno **30 veces** y comparándolos en base a su mejor fitness, a la media de los fitness de cada algoritmo y al tiempo de ejecución (en 30 ejecuciones).

Función 1

Algoritmo	Mejor fitness	Media fitness	Tiempo ejecución
1+1	0.7064	1.2120	0.357371 s

- Sigma = 0.2
- Iteraciones = 5000

Solución

0.0665544326431217 0.159106940385291 0.0840079285191505 0.115549569167471 -0.194227788879463 -0.100712437663754 0.0147670953469231 -0.152758653623901 -0.00745353098481005 -0.129680815974849 0.349106378183917 0.158916141778468 -0.0335824496040918 -0.0273021608837633 0.331025265963762 0.0313778460338757 -0.198036524132858 0.150996458791069 0.108821574564248 0.0820336022710629 0.0874838876779758 -0.00565654734953051 -0.212065513602559 -0.107649616170483 0.171644320710419 0.110607274132159 -0.304905714900615 -0.0676267586124232 0.126890615586144 0.0907119974719557

μ + λ	0.2337	0.4018	145.667549 s

- Sigma = 0.2
- Número de padres = 50
- Número de hijos por padre = 5
- Iteraciones = 3000

Solución

0.0917848271394600 0.135461943040777 0.137810227488488 0.129432787199406 -0.138204506001924 0.0874254546460473 0.00510788015278591 0.0244638982200433 -0.0108562669902683 -0.0362616675853551 0.0370283010397925 0.151886192743997 0.0402704115153047 -0.0571637454198524 -0.0581339866170117 0.0825399240275525 -0.0298806314239685 -0.0646822183028646 -0.0504923219120770 -0.0573186175081210 -0.0433145952248605 -0.183225580292180 -0.0309814552220990 -0.132465446742263 -0.0317133036381552 -0.149363830661291 -0.0732470542714134 -0.0124285733824752 0.0418369821166901 0.0862917657508815

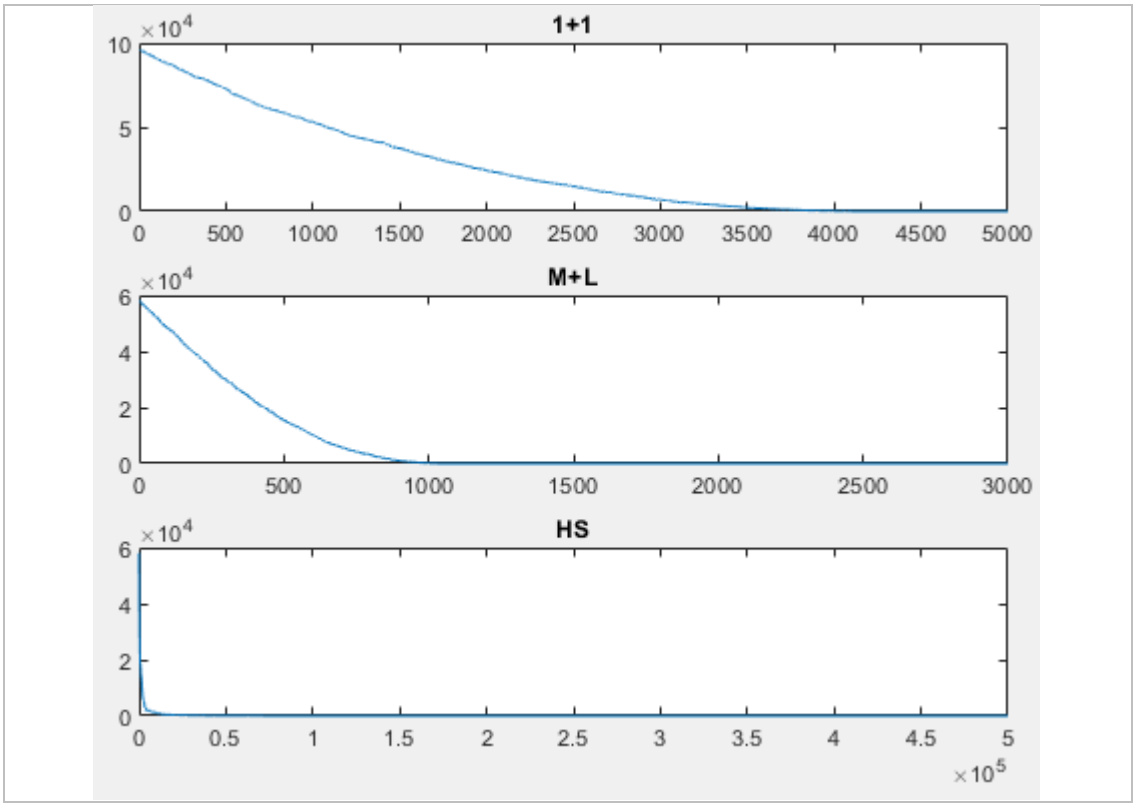
HS	0.0359	0.1006	907.073360 s

- Sigma = 0.01
- Número de padres = 50
- Iteraciones = 500000

Solución

0.0393725321740328 0.00482593223458066 0.0271939364870248 0.0195976947191326 0.0220546319707322 0.0139149755463355 0.0196871320266712 0.0364624868962714 0.0331756702702382 0.0123488842663338 -0.00878110833001666 0.00178966100650769 0.00926108954400600 0.0907302535907527 0.0218509680380238 0.0218575201397642 0.0220494756584280 0.0114803939260818 0.0389305797193765 -0.00276227267324661 0.0147218880397727 0.00562218973428665 0.0573356937994618 0.00490285466047729 0.0158383972259590 0.0528015522017463 -0.103533086833813 0.0159571125015094 0.0204810750993747 0.0105972273977566

Resultados



A la vista de los resultados se puede concluir que:

- La estrategia **1+1** necesita unas 3500 generaciones para converger, consiguiendo un fitness de 0,7.
- En el caso de **μ + λ** se converge en 1000 iteraciones siendo su mejor fitness 0,2.
- Por último, con **Harmony Search**, se necesita un gran número de iteraciones para converger, pero se obtiene un mejor resultado (mejor fitness 0,03).

Por tanto, la mejor estrategia para la función 1 es μ + λ ya que en un corto número de iteraciones se consigue un fitness adecuado. Sin embargo, con Harmony Search se consigue el mejor fitness, aunque con mucho mayor número de iteraciones.

Función 2

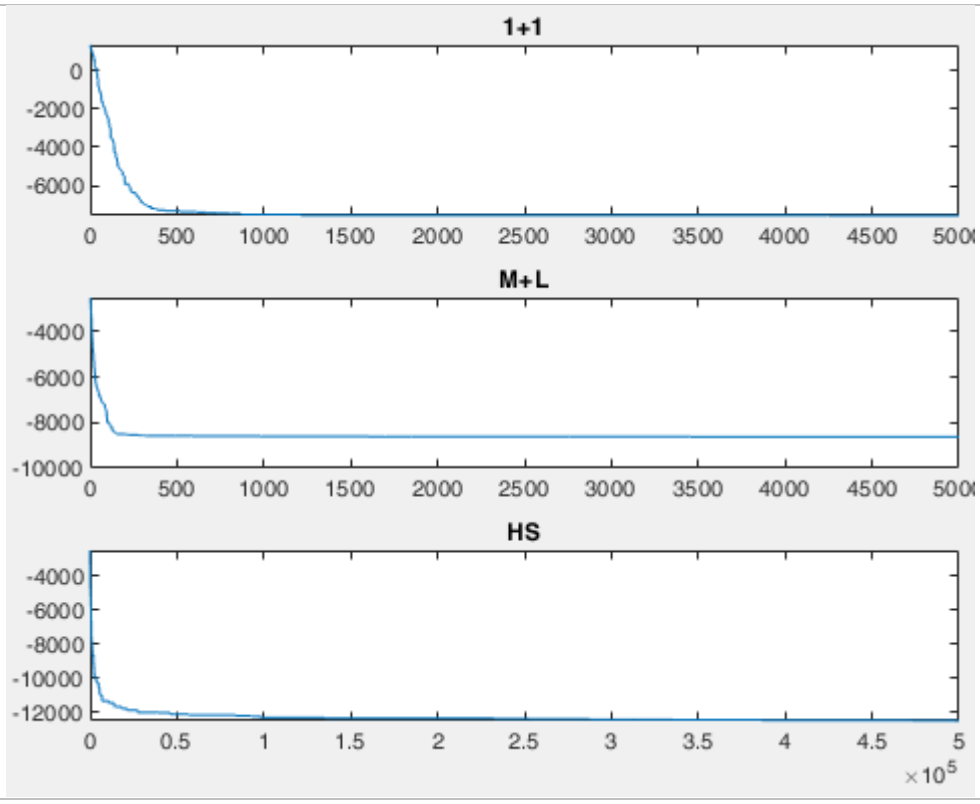
Algoritmo	Mejor fitness	Media fitness	Tiempo ejecución
1+1	-7.5635e+03	-6.9975e+03	1.541798 s
• Sigma = 5 • Iteraciones = 5000			
Solución			
60.1554533764305 -302.147922312075 -500 416.858728567724 410.572467857071 -120.959065583576 -123.562527688176 53.6829794640862 418.374230826444 207.666014468631 414.977170060992 -306.085555415499 -115.468301228431 193.504693022312 -124.824321503297 420.744895091627 420.726708851338 6.99685444490062 -295.081466635809 -500 201.049018225382 -309.404963686176 -306.453792992407 -299.908954130466 414.947541165418 -500 416.979237475675 209.473927731211 421.911414687602 -118.349483274227			
μ + λ	-8.6245e+03	-8.2384e+03	397.734137 s
• Sigma = 5 • Número de padres = 50 • Número de hijos por padre = 5 • Iteraciones = 5000			

Solución			
417.366420041429 417.759769924432 200.769396161898 423.488434818692 -300.372801716005 -500 -295.503452914693 -301.682808771212 199.739275782071 -306.588607122472 -126.247803851014 59.8581651792329 59.4857536495637 203.921299842000 415.502841507564 422.295147614466 418.811204860495 -302.971760616451 420.662021468135 421.088815087762 -124.391512870532 -500 419.282226310689 -300.967143811264 -500 423.284914976282 418.448666222064 -29.5351567686468 419.263202688335 -290.759495434092			
HS	-1.2503e+04	-1.2456e+04	955.65 s

- Sigma = 5
- Número de padres = 50
- Iteraciones = 500000

Solución
421.034784414134 426.453273064058 423.794029051142 427.439937077422 423.185764808786 417.438493069103 418.352216212684 422.862852558466 426.451510043732 425.639599419216 422.570162397645 424.424164097201 420.436282977158 423.751622986250 417.233207043550 418.017400823465 420.257749283436 422.374534006241 426.195327432428 423.726126900911 428.072497064134 423.421927796265 424.424041505829 426.188484472693 419.030951688913 427.560770319963 414.017738898708 417.265480363915 428.897669924637 425.055753817003

Resultados



A la vista de los resultados se puede concluir que:

- La estrategia **1+1** necesita unas 2000 generaciones para converger, consiguiendo un fitness de alrededor de -7563.
- En el caso de **μ + λ** se converge en 2500 iteraciones siendo su mejor fitness -8624.
- Por último, con **Harmony Search**, se necesita un gran número de iteraciones para converger, pero se obtiene un mejor resultado (mejor fitness -12503).

Por tanto, la mejor estrategia para la función 2 es Harmony Search ya que su fitness es óptimo, aunque necesita 100 veces más generaciones que las otras estrategias.

Función 3

Algoritmo	Mejor fitness	Media fitness	Tiempo ejecución
1+1	27.8317	34.8855	0.766062 s

- Sigma = 0.05
- Iteraciones = 10000

Solución

0.954354914708236 0.963730565468008 0.881836448597192 0.820058503698483 0.707689830214779
0.550548178065436 0.252156691910398 0.0189608608887349 -0.0202229620175474
0.00588836969060202 -0.0118251915500112 -0.00995295792862828 -
0.0653941468179941 0.0891525090715560 0.0815021652149698 8.39609590128736e-05
0.000866655329245450 -0.0572847883162223 0.00986612917723414 -
0.00361342100561988 3.58310484406543e-05 0.0583265932718849 -0.0419963786813847
0.0681820337206138 -0.00346353973115265 0.0309389104983075
0.0161973380861787 0.00632309645955144 0.0390927024255986 -
0.0651801263571747

μ + λ	25.1875	26.7915	159.458217 s

- Sigma = 0.05
- Número de padres = 50
- Número de hijos por padre = 5
- Iteraciones = 3000

Solución

0.980778047669939 1.02611211288598 1.02045558640896 0.986352448648526 0.978395134385911
0.923248146220712 0.855799363196166 0.709630623779873 0.496287124051024 0.242831808118867
0.0435918774507155 0.0400371342675929 -0.108597344599813 -0.0107846858770300
0.0121974728938804 0.0234701049182273 0.0589340799995347
0.0490483111381294 0.0604869523856815 0.0845370822374556 -
0.0181728731699305 0.0471340769904043 0.000568586626853336 0.0710677320859009
0.0506994504437761 0.0449617397945302 0.0179069948402289
0.105331717081392 0.0349386930387363 -0.00306018103926581

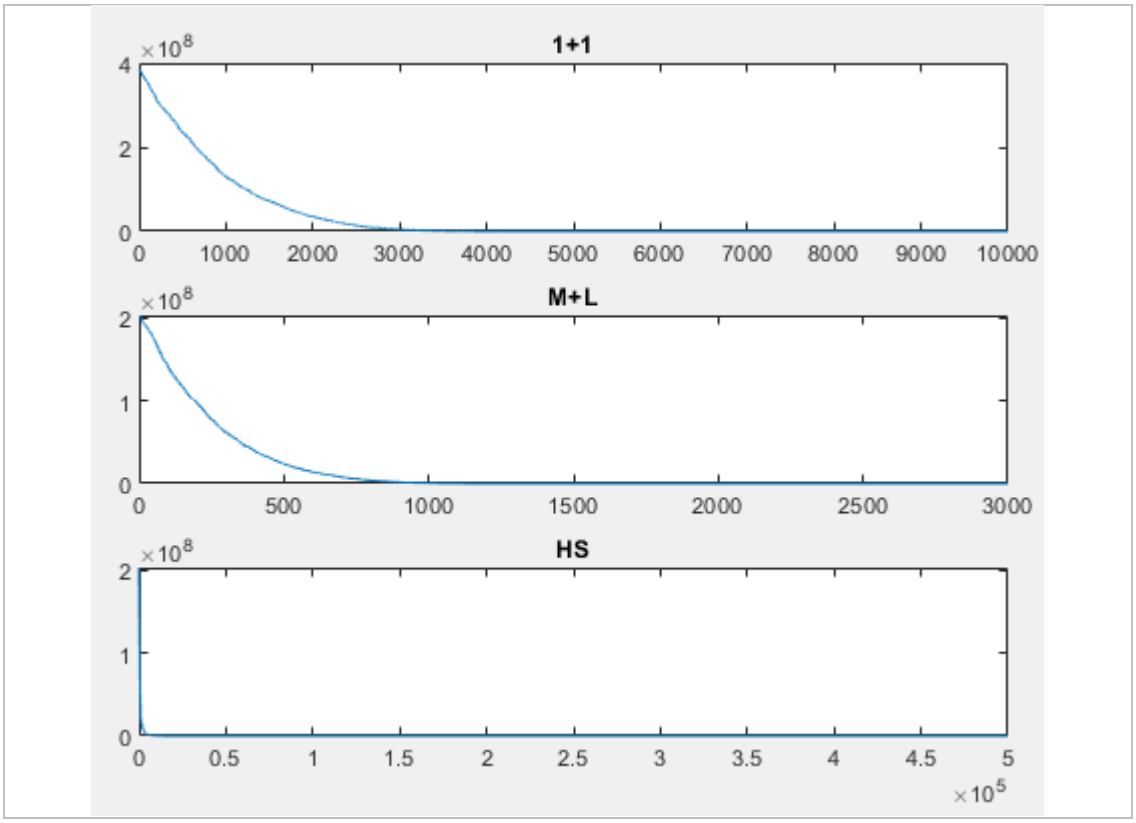
HS	5.5250	38.2326	930.257418 s

- Sigma = 0.005
- Número de padres = 50
- Iteraciones = 1000000

Solución

1.01011730129917 1.02082790810912 1.03411079867230 1.02878766770476 1.03225580420157
1.03778065967955 1.03515932439818 1.04712589501706 1.02908369515702 1.02113470781025
1.01491800131540 1.01091071562931 1.01320090044219 1.01706804352220 1.02397056082531
1.03541136887877 1.04581791825027 1.04143666412745 1.02096485240026 1.00316937284562
0.982830889518838 0.949870018210043 0.894899923219380 0.798675587164365 0.629025194413160
0.399749654586761 0.158082604132531 0.0445645380966351 0.0140268422138243
0.00642716158334595

Resultados



A la vista de los resultados se puede concluir que:

- La estrategia **1+1** necesita unas 3000 generaciones para converger, consiguiendo un fitness de 27,83.
- En el caso de **μ + λ** se converge en 1000 iteraciones siendo su mejor fitness 25,18.
- Por último, con **Harmony Search**, se necesita un gran número de iteraciones para converger, pero se obtiene un mejor resultado (mejor fitness 5,52).

El mejor fitness conseguido para esta función ha sido con **μ + λ** con 10000 iteraciones y sigma = 0.05. El resultado obtenido es:

0.00497773184755093 0.0105134097502678 0.0201331107668091 0.0320960732895464
0.0413821189063304 0.0299064839108819 0.00172641800465991 -
0.0132656898435302 -0.0137717926206546 -0.0138749516462330 -0.0109086334365759
-0.0249564868455399 0.0378198245436189 0.0214764015393114 -
0.0118098438078180 0.0216310304409008 -0.00163236522128882 -0.0111691606632729
0.0141630795919596 0.0305164349921451 -0.00282795870423269
0.00856987102980609 0.0258550244052130 0.00874347775432418 -
0.0144264330276408 0.0435231656915743 0.00212470778565953 0.0109293053974563
-0.0285616989522280 -0.0138730163746071

En este caso y en general, a pesar de que **Harmony Search** necesita un gran número de iteraciones para converger, consigue obtener los mejores resultados.