Cuckoo Search basado en vuelos Lévy
Metaheurísticas - UGR

José Carlos Martínez Velázquez - jcarlosmv@correo.ugr.es
Benjamín Vega Herrera - benjaminvh@correo.ugr.es

Julio de 2016

Contents

Introducción.

Cuckoo Search es una técnica de optimización de funciones propuesta por Xin-she Yang and Suash Deb en el año 2009. La técnica intenta modelar e imitar el comportamiento parasitario de los cucos. Los cucos son aves que ponen huevos en nidos ajenos para que sus polluelos sean criados por otras especies. Existe una probabilidad de que el ave anfitrión descubra el huevo del cuco y se deshaga de él.

La metaheurística modela este comportamiento, con movimientos aleatorios por el espacio de búsqueda (Lévy flights), probabilidad de descubrimiento (pD) y numero de nidos parasitados (nN). Estos son los parámetros básicos, pero no son los únicos. Más adelante veremos algunos parámetros avanzados que nos permitirán controlar otros conceptos. Propondremos mejoras al algoritmo y nuevos parámetros para controlarlas.

Finalmente analizaremos la técnica optimizando algunas funciones y evaluando su comportamiento.

Explicación en profundidad de la técnica.

Para analizar en profundidad la técnica tal y como la propusieron los autores en su artículo original [1], vamos a visualizar el pseudocódigo que propone:

Cuckoo Search via Lévy Flights

begin
  Objective function $f(\mathbf{x}),\ \mathbf{x} = (x_1, ..., x_d)^T$
  Generate initial population of
    n *host nests* $\mathbf{x}_i$ $(i = 1, 2, ..., n)$
  while (t <*MaxGeneration*) *or* (*stop criterion*)
    Get a cuckoo randomly by Lévy flights
      evaluate its quality/fitness F_i
    Choose a nest among n *(say,* j*) randomly*
    if $(F_i > F_j)$,
      replace j *by the new solution;*
    end
    A fraction (p_a) *of worse nests*
      are abandoned and new ones are built;
    Keep the best solutions
      (or nests with quality solutions);
    Rank the solutions and find the current best
  end while
  Postprocess results and visualization
end

Figura 1. Pseudocódigo original propuesto por Yan y Deb para Cuckoo search.

- Paso 1. Como cualquier técnica de búsqueda, Cuckoo search requiere saber la función objetivo que se pretende maximizar o minimizar.
- Paso 2. Inicialización de la población. Una población consiste en un conjunto de nidos en el que se ha puesto un huevo. Computacionalmente, un nido es una solución x_i del espacio de búsqueda, que tendrá un determinado valor asociado de la función objetivo. Cada nido de la población inicial debería ser evaluado con la función objetivo. El primer parámetro, entonces, a determinar del algoritmo es el número de nidos de que se compone la población.

- Paso 3. Evolución de la población. Vamos a evolucionar la población inicial mediante los pasos descritos a continuación. El algoritmo terminará cuando se alcancen un número máximo de generaciones, evaluaciones o cualquier otro criterio de parada que se quiera establecer.

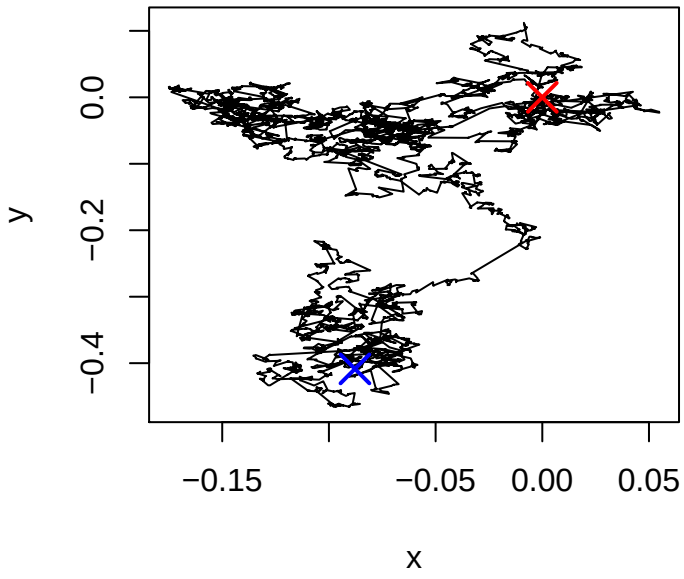
  - Paso 3.1. Obtener un nuevo nido (fuera de la población) mediante vuelos de Lévy (*Lévy flights*) y obtener su valor de fitness (valor de la función objetivo asociado a la solución x_i). Esta es la parte innovadora de la técnica, pues sin este paso, el algoritmo sería uno más. Dedicaremos un apartado sólo para explicar el modelo matemático de los vuelos de Lévy, pues para entender bien la técnica, se requiere entender muy bien en qué consiste realizar un vuelo de Lévy. Denotaremos como F_i al valor de fitness asociado a x_i.
  - Paso 3.2. Seleccionaremos aleatoriamente un nido dentro de la población actual, x_j. Denotaremos como F_j al valor fitness asociado a x_j.
  - Paso 3.3. Debemos comprobar si F_i es *mejor* que F_j (depende de si pretendemos maximizar o minimizar la función objetivo). Si es así, en la población inicial hay que reemplazar el nuevo nido encontrado (x_i) por el nido seleccionado de entre la población (x_j).
  - Paso 3.4. Para introducir diversidad, entra en juego la probabilidad de descubrimiento (pD). Un descubrimiento significa que el ave anfitriona descubre que el huevo que hay en su nido no es de su especie y, por lo tanto, lo desecha. En el algoritmo, un descubrimiento se traduce en que cada solución tiene una probabilidad pD de desaperecer de la población inicial. Cuando un nido desaparece, el cuco rellena esos huecos poniendo huevos en nuevos nidos, esto es, se generan tantas nuevas soluciones aleatorias como nidos hayan sido descubiertos.
  - Paso 3.5. Para introducir convergencia, nos quedamos con el mejor nido de la población resultante y volvemos al paso 3.1.

- Paso 4. Devolver la mejor solución encontrada de entre todas las iteraciones y su valor de fitness.

Vuelos de Lévy (Lévy Flights).

Con esta técnica, el autor pretende modelar matemáticamente el comportamiento del cuco cuando busca los nidos que parasitar. El aspecto que más influye en la búsqueda (por no decir el único) es el vuelo del cuco, es decir, la trayectoria que sigue desde un nido al siguiente. El autor se dió cuenta que el cuco sigue un comportamiento conocido, similar al de los depredadores como el tiburón, que consiste en hacer giros repentinos aleatorios. La técnica Lévy Flights es una mejora de la técnica Random Walks, que a su vez se basa en los movimientos Brownianos. Un ejemplo de este tipo de movimientos, lo podemos ver en [3], donde nos explican en qué consisten y por qué los números aleatorios que intervienen provienen de distribuciones normales de probabilidad.

Veamos un ejemplo completo de una trayectoria con vuelos de Lévy, donde el cuco pone 10000 huevos (se buscan 10000 soluciones aleatorias).



En el gráfico podemos ver desde donde parte el cuco en la iteración 0 (aspa roja) y dónde acaba las 10000 iteraciones (aspa azul). La línea negra describe la trayectoria del vuelo del cuco durante la puesta de huevos (búsqueda de soluciones).

Veamos cómo conseguir ésto matemáticamente. Supongamos que en el instante i del algoritmo, el cuco está situado en una posición P_i, donde P_i es una solución. El objetivo de un vuelo de Lévy es saber la posición dónde debe estar el cuco en el instante $j = i + 1$, que llamaremos P_j.

Para que el algoritmo nos diga a qué posición debería moverse el cuco necesitamos saber dos cosas: P_i, que como dijimos antes es la posición en la que el cuco se encuentra en el instante i y la mejor solución encontrada hasta el momento P_{best}.

Para situarnos, diremos rápidamente que la siguiente solución a una actual viene dada por la siguiente expresión:

$$P_{i+1} = P_i + \varepsilon \oplus levy(\lambda)$$

Donde ε, es una cantidad que nos permite, de algún modo, discretizar el espacio de búsqueda y controlar el tamaño de un salto de un valor a otro. El valor de ε debería variar en cada problema, pero usualemente está entre 0.001 y 0.01. La variable λ es la que describe la distribución de probabilidad de Lévy, y debe cumplirse que $1 < \lambda \leq 3$. El símbolo $\oplus$ indica que ε se multiplica a cada componente del vector generado por la distribución levy, es decir, $levy(\lambda)$ es un vector del mismo tamaño que una solución cuyos componentes están generados por una distribución de probabilidad Lévy.

Vamos a ampliar esta laxa definición paso a paso y en profundidad, describiendo un procedimiento que nos permita encontrar la posición P_j a partir de una posición P_i a través de un vuelo de Lévy.

Los aspectos comunes (que pueden ser hechos una sola vez) a cualquier posición generada son las distribuciones normales de probabilidad. Una distribución normal de probabilidad está descrita por una media (μ) y una desviación estándar ($sigma$), es decir, $\mathcal{N}(\mu, \sigma)$. El procedimiento de los vuelos de Lévy genera dos coeficientes:

- v: número aleatorio procedente de una normal $\mathcal{N}(0, 1)$
- u: número aleatorio procedente de una normal $\mathcal{N}(0, \hat{\phi})$

Donde:

$$\hat{\phi} = \left(\frac{\Gamma(1+\hat{\beta}) \cdot sin\left(\frac{\pi\hat{\beta}}{2}\right)}{\Gamma\left(\left(\frac{(1+\hat{\beta})}{2}\right) \cdot \hat{\beta} \cdot 2^{\frac{(\hat{\beta}-1)}{2}}\right)} \right)^{\frac{1}{\hat{\beta}}}$$

En esta expresión tenemos que Γ es una función de la que podemos saber más en [4], básicamente $\Gamma(n) = (n-1)!$. La constante $\hat{\beta} = \frac{3}{2}$, según el artículo original del autor.

Una vez calculadas u y v, podemos calcular el paso (ζ) como:

$$\zeta = \frac{u}{|v|^{-\lambda}}$$

Conocido el paso, ahora deberíamos computar la longitud del paso (η) así:

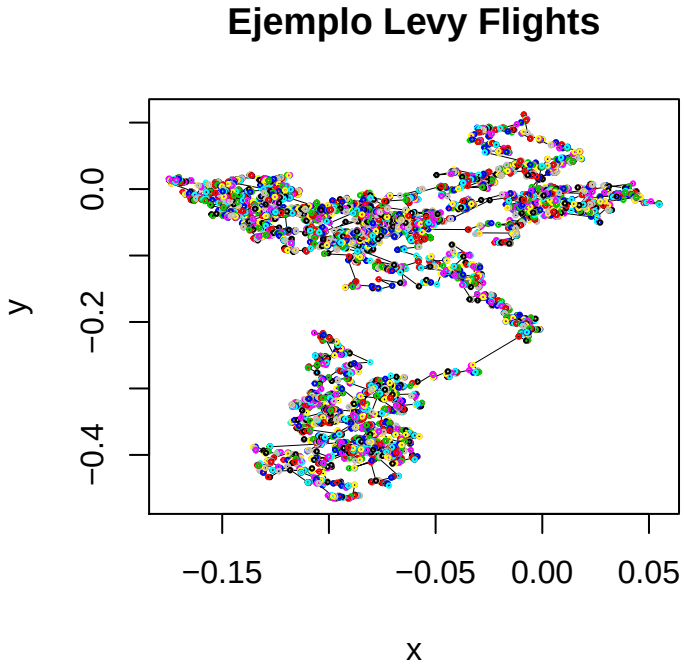
$$\eta = \varepsilon \cdot \zeta \cdot (P_i - P_{best})$$

Finalmente, podemos obtener $P_j = P_{i+1}$ como sigue:

$$P_j = P_i + \eta \cdot \Upsilon$$

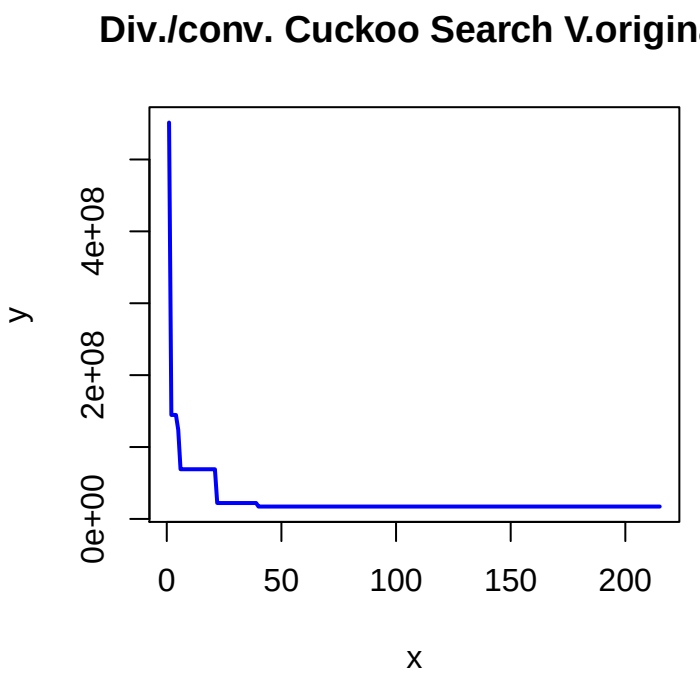
Donde Υ es un vector aleatorio del mismo tamaño que una solución que sigue una distribución de probabilidad normal $\mathcal{N}(0, 1)$.

En el ejemplo anterior, estos son todos los huevos que habría puesto el cuco (soluciones propuestas) mediante vuelos de Lévy en 10000 iteraciones.



Estudio experimental del algoritmo original

En primer lugar veamos la relación diversidad/convergencia del algoritmo. Elegiremos una función al azar de entre las 30 de la competición CEC 2014 y evaluaremos este aspecto.



Como podemos comprobar, el algoritmo no va a tener buenos resultados en la competición y estará lejos de ser competitivo, pues tiene una convergencia extremadamente rápida. Habrá que esforzarse en mejorar este aspecto.

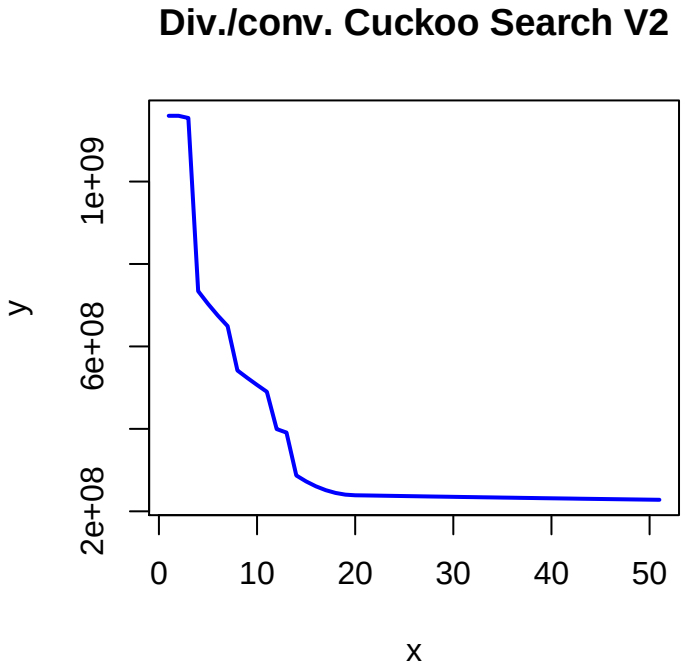
En apartados posteriores veremos cómo se desenvuelve el algoritmo original con las funciones de la competición CEC2014. Recordemos que hemos vaticinado que no va a tener un gran comportamiento debido a su rapidísima convergencia.

Mejoras realizadas

Versión 2

Vimos como el algoritmo anterior no conseguía en ocasiones buenas soluciones. Aún no pretendemos enfrentarnos al equilibrio diversidad/convergencia, sino obtener buenas soluciones.

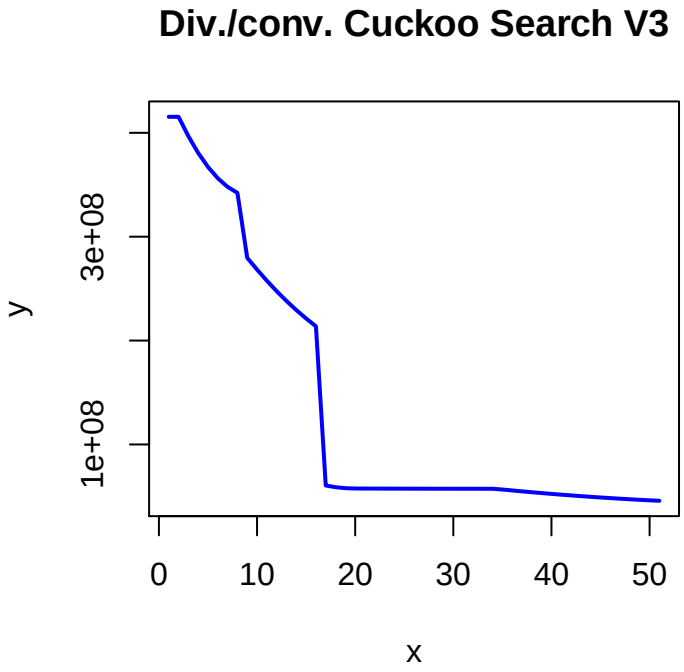
Como experiencia previa, en la última práctica de la asignatura Metaheurísticas, nos enfrentamos a tres versiones de algoritmos meméticos cuyo análisis determinó que era mejor aplicar búsqueda local al 10% de las mejores soluciones que teníamos. La búsqueda local que usaremos tiene que usar el valor de ε descrito en los vuelos de Lévy para tratar de discretizar el espacio. En este momento la búsqueda local se moverá positiva o negativamente en cada dimensión para tratar de mejorar la posición actual. Evidentemente, al ser optimización continua, necesita muchas evaluaciones para converger a un óptimo, lo que sería un suicidio teniendo un máximo de evaluaciones. Lo que haremos será aplicar búsqueda local "un poquito" al mejor individuo de la población, como máximo un 10% de las evaluaciones.



Observando la gráfica de diversidad/convergencia, vemos cómo la version 2 logra retrasar un poco la convergencia, esto es, meter un poco de diversidad pero empeora la convergencia, pues no llega tan al fondo como la versión original.

Versión 3

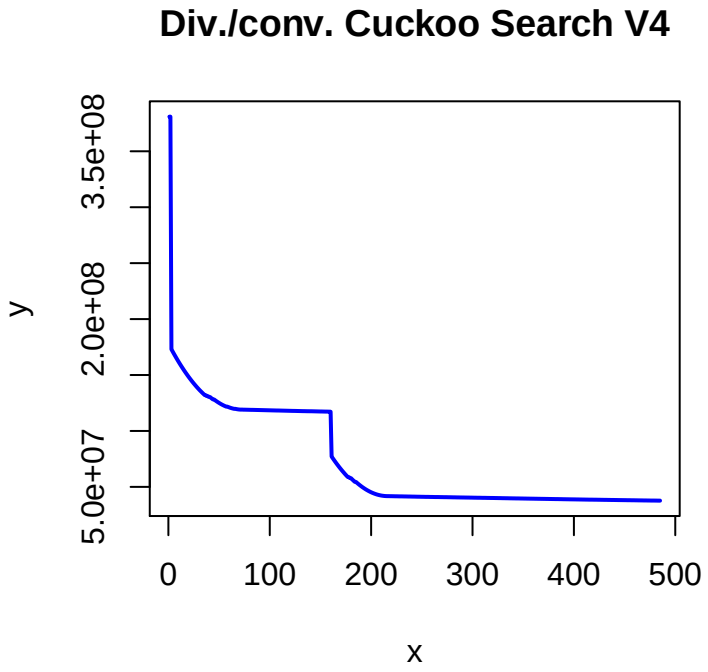
Para intentar mejorar la falta que cometía la versión 2, vamos a llevar al extremo el elitismo. La probabilidad de descubrimiento puede eliminar soluciones prometedoras (aunque sean malas), vamos a considerar prometedoras sólo a las mejores, de manera que los mejores individuos puedan repetirse. Esto, teóricamente mejoraría la convergencia, en detrimento de la diversificación.



Viendo la gráfica de diversidad/convergencia podemos ver que esta versión del algoritmo llega más al fondo que la versión 2. Ahora bien, no es oro todo lo que reluce. En la gráfica podemos apreciar cómo llega un momento en que la convergencia se produce radicalmente, esto seguramente se haya producido cuando hay un montón de individuos repetidos en la población.

Versión 4

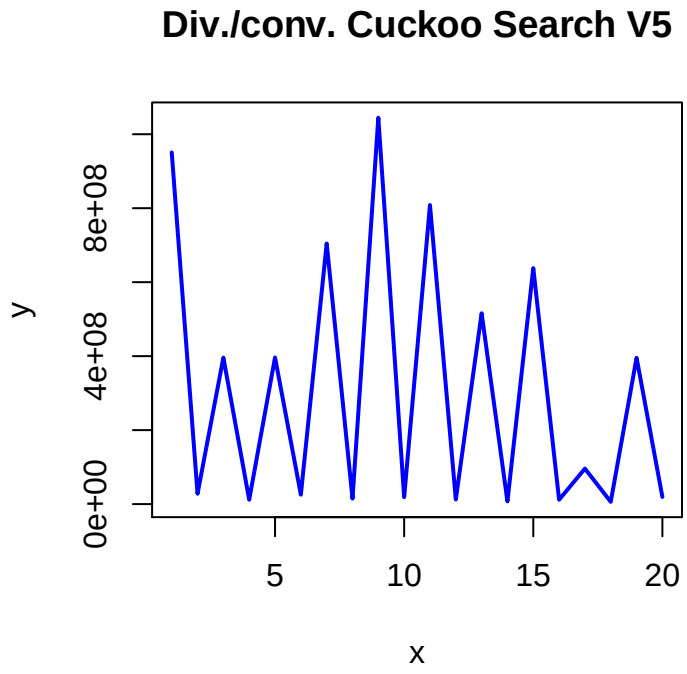
Para evitar el problema de los individuos repetidos, vamos a olvidar de la versión 3 y partiremos de la versión 2. La versión 4 va a introducir la reinicialización de la población. Para introducir convergencia, no lo haremos mediante un contador de generaciones, sino mediante una probabilidad de reinicialización. Recordemos que siempre guardamos la solución más prometedora, entonces siempre deberíamos partir de un mínimo de calidad.



En la versión 4 podemos apreciar cómo tenemos diversidad hasta casi el final de las ejecuciones, pero el algoritmo no permite empeorar, algo que define la diversidad de la población, esto es apreciable en la gráfica con esa especie de mesetas formadas, que era a lo que nos referíamos en el análisis previo con la palabra checkpoint.

Versión 5

Vamos a tratar de introducir diversidad copiando la principal idea de los algoritmos basados en enjambres de partículas, esto es, la idea de multiagente. Cada agente (cada cuco) hará su búsqueda independiente de los demás, lo que forzosamente debe introducir diversidad. Dado que las evaluaciones han de ser repartidas entre el número total de agentes, la convergencia podría verse afectada. Además de ello, vamos a modificar cuándo se realiza la búsqueda local, que se hará cada 10% del máximo de evaluaciones. Por ejemplo, si tenemos 100000 evaluaciones como máximo, la búsqueda local se aplicará al mejor individuo de la población cada 10000 evaluaciones.



En este caso vemos claramente como se ha introducido bastante diversidad, aunque curiosamente, tenemos la misma convergencia que en la versión 4. Debido a la falta de tiempo para poder extender más el estudio, concluímos que esta es la versión más potente de la que podemos disponer y competiremos humildemente con ella.

Versión 6

De nuevo, por falta de tiempo, esta versión se va a quedar en proyecto. Dado que los algoritmos genéticos tienen buen equilibrio diversidad convergencia, la idea era aplicar lo que conocemos de genéticos aquí.

Podríamos introducir un operador de selección clásico, por torneo. Un operador de cruce de soluciones, basado en corte a dos puntos, intercambiando las partes internas de los individuos. Además el operador de mutación sería cambiar el valor de una dimensión aleatoriamente a un valor en el rango con una determinada probabilidad.

En teoría, si nuestro algoritmo tiene rápida convergencia, esta modificación no tendría, en principio, mala pinta de cara a mejorar los resultados en la competición CEC2014.

VER ANEXO I.

Resultados con las funciones de CEC2014 y comparación con otros algoritmos

En este apartado vamos a poner las distintas versiones realizadas del algoritmo a optimizar las funciones de la competición CEC2014. Los resultados aquí descritos es una media aritmética obtenida de los resultados de 25 ejecuciones por cada función y versión del algoritmo. Debemos tener en cuenta que la salida esperada para cada función es el número de la función multiplicado por 100. Vamos a verlo.

Resultados con la versión original

Cuckoo Search Version Original		
	Dim 10	Dim 30
Funcion 1	7,44E+008	4,45E+009
Funcion 2	1,73E+010	1,58E+011
Funcion 3	9,98E+006	1,61E+007
Funcion 4	5250,25	52776,8
Funcion 5	521,191	521,456
Funcion 6	614,779	649,709
Funcion 7	959,847	2099,81
Funcion 8	949,093	1357,19
Funcion 9	1042,19	1574,37
Funcion 10	3624,59	10606,9
Funcion 11	3879,1	11018,5
Funcion 12	1205,7	1207,58
Funcion 13	1306,68	1312,34
Funcion 14	1466,91	1880,13
Funcion 15	646732	4,36E+007
Funcion 16	1604,6	1614,46
Funcion 17	2,97E+007	7,20E+008
Funcion 18	5,60E+008	1,36E+010
Funcion 19	2047,31	3814,91
Funcion 20	1,15E+008	8,03E+007
Funcion 21	2,75E+007	3,91E+008
Funcion 22	3108,04	226193
Funcion 23	3101,61	5046,66
Funcion 24	2663,18	2992,69
Funcion 25	2734,46	3022,48
Funcion 26	2725,19	3120,87
Funcion 27	3472,83	4900,7
Funcion 28	4871,13	12170,3
Funcion 29	5,71E+007	1,04E+009
Funcion 30	1,87E+006	2,09E+007

Figura 2. Resultados obtenidos para CuckooSearchAlgorithm.

Resultados con la versión 2

Cuckoo Search V2		
	Dim 10	Dim 30
Funcion 1	3,10E+008	4,84E+009
Funcion 2	1,60E+010	1,52E+011
Funcion 3	2,60E+006	2,67E+005
Funcion 4	4929,32	45578,9
Funcion 5	520	520,001
Funcion 6	614,672	650,733
Funcion 7	935,912	2103,61
Funcion 8	903,516	1280,89
Funcion 9	1018,43	1501,06
Funcion 10	2950,74	9323,11
Funcion 11	3172,41	9334,52
Funcion 12	1204,82	1205,63
Funcion 13	1305,98	1312,17
Funcion 14	1464,16	1861
Funcion 15	424767	4,21E+007
Funcion 16	1604,3	1613,87
Funcion 17	1,64E+007	4,97E+008
Funcion 18	4,25E+008	1,39E+010
Funcion 19	2036,77	3247,25
Funcion 20	487638	2,82E+007
Funcion 21	4,53E+006	2,08E+008
Funcion 22	3013,59	226943
Funcion 23	3006,96	4941,48
Funcion 24	2643,29	2950,14
Funcion 25	2726,66	2951,57
Funcion 26	2707,05	3005,52
Funcion 27	3485,89	4908,77
Funcion 28	4507,08	11018,4
Funcion 29	4,60E+007	7,17E+008
Funcion 30	1,33E+006	9,98E+006

Figura 3. Resultados obtenidos para CuckooSearchAlgorithmV2.

Resultados con la versión 3

Cuckoo SearchV3		
	Dim 10	Dim 30
Funcion 1	3,74E+008	4,07E+009
Funcion 2	1,56E+010	1,57E+011
Funcion 3	3,51E+006	2,54E+005
Funcion 4	4178,2	41630
Funcion 5	520,065	520,06
Funcion 6	614,236	651,702
Funcion 7	965,785	1943,05
Funcion 8	913,074	1273,25
Funcion 9	1016,78	1501,9
Funcion 10	3190,45	9365,47
Funcion 11	3269,33	9349,02
Funcion 12	1204,86	1206,31
Funcion 13	1306,17	1311,3
Funcion 14	1456,84	1898,32
Funcion 15	632830	3,25E+007
Funcion 16	1604,37	1613,83
Funcion 17	1,67E+007	4,66E+008
Funcion 18	4,96E+008	1,34E+010
Funcion 19	2054,63	3159,01
Funcion 20	2,64E+006	3,45E+007
Funcion 21	7,28E+006	2,63E+008
Funcion 22	3115,27	278621
Funcion 23	3140,43	4538,86
Funcion 24	2650,28	2955,91
Funcion 25	2722,06	2954,74
Funcion 26	2724,8	3015,58
Funcion 27	3558,68	5044,56
Funcion 28	4568,9	11360,9
Funcion 29	3,86E+007	7,16E+008
Funcion 30	644995	1,08E+007

Figura 4. Resultados obtenidos para CuckooSearchAlgorithmV3.

Resultados con la versión 4

Cuckoo SearchV4		
	Dim 10	Dim 30
Funcion 1	2,24E+008	3,30E+009
Funcion 2	1,34E+010	1,28E+011
Funcion 3	5,16E+006	246244
Funcion 4	3551,53	32155,6
Funcion 5	520	520
Funcion 6	609,956	642,289
Funcion 7	898,503	1739,34
Funcion 8	901,344	1257,95
Funcion 9	1010,17	1452,62
Funcion 10	3015,63	9182,57
Funcion 11	3152,51	9107,69
Funcion 12	1201,13	1202,79
Funcion 13	1304,98	1311,1
Funcion 14	1462,52	1813,95
Funcion 15	257187	1,87E+007
Funcion 16	1604,24	1613,87
Funcion 17	4,72E+006	2,92E+008
Funcion 18	2,61E+008	9,19E+009
Funcion 19	2005,92	2805,02
Funcion 20	7,42E+006	1,29E+007
Funcion 21	3,97E+006	1,25E+008
Funcion 22	3096,51	27490,1
Funcion 23	2962,57	4293,23
Funcion 24	2638,41	2922,82
Funcion 25	2720,38	2895,51
Funcion 26	2716,38	2856,44
Funcion 27	3107,39	3933,75
Funcion 28	4486,57	10283,3
Funcion 29	3,48E+007	6,14E+008
Funcion 30	337284	7,39E+006

Figura 5. Resultados obtenidos para CuckooSearchAlgorithmV4.

Resultados con la versión 5

Cuckoo SearchV5		
	Dim 10	Dim 30
Funcion 1	2,42E+007	7,32E+008
Funcion 2	2,94E+009	6,47E+010
Funcion 3	1,98E+004	113883
Funcion 4	678,108	10010,8
Funcion 5	520,408	520,956
Funcion 6	609,169	639,423
Funcion 7	745,698	1252,16
Funcion 8	864,929	1159,86
Funcion 9	967,157	1303,42
Funcion 10	2355,37	7927,35
Funcion 11	2597,28	8404,43
Funcion 12	1201,2	1202,57
Funcion 13	1301,89	1306,53
Funcion 14	1410,69	1595,02
Funcion 15	2305,55	1,17E+006
Funcion 16	1603,59	1613,13
Funcion 17	89982,7	1,99E+007
Funcion 18	1,87E+005	1,15E+009
Funcion 19	1908,89	2170,01
Funcion 20	7930,05	95030,5
Funcion 21	15358,3	6,30E+006
Funcion 22	2313,71	3577,8
Funcion 23	2676,85	3000,51
Funcion 24	2583,44	2803,71
Funcion 25	2693,11	2759,17
Funcion 26	2701,63	2706,67
Funcion 27	2873,32	3582,02
Funcion 28	3602,26	7550,77
Funcion 29	69697,6	1,40E+008
Funcion 30	7638,04	8,99E+005

Figura 6. Resultados obtenidos para CuckooSearchAlgorithmV5.

Los resultados nos muestran que la versión 5 es, en términos generales, la mejor. Sólo vemos que dicha versión es un poquito peor cuando la comparamos con las demás versiones (excepto con la original) en la optimización de la función 5 en todas las dimensiones. La versión 4, además gana a la versión 5 en la función 12 sólo para dimensión 10. Aunque esto ocurre, no es relevante, pues en las funciones en las que pierde la versión 5 contra sus anteriores versiones, lo hace por pocas unidades (o décimas), sin embargo, donde gana, es hasta 100 veces mejor que sus anteriores versiones, lo que, como comentábamos, convierte a la versión 5 en la mejor de la que disponemos.

Podemos considerar que la versión 5 es un éxito, pues el objetivo de mejorar la propuesta inicial del autor ha sido logrado. Como podemos comprobar, la versión 5 consigue mejores resultados en todas y cada una de las funciones para todas las dimensiones trabajadas que los conseguidos por la versión original del autor.

Comparación con otros algoritmos

Comparación con el algoritmo Differential Evolution de Daniel Molina.

Vamos a ver los resultados obtenidos contra el algoritmo Dif. Evolution de Daniel Molina.

	DEBin	DEexp	CS			
F1	0,00E+00	0,00E+00	24218900			
F2	0,00E+00	0,00E+00	2942889800			
F3	0,00E+00	0,00E+00	19512			
F4	2,16E+01	2,19E+01	278,108	**Características del DE**		
F5	2,02E+01	2,01E+01	20,408	F	0,5	
F6	5,84E-01	1,84E-01	9,169	CR	0,9	
F7	3,66E-02	3,58E-02	45,698	Popsize	60	
F8	4,35E+00	3,98E-02	64,929			
F9	1,20E+01	8,79E+00	67,157	**Diferencia entre ambos algoritmos**		
F10	5,20E+01	1,15E+01	1355,37	DEBin = Operador de mutación rand/1/bin		
F11	4,44E+02	5,55E+02	1497,28	DExp = Operador de mutación clásico rand/1/exp		
F12	4,26E-01	4,39E-01	1,2			
F13	1,14E-01	1,12E-01	1,89			
F14	1,76E-01	1,63E-01	10,69			
F15	1,77E+00	1,24E+00	805,55			
F16	2,34E+00	2,35E+00	3,59			
F17	1,64E+01	8,49E+00	88282,7			
F18	5,40E-01	5,54E-01	185276			
F19	3,11E-01	4,53E-01	8,89			
F20	2,04E-01	6,60E-02	5930,05			

Figura 7. Comparación CSV5 con DE de D. Molina en Dimension 10.

	DEBin	DEexp	CS			
F1	8,88E+04	2,95E+05	7,08E+08			
F2	2,00E+02	2,00E+02	6,17E+10			
F3	3,00E+02	3,00E+02	9,44E+04			
F4	3,85E+02	4,11E+02	9,73E+03			
F5	5,01E+02	5,00E+02	5,01E+02	**Características del DE**		
F6	6,03E+02	6,17E+02	6,30E+02	F	0,5	
F7	7,00E+02	7,00E+02	1,21E+03	CR	0,9	
F8	8,10E+02	8,00E+02	1,09E+03	Popsize	60	
F9	1,02E+03	9,73E+02	1,24E+03			
F10	1,43E+03	9,93E+02	6,57E+03	**Diferencia entre ambos algoritmos**		
F11	6,76E+03	4,31E+03	6,91E+03	DEBin = Operador de mutación rand/1/bin		
F12	1,20E+03	1,20E+03	1,20E+03	DExp = Operador de mutación clásico rand/1/exp		
F13	1,30E+03	1,30E+03	1,30E+03			
F14	1,40E+03	1,40E+03	1,58E+03			
F15	1,51E+03	1,51E+03	1,17E+06			
F16	1,61E+03	1,61E+03	1,61E+03			
F17	4,95E+03	3,93E+03	1,98E+07			
F18	1,82E+03	1,84E+03	1,15E+09			
F19	1,90E+03	1,91E+03	2,16E+03			
F20	2,01E+03	2,03E+03	8,91E+04			

Figura 8. Comparación CSV5 con DE de D. Molina en Dimension 30.

Lamentablemente, la mejor versión de Cuckoo Search implementada por nosotros no consigue ganarle al algoritmo DE de Daniel Molina. En algún caso nos quedamos cerca (función 5), pero en la mayoría hay diferencias significativas. El algoritmo necesita seguir mejorando para poder ser competitivo.

Comparación con los algoritmos participantes en la competición CEC2014.

Vamos a ver los resultados obtenidos contra todos los algoritmos participantes en CEC2014.

	CMLSP	FCDE	FERDE	FWA-DM	GaAPADE	L-SHADE	MVMO	NRGA	OptBees	OBL_ADI	RSDE	SOO	O+BOBY	UMOEAS	3e3pbes	nalschcm	CS
F1	1,8E-007	1150613	5234091	5013,049	0	0	0,000495	27904,5	784,1906	16226,57	0	8810740	4569,72	0	2626512	0	2E+007
F2	1,1E-015	2E+007	1E+008	0,000134	0	0	7,1E-009	914,6605	0,009883	2273,055	0	6,343	0,036	0	2E+008	0	3E+009
F3	0,000106	1234,988	4545,647	1,9E-009	0	0	9,9E-011	1516,806	0,921307	0,000574	0	6643,67	5842,92	0	4305,4	1,0E-007	19512
F4	3,3E-015	41,97763	48,45589	1,413235	30,68824	29,40955	9,545624	15,43558	2,690817	25,50826	2,81097	0,678	0	0	52,53329	0,085008	278,108
F5	16,86305	20,38114	20,09145	20,02724	19,6767	14,1456	16,58058	19,60688	20	19,08704	19,21631	20	20	16,831	20,21299	13,65196	20,408
F6	0,06201	5,327998	3,473249	0,706304	0,148376	0,01754	0,003445	2,449826	3,016623	1,039458	0,052908	0,002	0,002	0	2,790775	0,000148	9,169
F7	0	0,822638	2,193035	0,094799	0,003163	0,003043	0,018584	0,20303	0,156159	0,162686	0,035496	0,049	0,049	0	3,306893	0	45,698
F8	2,070678	14,22719	9,76986	0,253617	0	0	6,7E-015	5,584734	1,2E-013	7,808883	0,660805	18,904	18,904	0	13,24861	0	64,929
F9	1,658501	35,79322	18,74572	6,008483	3,379005	2,344598	3,492111	8,693676	20,83557	7,630765	8,522406	8,955	8,955	2,725476	21,03687	3,31653	67,157
F10	196,1156	488,5916	151,5188	1,592692	0,151789	0,008572	2,136919	119,43	219,2256	153,4006	68,44158	130,39	130,39	0,37386	330,8471	7,677946	1355,37
F11	152,9958	976,2718	537,6227	372,2404	183,1153	32,05583	96,27604	575,9476	392,748	208,2013	290,6419	349,05	349,05	144,0443	782,9861	20,13497	1497,28
F12	0,030265	1,128525	0,629776	0,042495	0,14022	0,068167	0,042228	0,124164	0,130399	0,269454	0,220648	0	0	0	0,803077	0,016465	1,2
F13	0,02725	0,305439	0,335213	0,120614	0,060087	0,051562	0,035533	0,157686	0,416158	0,131173	0,127667	0,03	0,03	0,009436	0,31249	0,032923	1,89
F14	0,189165	0,338071	1,007054	0,213906	0,09424	0,081362	0,089059	0,253702	0,368651	0,260272	0,135988	0,13	0,13	0,110034	0,688413	0,12649	10,69
F15	0,896629	5,766517	19,13657	0,774837	0,605651	0,366099	0,434601	1,021838	2,438871	0,711842	0,983009	0,44	0,42	0,666698	47,01128	0,471494	805,55
F16	1,554589	3,50591	2,922204	1,75707	1,977198	1,240797	1,448515	2,746937	2,639589	1,409093	2,233398	2,52	2,52	1,530246	2,826145	1,054166	3,59
F17	312,7451	1765,618	314732	254,5371	9,914349	0,976664	9,356666	16074,91	684,3999	257,2398	47,70168	3122910	422,57	8,476833	101237	78,33846	88282,7
F18	30,85294	309,4347	234240	25,15813	0,222974	0,244093	0,7826	7419,754	33,5043	33,16152	1,996367	12932,1	3951,62	0,784029	225413	5,220721	185276
F19	1,25112	1,614197	2,032065	1,299117	0,256564	0,0773	0,15834	2,093335	0,933042	2,087875	1,030208	0,55	0,55	0,199971	2,374421	0,076607	8,89
F20	19,94066	235,2392	407578	13,37053	0,431554	0,184883	0,312556	1719,184	8,957556	12,58948	0,721467	9364,2	6925,1	0,370588	5688,917	8,056691	5930,05
F21	36,39071	1664,77	83860,17	94,64253	0,508552	0,408069	1,934721	4823,427	57,05615	102,5059	1,208642	24694,9	1940,39	0,540407	5004,723	49,28617	13258,3
F22	89,52842	26,10666	43,77415	34,08745	3,247398	0,044103	0,262886	37,56658	17,02468	30,02275	11,6549	126,46	126,47	0,244755	67,56668	8,474625	113,71
F23	201,8202	329,9485	332,5387	329,4575	329,46	329,4575	329,4575	329,4575	272,3515	329,4575	329,4575	200	200	329,4575	333,9172	329,4575	376,85
F24	109,9103	145,5083	126,9949	127,3903	108,9051	107,4896	109,2258	130,7641	137,378	123,8384	119,133	115,65	115,65	108,2987	130,9645	108,443	183,44
F25	127,5298	186,1197	156,8479	178,7233	163,6245	132,7388	116,126	183,6782	145,9907	186,1996	129,5116	145,16	139,08	126,0032	185,6212	175,0708	193,11
F26	100,0194	100,3135	100,3318	100,1384	100,0688	100,05	100,0323	100,1366	100,3964	100,1207	100,1291	100,05	100,05	100,014	100,3284	100,0364	101,63
F27	41,13156	173,6789	135,3908	321,2754	89,68693	58,06393	17,20188	280,7776	7,422917	255,937	91,2492	200	200	25,477	158,9575	184,7796	173,32
F28	280,3234	508,0455	435,8938	347,1652	383,2116	380,8105	361,0548	477,1474	306,6715	423,3538	386,9035	200	200	312,9189	420,1244	388,7168	802,26
F29	200	367,5578	83243,14	211,7373	222,3229	221,9867	181,4021	413,291	219,9635	355435	212,6059	200	200	195,4616	16758,58	227,0654	66797,6
F30	216,4126	1390,678	2917,534	394,2959	467,1992	464,8828	491,7453	1727,538	389,1652	637,9676	505,2479	200	200	233,8867	1320,655	585,1143	4638,04

Figura 9. Ranking CSV5 en CEC2014 Dimension 10.

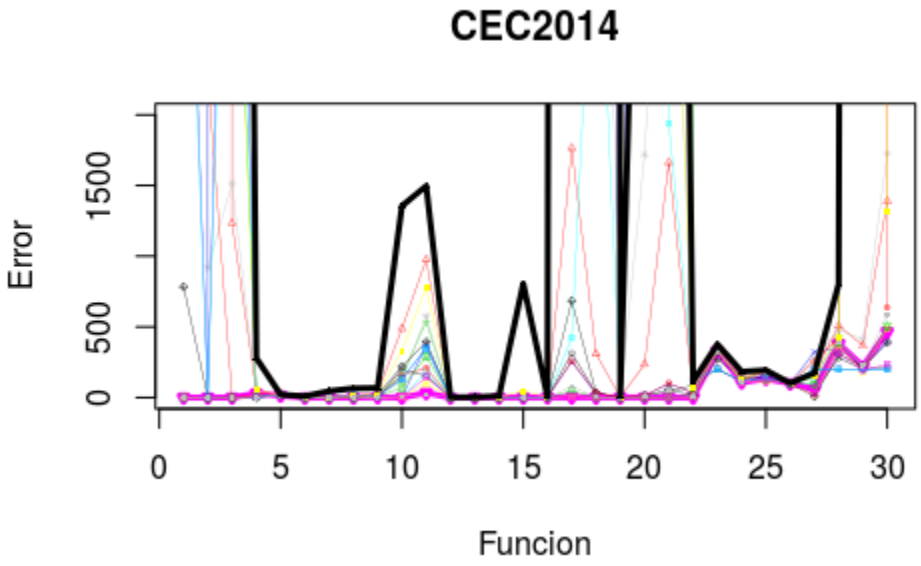


Figura 10. Comparación gráfica de los algoritmos en dimensión 10.

En dimensión 10, el algoritmo gana a algunas funciones, aunque en términos generales es una cota superior de todos los algoritmos. La línea negra gruesa es nuestro algoritmo y el de color rosa (grueso) es el algoritmo que ganó la competición: L-SHADE. Vamos a analizar en qué posición nos quedaríamos si estuviéramos compitiendo realmente.

Función	Posición
F1	17
F2	17
F3	17
F4	17
F5	17
F6	17

Función	Posición
F7	17
F8	17
F9	17
F10	17
F11	17
F12	17
F13	17
F14	17
F15	17
F16	15
F17	15
F18	17
F19	17
F20	14
F21	15
F22	15
F23	17
F24	17
F25	17
F26	17
F27	10
F28	15
F29	17
F30	17

Posición media: $(23 \cdot 17 + 5 \cdot 15 + 14 + 10)/30 = 16,33$. Hemos quedado últimos en la mayoría de las ocasiones, por ende, quedamos últimos en la competición.

Vamos a ver ahora qué ocurre en dimensión 30.

	CMLSP	FCDE	FERDE	FWA-DM	GaAPADE	L-SHADE	MVMO	NRGA	OptBees	OBL_ADI	RSDE	SOO	UMOEAS	CS
F1	4,0E-010	4E+007	7E+007	276357	1,0E-014	0	0,001066	574302	85681,08	15954,83	1500,289	2E+008	0	7E+008
F2	0	2E+009	3E+009	1,1E-016	1,7E-015	0	2,4E-005	9276,281	3,3E-012	313,6771	1,2E-009	31387	0	6E+010
F3	1,2E-008	9119,318	13921,84	4,4E-016	2,2E-015	0	0,001106	4581,834	0,008414	6,4E-010	0,047433	10810,2	0	94371
F4	2,2E-006	196,6229	489,4935	20,36249	2,8E-012	0	4,4E-013	80,58102	12,56348	63,44831	3,05179	109,346	0	9732,69
F5	19,99896	20,95678	20,19698	20,50597	20,00229	20,11468	19,99956	20,00014	20	20,6377	20,33411	20	20	500,548
F6	0	23,29159	24,03274	12,86363	0,613421	1,4E-007	3,620226	17,82335	16,3762	5,191192	5,160633	1,897	1,907	630,254
F7	0	25,56764	31,10055	0,008546	2,2E-015	0	0,00299	0,015894	0,037456	0,023727	0,000846	0,996	0,409	1206,46
F8	9,837491	119,9245	46,40362	1,1E-013	1,746266	0	0,858412	26,58988	3,6E-013	55,86554	20,41341	92,531	92,531	1094,93
F9	2,18526	201,6079	122,0642	56,61793	16,99616	6,784876	25,12758	45,69018	137,1473	84,63108	57,95042	59,706	59,697	1236,26
F10	1469,479	3548,67	1339,702	8,525867	8,143284	0,016329	17,86361	1073,473	1041,298	2167,675	329,1781	2312,38	2131,47	6571,98
F11	1822,477	5314,85	3551,51	2629,753	1896,767	1229,479	1541,69	3405,553	2716,635	3858,27	2737,263	2151,25	2091,05	6907,15
F12	0,000146	1,9111	1,35218	0,371251	0,203389	0,160577	0,072055	0,150542	0,181246	0,950506	0,443941	0,03	0,03	1201,37
F13	0,048135	0,741534	0,845991	0,38862	0,144886	0,124117	0,157294	0,28111	0,560833	0,285834	0,305364	0,35	0,34	1304,64
F14	0,311895	3,212646	12,36815	0,268582	0,211286	0,241702	0,198856	0,186616	0,399547	0,225915	0,236281	0,29	0,28	1584,33
F15	3,020714	4496,495	41716,55	7,373282	3,060288	2,14637	2,855227	13,7297	12,71035	7,733778	5,92202	22,51	21,69	1172354
F16	12,78485	12,86934	12,14822	10,9783	9,880971	8,499015	10,20949	11,46696	10,90813	10,44072	10,60218	9,86	9,81	1609,54
F17	934,8205	705470	7040904	6285,723	199,7495	187,5081	900,8198	234648	27401,73	1103,692	1239,143	3E+007	42148,7	2E+007
F18	75,44929	2238462	1E+008	76,66202	9,324016	5,910067	28,93824	550,4614	195,5904	109,7573	95,4482	2854,99	41,58	1E+009
F19	3,877261	25,99335	47,70424	9,94723	3,620694	3,681779	3,079965	13,73975	7,898369	8,881846	5,652966	183,62	16,3	2161,12
F20	9,79078	5091,055	10330,37	42,7794	5,585408	3,081863	109,1572	11378	852,6656	38,90711	37,30235	38149,6	34381,2	89100,5
F21	229,6321	55807,81	3113705	729,4119	118,2128	86,8329	466,617	180671	17431,9	386,3059	470,9285	2E+007	15435	6285442
F22	60,94744	389,7187	456,6456	145,653	62,6329	27,6009	144,5745	407,1396	232,0181	230,5926	191,3664	1019,94	956,48	3464,09
F23	200	326,0803	340,83	314,0129	315,24	315,2441	315,2441	315,2454	314,7704	315,2441	315,2441	200	200	2623,66
F24	200	250,5472	245,6888	226,0679	224,2922	224,0462	224,7538	228,4003	236,155	221,5734	224,3782	200	200	2620,27
F25	200	209,0371	211,1376	200,5653	202,5916	202,6063	203,2951	210,5359	200,975	203,6276	202,727	200	200	2566,06
F26	123,6993	100,8767	100,9757	100,3497	100,1431	100,1136	100,1622	100,3635	100,551	139,4971	100,337	200	200	2605,04
F27	200	891,4109	582,5453	401,0319	327,6451	300	401,132	587,6208	402,4	420,5095	468,5444	200	200	3408,7
F28	200	1106,042	1328,12	393,0306	833,5894	840,3043	876,7701	1593,633	431,3996	916,3845	905,0781	200	200	6748,51
F29	200	315920	4349024	211,0373	500072	716,8803	736,0847	1320,599	215,901	339290	651806	200	200	1E+008
F30	200	9424,439	100494	451,3509	1646,424	1246,381	2001,147	2889,442	592,8338	1292	1695,621	200	200	894428

Figura 11. Ranking CSV5 en CEC2014 Dimension 30.

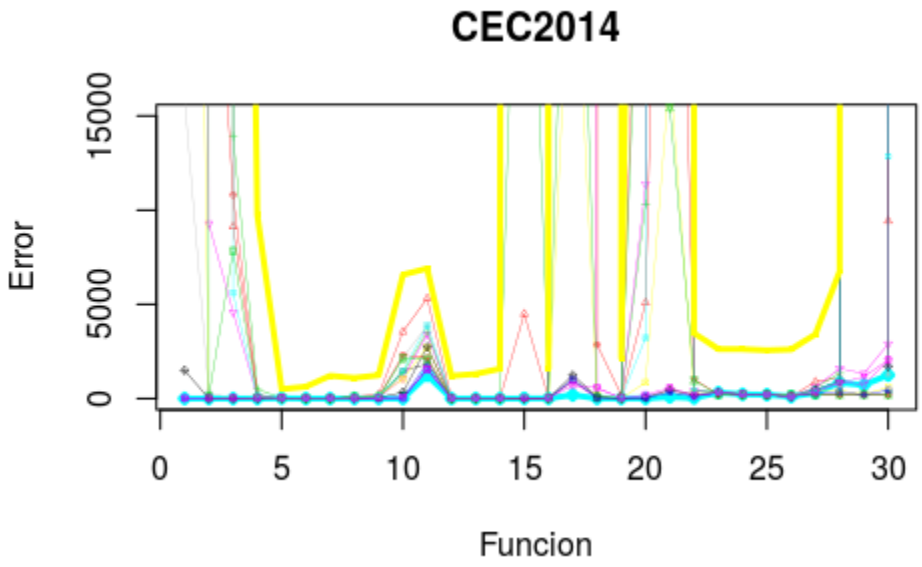


Figura 12. Comparación gráfica de los algoritmos en dimensión 30.

En dimensión 30, el algoritmo es un desastre en comparación con los demás participantes en la competición, como podemos ver. Sólo en dos ocasiones batimos al peor que participó en la competición. En color amarillo tenemos nuestro algoritmo y en color azul grueso el ganador L-SHADE.

Función	Posición
F1	17
F2	17
F3	17
F4	17
F5	17
F6	17
F7	17
F8	17
F9	17
F10	17
F11	17
F12	17
F13	17
F14	17
F15	17
F16	17
F17	16
F18	17
F19	17
F20	17
F21	16
F22	17
F23	17
F24	17
F25	17
F26	17

Función	Posición
F27	17
F28	17
F29	17
F30	17

Posición media: $(28 \cdot 17 + 2 \cdot 16)/30 = 16,93$. Participando en esta dimensión, hubiéramos quedado los últimos.

Conclusiones

El algoritmo Cuckoo Search posee una rápida convergencia, lo que supone un inconveniente de cara a salir de óptimos locales. A pesar de los esfuerzos realizados, no hemos sido capaces de hacer cosas importantes en la competición CEC2014, no obstante, hemos conseguido mejorar totalmente el algoritmo propuesto por el autor, pues la versión que pusimos a competir batía en todas las funciones y dimensiones a la versión original. Lamentablemente, dicha versión está en los últimos puestos de la competición y a años luz del ganador. Debemos seguir mejorando. Siempre.

Referencias

[1] Cuckoo search via Lévy flights- Yang, Deb - 2009

http://www.cs.tufts.edu/comp/150GA/homeworks/hw3/_reading7%20Cuckoo%20search.pdf

[2] Akemi Gálvez, Andrés Iglesias, and Luis Cabellos, "Cuckoo Search with Lévy Flights for Weighted Bayesian Energy Functional Optimization in Global-Support Curve Data Fitting," The Scientific World Journal, vol. 2014, Article ID 138760, 11 pages, 2014. doi:10.1155/2014/138760

http://www.hindawi.com/journals/tswj/2014/138760/#EEq8

[3] Brownian Motion Animation - mathslug

https://www.youtube.com/watch?v=pdz7wFHSLD0

[4] Función gamma - Wikipedia

https://es.wikipedia.org/wiki/Funci%C3%B3n_gamma

ANEXO I

Visto que por falta de tiempo no hemos podido incluir esta versión (versión 6) en la memoria, incluimos como anexo lo último que hemos podido mejorar con esta versión.

Como hemos visto antes la versión 6 incluye cruze y mutación como en geneticos para dar diversificación al algoritmo. Hemos probado varias subversiones: 1. Cruzando todos los individuos de la población, por el operador de cruce .... 2. Cruzando un porcentaje aleatorio de la población, por el operador de cruce . 3. Cruzando los 3 peores de la población con otros elegidos aleatoriamente, por el operador de cruce ... La mutación a sido en cada caso con una probabilidad del 0.01. Tras analizar los resultados obtenidos la mejor subversión ha sido la 1 y en todas la mutación no era relevante por lo que en la versión final no está incluida.

Otra mejora que hemos tenido que hacer es que al es decrementar la frecuencia de uso de la búsqueda local, pasamos de un 10% de las evaluaciones totales al 50%.

Resultados con la versión 6

Cuckoo SearchV6		
	Dim 10	Dim 30
Funcion 1	2,24E+007	7,82E+008
Funcion 2	2,50E+009	6,15E+010
Funcion 3	1,84E+004	122936
Funcion 4	654,091	9382,61
Funcion 5	520,373	520,967
Funcion 6	609,007	639,454
Funcion 7	740,055	1282,64
Funcion 8	861,492	1155,9
Funcion 9	966,132	1311,35
Funcion 10	2375,68	8005,15
Funcion 11	2574,89	8351,33
Funcion 12	1201,23	1202,57
Funcion 13	1301,94	1306,57
Funcion 14	1411,5	1593,28
Funcion 15	2062,12	1,10E+006
Funcion 16	1603,58	1613,1
Funcion 17	96951,5	2,03E+007
Funcion 18	2,28E+005	1,18E+009
Funcion 19	1909,01	2188,32
Funcion 20	6335,42	75036
Funcion 21	14791,6	5,09E+006
Funcion 22	2309,91	3556,1
Funcion 23	2666,69	3007,73
Funcion 24	2582,76	2802,7
Funcion 25	2692,22	2763,41
Funcion 26	2701,37	2706,48
Funcion 27	2851,64	3570,36
Funcion 28	3567,82	7377,43
Funcion 29	110017	1,37E+008
Funcion 30	7193,6	8,73E+005

Figura 13. Resultados obtenidos para CuckooSearchAlgorithmV6.

Cuckoo SearchV5			Cuckoo SearchV6		
	Dim 10	Dim 30		Dim 10	Dim 30
Funcion 1	2,42E+007	7,32E+008	Funcion 1	2,24E+007	7,82E+008
Funcion 2	2,94E+009	6,47E+010	Funcion 2	2,50E+009	6,15E+010
Funcion 3	1,98E+004	113883	Funcion 3	1,84E+004	122936
Funcion 4	678,108	10010,8	Funcion 4	654,091	9382,61
Funcion 5	520,408	520,956	Funcion 5	520,373	520,967
Funcion 6	609,169	639,423	Funcion 6	609,007	639,454
Funcion 7	745,698	1252,16	Funcion 7	740,055	1282,64
Funcion 8	864,929	1159,86	Funcion 8	861,492	1155,9
Funcion 9	967,157	1303,42	Funcion 9	966,132	1311,35
Funcion 10	2355,37	7927,35	Funcion 10	2375,68	8005,15
Funcion 11	2597,28	8404,43	Funcion 11	2574,89	8351,33
Funcion 12	1201,2	1202,57	Funcion 12	1201,23	1202,57
Funcion 13	1301,89	1306,53	Funcion 13	1301,94	1306,57
Funcion 14	1410,69	1595,02	Funcion 14	1411,5	1593,28
Funcion 15	2305,55	1,17E+006	Funcion 15	2062,12	1,10E+006
Funcion 16	1603,59	1613,13	Funcion 16	1603,58	1613,1
Funcion 17	89982,7	1,99E+007	Funcion 17	96951,5	2,03E+007
Funcion 18	1,87E+005	1,15E+009	Funcion 18	2,28E+005	1,18E+009
Funcion 19	1908,89	2170,01	Funcion 19	1909,01	2188,32
Funcion 20	7930,05	95030,5	Funcion 20	6335,42	75036
Funcion 21	15358,3	6,30E+006	Funcion 21	14791,6	5,09E+006
Funcion 22	2313,71	3577,8	Funcion 22	2309,91	3556,1
Funcion 23	2676,85	3000,51	Funcion 23	2666,69	3007,73
Funcion 24	2583,44	2803,71	Funcion 24	2582,76	2802,7
Funcion 25	2693,11	2759,17	Funcion 25	2692,22	2763,41
Funcion 26	2701,63	2706,67	Funcion 26	2701,37	2706,48
Funcion 27	2873,32	3582,02	Funcion 27	2851,64	3570,36
Funcion 28	3602,26	7550,77	Funcion 28	3567,82	7377,43
Funcion 29	69697,6	1,40E+008	Funcion 29	110017	1,37E+008
Funcion 30	7638,04	8,99E+005	Funcion 30	7193,6	8,73E+005

Figura 14. Comparación entre v5 y v6.

Los resultados nos muestran que la versión 6 es algo mejor que la v5. Sólo vemos que dicha versión es un poquito peor cuando la comparamos con la versión 5 en la optimización de las funciónes 10,12,13,14,17,18 y 29 en la dimensión 10 y sólo las funciones 10,13 y 29 en dimensión 30. En todas estas funciones no mejora a la v5 pero en las demás las mejora no demasiado pero podemos subir un poco en la competición, del puesto 16,33 que teníamos al puesto 16,26 en dimensión 10, en dimensión 30 nos quedamos igual.

Veamos resultados en la competición

Comparación con el algoritmo Differential Evolution de Daniel Molina.

Vamos a ver los resultados obtenidos contra el algoritmo Dif. Evolution de Daniel Molina.

	DEBin	DEexp	CS			
F1	0,00E+00	0,00E+00	22422100			
F2	0,00E+00	0,00E+00	2502049800			
F3	0,00E+00	0,00E+00	18092			
F4	2,16E+01	2,19E+01	254,091	**Características del DE**		
F5	2,02E+01	2,01E+01	20,373	F	0,5	
F6	5,84E-01	1,84E-01	9,007	CR	0,9	
F7	3,66E-02	3,58E-02	40,055	Popsize	60	
F8	4,35E+00	3,98E-02	61,492			
F9	1,20E+01	8,79E+00	66,132	**Diferencia entre ambos algoritmos**		
F10	5,20E+01	1,15E+01	1375,68	DEBin = Operador de mutación rand/1/bin		
F11	4,44E+02	5,55E+02	1474,89	DExp = Operador de mutación clásico rand/1/exp		
F12	4,26E-01	4,39E-01	1,23			
F13	1,14E-01	1,12E-01	1,94			
F14	1,76E-01	1,63E-01	11,5			
F15	1,77E+00	1,24E+00	562,12			
F16	2,34E+00	2,35E+00	3,58			
F17	1,64E+01	8,49E+00	95251,5			
F18	5,40E-01	5,54E-01	225993			
F19	3,11E-01	4,53E-01	9,01			
F20	2,04E-01	6,60E-02	4335,42			

Figura 14. Comparación CSV6 con DE de D. Molina en Dimension 10.

	DEBin	DEexp	CS			
F1	8,88E+04	2,95E+05	7,60E+08			
F2	2,00E+02	2,00E+02	5,90E+10			
F3	3,00E+02	3,00E+02	1,05E+05			
F4	3,85E+02	4,11E+02	9,13E+03			
F5	5,01E+02	5,00E+02	5,01E+02	**Características del DE**		
F6	6,03E+02	6,17E+02	6,30E+02	F	0,5	
F7	7,00E+02	7,00E+02	1,24E+03	CR	0,9	
F8	8,10E+02	8,00E+02	1,09E+03	Popsize	60	
F9	1,02E+03	9,73E+02	1,25E+03			
F10	1,43E+03	9,93E+02	6,63E+03	**Diferencia entre ambos algoritmos**		
F11	6,76E+03	4,31E+03	6,88E+03	DEBin = Operador de mutación rand/1/bin		
F12	1,20E+03	1,20E+03	1,20E+03	DExp = Operador de mutación clásico rand/1/exp		
F13	1,30E+03	1,30E+03	1,30E+03			
F14	1,40E+03	1,40E+03	1,58E+03			
F15	1,51E+03	1,51E+03	1,10E+06			
F16	1,61E+03	1,61E+03	1,61E+03			
F17	4,95E+03	3,93E+03	2,02E+07			
F18	1,82E+03	1,84E+03	1,18E+09			
F19	1,90E+03	1,91E+03	2,18E+03			
F20	2,01E+03	2,03E+03	7,07E+04			

Figura 15. Comparación CSV6 con DE de D. Molina en Dimension 30.

Como vemos seguimos sin mejorar el DE de Daniel Molina.

Comparación con los algoritmos participantes en la competición CEC2014.

Vamos a ver los resultados obtenidos contra todos los algoritmos participantes en CEC2014.

	CMLSP	FCDE	FERDE	FWA-DM	GaAPADE	L-SHADE	MVMO	NRGA	OptBees	POBL_ADE	RSDE	SOO	DO+BOBYQ	UMOEAS	b3e3pbest	malschcm	CS_V6
F1	1,77E-007	1150613	5234091	5013,049	0	0	0,000495	27904,5	784,1906	16226,57	0	8810740	4569,72	0	2626512	0	2E+007
F2	1,11E-015	16034576	1,4E+008	0,000134	0	0	7,10E-009	914,6605	0,009883	2273,055	0	6,343	0,036	0	2,2E+008	0	3E+009
F3	0,000106	1234,988	4545,647	1,88E-009	0	0	9,86E-011	1516,806	0,921307	0,000574	0	6643,67	5842,92	0	4305,4	1,03E-007	18092
F4	3,34E-015	41,97763	48,45589	1,413235	30,68824	29,40955	9,545624	15,43558	2,690817	25,50826	2,81097	0,678	0	0	52,53329	0,085008	254,091
F5	16,86305	20,38114	20,09145	20,02724	19,6767	14,1456	16,58058	19,60688	19,99997	19,08704	19,21631	20	20	16,831	20,21299	13,65196	20,373
F6	0,06201	5,327998	3,473249	0,706304	0,148376	0,01754	0,003445	2,449826	3,016623	1,039458	0,052908	0,002	0,002	0	2,790775	0,000148	9,007
F7	0	0,822638	2,193035	0,094799	0,003163	0,003043	0,018584	0,20303	0,156159	0,162686	0,035496	0,049	0,049	0	3,306893	0	40,055
F8	2,070678	14,22719	9,76986	0,253617	0	0	6,69E-015	5,584734	1,16E-013	7,808883	0,660805	18,904	18,904	0	13,24861	0	61,492
F9	1,658501	35,79322	18,74572	6,008483	3,379005	2,344598	3,492111	8,693676	20,83557	7,630765	8,522406	8,955	8,955	2,725476	21,03687	3,31653	66,132
F10	196,1156	488,5916	151,5188	1,592692	0,151789	0,008572	2,136919	119,43	219,2256	153,4006	68,44158	130,39	130,39	0,37386	330,8471	7,677946	1375,68
F11	152,9958	976,2718	537,6227	372,2404	183,1153	32,05583	96,27604	575,9476	392,748	208,2013	290,6419	349,05	349,05	144,0443	782,9861	20,13497	1474,89
F12	0,030265	1,128525	0,629776	0,042495	0,14022	0,068167	0,042228	0,124164	0,130399	0,269454	0,220648	0	0	0	0,803077	0,016465	1,23
F13	0,02725	0,305439	0,335213	0,120614	0,060087	0,051562	0,035533	0,157686	0,416158	0,131173	0,127667	0,03	0,03	0,009436	0,31249	0,032923	1,94
F14	0,189165	0,338071	1,007054	0,213906	0,09424	0,081362	0,089059	0,253702	0,368651	0,260272	0,135988	0,13	0,13	0,110034	0,688413	0,12649	11,5
F15	0,896629	5,766517	19,13657	0,774837	0,605651	0,366099	0,434601	1,021838	2,438871	0,711842	0,983009	0,44	0,42	0,666698	47,01128	0,471494	562,12
F16	1,554589	3,50591	2,922204	1,75707	1,977198	1,240797	1,448515	2,746937	2,639589	1,409093	2,233398	2,52	2,52	1,530246	2,826145	1,054166	3,58
F17	312,7451	1765,618	314731,7	254,5371	9,914349	0,976664	9,356666	16074,91	684,3999	257,2398	47,70168	3122910	422,57	8,476833	101237,2	78,33846	95251,5
F18	30,85294	309,4347	234239,6	25,15813	0,222974	0,244093	0,7826	7419,754	33,5043	33,16152	1,996367	12932,1	3951,62	0,784029	225413,5	5,220721	225993
F19	1,25112	1,614197	2,032065	1,299117	0,256564	0,0773	0,15834	2,093335	0,933042	2,087875	1,030208	0,55	0,55	0,199971	2,374421	0,076607	9,01
F20	19,94066	235,2392	407578,2	13,37053	0,431554	0,184883	0,312556	1719,184	8,957556	12,58948	0,721467	9364,2	6925,1	0,370588	5688,917	8,056691	4335,42
F21	36,39071	1664,77	83860,17	94,64253	0,508552	0,408069	1,934721	4823,427	57,05615	102,5059	1,208642	24694,9	1940,39	0,540407	5004,723	49,28617	12691,6
F22	89,52842	26,10666	43,77415	34,08745	3,247398	0,044103	0,262886	37,56658	17,02468	30,02275	11,6549	126,46	126,47	0,244755	67,56668	8,474625	109,91
F23	201,8202	329,9485	332,5387	329,4575	329,46	329,4575	329,4575	329,4575	272,3515	329,4575	329,4575	200	200	329,4575	333,9172	329,4575	366,69
F24	109,9103	145,5083	126,9949	127,3903	108,9051	107,4896	109,2258	130,7641	137,378	123,8384	119,133	115,65	115,65	108,2987	130,9645	108,443	182,76
F25	127,5298	186,1197	156,8479	178,7233	163,6245	132,7388	116,126	183,6782	145,9907	186,1996	129,5116	145,16	139,08	126,0032	185,6212	175,0708	192,22
F26	100,0194	100,3135	100,3318	100,1384	100,0688	100,05	100,0323	100,1366	100,3964	100,1207	100,1291	100,05	100,05	100,014	100,3284	100,0364	101,37
F27	41,13156	173,6789	135,3908	321,2754	89,68693	58,06393	17,20188	280,7776	7,422917	255,937	91,2492	200	200	25,477	158,9575	184,7796	151,64
F28	280,3234	508,0455	435,8938	347,1652	383,2116	380,8105	361,0548	477,1474	306,6715	423,3538	386,9035	200	200	312,9189	420,1244	388,7168	767,82
F29	200	367,5578	83243,14	211,7373	222,3229	221,9867	181,4021	413,291	219,9635	355435	212,6059	200	200	195,4616	16758,58	227,0654	107117
F30	216,4126	1390,678	2917,534	394,2959	467,1992	464,8828	491,7453	1727,538	389,1652	637,9676	505,2479	200	200	233,8867	1320,655	585,1143	4193,6

Figura 16. Ranking CSV6 en CEC2014 Dimension 10.

Función	Posición
F1	17
F2	17
F3	17
F4	17
F5	16
F6	17
F7	17
F8	17
F9	17
F10	17
F11	17
F12	17
F13	17
F14	17
F15	17
F16	17
F17	14
F18	16
F19	17
F20	13
F21	15
F22	15
F23	17
F24	17
F25	17
F26	17
F27	9
F28	17
F29	16
F30	17

Posición media: $(22 \cdot 17 + 3\dot{1}6 + 2 \cdot 15 + 14 + 13 + 9)/30 = 16,26$. Hemos quedado últimos en la mayoría de las ocasiones, por ende, quedamos últimos en la competición.

	CMLSP	FCDE	FERDE	FWA-DM	GaAPADE	L-SHADE	MVMO	NRGA	OptBees	POBL_ADE	RSDE	SOO	DO+BOBYQ	UMOEAS	b3e3pbest	malschcm	CS_V6
F1	3,98E-010	39737210	74247963	276356,9	1,00E-014	0	0,001066	574302,5	85681,08	15954,83	1500,289	2,2E+008	2674850	0	13225063	0	8E+008
F2	0	1,8E+009	3,5E+009	1,08E-016	1,67E-015	0	2,38E-005	9276,281	3,32E-012	313,6771	1,19E-009	31387	99,611	0	5,2E+008	0	6E+010
F3	1,23E-008	9119,318	13921,84	4,42E-016	2,23E-015	0	0,001106	4581,834	0,008414	6,43E-010	0,047433	10810,2	7840,39	0	5609,163	26,19492	104844
F4	2,17E-006	196,6229	489,4935	20,36249	2,83E-012	0	4,38E-013	80,58102	12,56348	63,44831	3,05179	109,346	36,755	0	63,42428	0	9128,519
F5	19,99896	20,95678	20,19698	20,50597	20,00229	20,11468	19,99956	20,00014	20	20,6377	20,33411	20	20	20,16136	20,51938	19,99971	500,594
F6	0	23,29159	24,03274	12,86363	0,613421	1,38E-007	3,620226	17,82335	16,3762	5,191192	5,160633	1,897	1,907	0	21,98693	1,135849	630,447
F7	0	25,56764	31,10055	0,008546	2,23E-015	0	0,00299	0,015894	0,037456	0,023727	0,000846	0,996	0,409	0	4,091267	0,000193	1242,585
F8	9,837491	119,9245	46,40362	1,13E-013	1,746266	0	0,858412	26,58988	3,63E-013	55,86554	20,41341	92,531	92,531	1,9509	51,06523	0,019535	1094,408
F9	2,18526	201,6079	122,0642	56,61793	16,99616	6,784876	25,12758	45,69018	137,1473	84,63108	57,95042	59,706	59,697	8,447608	104,3559	17,92877	1245,218
F10	1469,479	3548,67	1339,702	8,525867	8,143284	0,016329	17,86361	1073,473	1041,298	2167,675	329,1781	2312,38	2131,47	13,64395	1462,061	81,2466	6629,47
F11	1822,477	5314,85	3551,51	2629,753	1896,767	1229,479	1541,69	3405,553	2716,635	3858,27	2737,263	2151,25	2091,05	1575,324	3877,698	1549,521	6876,44
F12	0,000146	1,9111	1,35218	0,371251	0,203389	0,160577	0,072055	0,150542	0,181246	0,950506	0,443941	0,03	0,03	0,002153	0,989508	0,015975	1201,34
F13	0,048135	0,741534	0,845991	0,38862	0,144886	0,124117	0,157294	0,28111	0,560833	0,285834	0,305364	0,35	0,34	0,056813	0,384483	0,137676	1304,63
F14	0,311895	3,212646	12,36815	0,268582	0,211286	0,241702	0,198856	0,186616	0,399547	0,225915	0,236281	0,29	0,28	0,211042	1,015493	0,221635	1581,78
F15	3,020714	4496,495	41716,55	7,373282	3,060288	2,14637	2,855227	13,7297	12,71035	7,733778	5,92202	22,51	21,69	3,088248	182,0983	2,450783	1100698
F16	12,78485	12,86934	12,14822	10,9783	9,880971	8,499015	10,20949	11,46696	10,90813	10,44072	10,60218	9,86	9,81	10,53998	11,2144	9,647416	1609,52
F17	934,8205	705470	7040904	6285,723	199,7495	187,5081	900,8198	234648	27401,73	1103,692	1239,143	28114700	42148,7	1009,247	520178,7	697,8608	2E+007
F18	75,44929	2238462	95603637	76,66202	9,324016	5,910067	28,93824	550,4614	195,5904	109,7573	95,4482	2854,99	41,58	23,17351	3170488	566,9142	1E+009
F19	3,877261	25,99335	47,70424	9,94723	3,620694	3,681779	3,079965	13,73975	7,898369	8,881846	5,652966	183,62	16,3	3,811122	10,08434	5,822519	2179,31
F20	9,79078	5091,055	10330,37	42,7794	5,585408	3,081863	109,1572	11377,95	852,6656	38,90711	37,30235	38149,6	34381,2	11,68166	3225,947	198,878	70700,58
F21	229,6321	55807,81	3113705	729,4119	118,2128	86,8329	466,617	180670,7	17431,9	386,3059	470,9285	16290100	15435	317,8185	434810,4	573,2908	5077008
F22	60,94744	389,7187	456,6456	145,653	62,6329	27,6009	144,5745	407,1396	232,0181	230,5926	191,3664	1019,94	956,48	95,94877	432,9887	158,8346	3446,19
F23	200	326,0803	340,83	314,0129	315,24	315,2441	315,2441	315,2454	314,7704	315,2441	315,2441	200	200	314,0129	319,2658	315,2442	2641,04
F24	200,0005	250,5472	245,6888	226,0679	224,2922	224,0462	224,7538	228,4003	236,155	221,5734	224,3782	200	200	224,4925	228,0547	222,2347	2619,94
F25	200	209,0371	211,1376	200,5653	202,5916	202,6063	203,2951	210,5359	200,975	203,6276	202,727	200	200	200,1947	205,1927	205,6072	2571,19
F26	123,6993	100,8767	100,9757	100,3497	100,1431	100,1136	100,1622	100,3635	100,551	139,4971	100,337	200	200	100,0587	100,3732	102,0888	2605,11
F27	200	891,4109	582,5453	401,0319	327,6451	300	401,132	587,6208	402,4	420,5095	468,5444	200	200	301,0253	416,6909	328,4447	3418,72
F28	200	1106,042	1328,12	393,0306	833,5894	840,3043	876,7701	1593,633	431,3996	916,3845	905,0781	200	200	368,3278	931,0008	823,383	6609,61
F29	200	315919,7	4349024	211,0373	500071,9	716,8803	736,0847	1320,599	215,901	339290,3	651806	200	200	205,7819	433867,1	1146,583	1E+008
F30	200	9424,439	100494	451,3509	1646,424	1246,381	2001,147	2889,442	592,8338	1292,004	1695,621	200	200	430,3113	12856,22	2040,669	869298,4

Figura 16. Ranking CSV6 en CEC2014 Dimension 30.

Función	Posición
F1	17
F2	17
F3	17
F4	17
F5	17
F6	17
F7	17
F8	17
F9	17
F10	17
F11	17
F12	17
F13	17
F14	17
F15	17
F16	17
F17	16
F18	17
F19	17
F20	17
F21	16
F22	17
F23	17
F24	17
F25	17
F26	17
F27	17
F28	17
F29	17
F30	17

Posición media: $(28 \cdot 17 + 2 \cdot 16)/30 = 16,93$. Participando en esta dimensión, hubiéramos quedado también

los últimos.