Universidad de Granada

Ingeniería Informática

Computación y Sistemas Inteligentes

Elephant Search Algorithm (ESA)

Autor: José Antonio Ruiz Millán
email: jantonioruiz@correo.ugr.es
Curso: 2017-2018
Asignatura: Metaheurísticas
20 de julio de 2018

Índice

1. Descripción del algoritmo

En esta sección haré una breve descripción del algoritmo estudiado. Todos los detalles podrán ser consultados a través de los diferentes enlaces indicandos en este documento.

En primer lugar, este algoritmo, como su nombre indica, es un algoritmo bio-inspirado, concretamente en los elefantes. Intenta simular la forma de exploración de las manadas de elefantes.

Cada manada de elefantes está diferenciada por sexo, donde cada uno de ellos realiza una tarea diferente, es decir, los elefantres machos serán los encargados de explorar nuevas zonas mejores que la actual para encontrar nueva comida y así guiar a la manada hacia allí, mientras que las hembras son las encargadas de explotar y examinar cada una de las zonas en la que se encuentran.

Dentro del grupo de hembras, existe una líder que será la que guie al grupo en una dirección para la explotación de esa zona.

Los elefantes tienen una vida, cuando llega a su fin, se crea un nuevo elefante combinando varias caracteristicas de otros elefantes de la manada que veremos a continuación.

Metaherística:

Toda esta teoría tenemos que traspasarla a una metaherística, por lo que tendremos los siguiente elementos:

- Desplazar a un elefante de zona:

  En mi caso utilizo un movimento utilizando una distribución de Levy. Este movimiento permite un mejor explorado de la zona completa a diferencia de otros métodos mas simples como puede ser un valor aleatorio uniforme.

  Lo que realizo es que cuando se realiza un movimiento sobre un elefante, modifico un valor del vector utilizando una distribución de Levy como he comentado anteriormete. Por lo que para cada elefante, saco un valor aleatorio que me dará la posicón a modificar y seguidamente se aplica lo siguiente:

  $$x_{new,i} = x_i + LevyValue(1{,}7)$$

  Donde LevyValue realiza una serie de calculos que pueden ser consultados en el código adjuntado ya que son varios calculos para devolver un valor y prefiero no añadirlo a este documento.

- Desplazar a una elefante de zona:

  Como he comentado anteriormente las elefantes hembras tienen una lider, y éstas se moveran hacia ella en cada movimiento, por lo que lo que haremos será, para cada hembra, recorrer todo el vector solución y aplicar lo siguiente:

  $$x_{new,i} = x_i + \alpha \cdot (lider_i - x_i) \cdot r)$$

  Donde $\alpha \in [0, 1]$ e indica la importancia que le damos a la lider, y $r \in [0, 1]$ es un valor aleatorio con una distribución uniforme.

- Choque de elefantes machos:

Cuando dos elefantes machos se encuentran lo suficientemente cerca, no nos interesa, es por ello que se aplica una transformación utilizando el mismo método que el movimiento de los machos pero en este caso se aplica a un 40 % del tamaño de la solución.

- Actualización de la vida de cada elefante:

  Obviamente, en cada generación sumaremos 1 a la vida de cada uno de los elefantes, sin embargo se utiliza una probabilidad para saber si un elefante debe morir o no en una generacion determinada, por lo que:

$$\text{si} \quad r > (\tfrac{vida(x)}{limitacionVida})^3 \quad \rightarrow 1$$
$$\text{en otro caso} \quad \rightarrow 0$$

  Donde $r \in [0, 1]$ es un numero aleatorio con una distribución uniforme, *vida(x)* es la vida actual del elefante x y *limitacionVida* es la vida total que un elefante puede tener (este es un parametro a establecer al comienzo). Por úlimo, el resultado se interpreta de tal forma que si obtenemos como resultado un 1 significa que el elfante sigue viviendo y si obtenemos como resultado un 0, el elefante debe morir y nacer un nuevo elefante.

- Crear descendiente:

  Cuando un elefante muere, se crea un nuevo conservando el sexo y el número. Esto nos permite tener una componente evolutiva donde cada hijo creado tendrá la siguiente posicion:

$$x_{new,i} = (0{,}6 \cdot madre_i + 0{,}3 \cdot padre_i + 0{,}1 \cdot lider_i)$$

  Esto nos permite colocar en una posición orientativa al nuevo descendiente y que ahora con una vida que empieza de nuevo poder seguir realizando su trabajo dependiendo del sexo.

- Movimiento hembras hacia macho:

  Una caraterística de este algorimo nos dice que cuando un macho encuentre una solución mejor que la actual hembra, todas las hembras se mueven a esa zona para que comiencen a explorarla.

Tanto el pseudocódigo como lo explicado anteriormente pero con más detalle puede ser consultado[1] y verificado en los documentos.

Por último, aclarar que este algoritmo se puede considerar un algoritmo memético ya que como he comentado tiene una componente evolutiva, y aparte de eso, a la elefante lider se le aplica una búsqueda local en cada generación. Por ello, podemos decir que este algoritmo es un algoritmo memético.

2. Diferentes versiones investigadas.

En mi caso, he estado estudiando distintas versiones del algoritmo, algunas ya experimentadas como VESA[2] y una version con optimización de parámetros[3], y algunas propuestas por mí viendo el funcionamiento del mismo.

2.1. Vitality Elephant Search Algorithm (VESA)

Esta mejora consiste en determinar una vida "variable" para cada elefante. Con esto lo que hacemos es priorizar a los elefantes con mejor valor de fitness y hacer que éstos duren mas tiempo de vida mientras que los elefantes con peor valor van a morir más rapido.

Los cambios a realizar son los siguientes:

- Actualizacion de vida:

  En el caso simple teníamos que en cada generación sumamos 1 a la vida de cada uno de los elefantes, ahora sin embargo utilizaremos un nuevo parámetro que llamaremos vitalidad y que se calcula de la siguiente forma:

$$\text{si} \quad \frac{valor(x)-optimo(f)}{ln(vida(x))} < 0{,}1 \quad \rightarrow \text{vitalidad} = -1$$

$$\text{si} \quad \frac{valor(x)-optimo(f)}{ln(vida(x))} > 2 \quad \rightarrow \text{vitalidad} = 2$$

$$\text{en otro caso} \quad \text{vitalidad} = 1$$

$$vida(x) = vida(x) + \text{vitalidad}$$

Con esto conseguimos que si el valor de un elefante concreto es bueno, este elefante irá viviendo mas que el resto mientras que si un elefante es malo, vivirá menos tiempo que el resto.

2.2. ESA MIN-MAX

Esta mejora nos permite ajustar el ratio de hembras y machos en la población. El ratio predeterminado es 1:4, es decir, para cada macho debe haber 4 hembras.

El procedimiento que sigue esta mejora es el siguiente:

- Comienza creando 3 distintas parejas de valores, de la siguiente forma:

  LimiteSuperior = (N-1,1); Media = (N/2,N/2); LimiteInferior = (1,N-1).

  Siendo N en número total de población.

- Ahora ejecuta el algorimo con los 3 distintos ratios que he mencionado y dependiendo del resultado obtenido realiza unas actualizaciones u otras. Llamaré ls, m y li a las 3 parejas para resumir le nombre.

  Si $fitness(ls) < fitness(m) < fitness(li)$ o $fitness(ls) > fitness(m) > fitness(li)$ diríamos que tenemos unos valores monótonos, por lo que asumimos que la mejor pareja es ls o li dependiendo de lo que busquemos y por lo tanto, si ls es la mejor, $li = m$ y $m = \frac{m+ls}{2}$; y en caso de que li sea la mejor, tenemos que $ls = m$ y $m = \frac{m+li}{2}$

  Por otra parte, si $fitness(ls) < fitness(m)$ y $fitness(li) < fitness(m)$, estaríamos en el caso de que m es el peor de las 3 parejas, por lo que lo que hacemos es moverlo hacia el mejor de los 2. Así que lo que haremos será, si $fitness(ls) < fitness(li)$, $m = \frac{m+ls}{2}$ y en caso contrario, si $fitness(li) < fitness(ls)$, $m = \frac{m+li}{2}$.

Por último, si $fitness(m) < fitness(ls)$ y $fitness(m) < fitness(li)$, estamos en el caso de que m es el mejor de los 3, por lo que haremos que los otros 2 se aproximen a él. Por lo que $ls = \frac{m+ls}{2}$ y $li = \frac{m+li}{2}$.

- Este proceso se va realizando iterativamente hasta que se cumpla que $\|ls - li <= 2\|$ y $\|m - ls <= 1\|$ y $\|m - ls <= 1\|$

Con esta mejora conseguríamos adaptar el ratio de machos y hembras a cada problema en vez de utilizar el predeterminado como he comentado al comienzo de esta sección. No obstante, toda la información sobre esta mejora se puede consultar en las referencias[3].

2.3. Jump Vitality Elephant Search Algorithm (J-VESA)

Esta mejora la he propuesto ya que este algoritmo realiza una búsqueda local en cada generación y en ocasiones suele atascarse. Con esta forma de trabajar, me di cuenta que una vez encuentra un "optimo" en alguna zona, la búsqueda local puede estar varias generaciones parada esperando mejores soluciones para poder ir a explorarlas. Por lo que definí un parámetro que en mi caso será $2 \cdot nDimensiones$ (aunque puede ser cualquier otro), para que cuando la busqueda local se quede atascada un número de veces, permita saltar al mejor macho visto en todas las generaciones aunque este macho no sea mejor que la propia hembra.

Simplemente esta mejora nos hace añadir un nuevo elemento para tener almacenado el mejor macho visto hasta el momento que sea peor que la hembra y cuando suceda lo comentado anteriormente, mover las hembras hacia éste y seguir como hasta ahora.

La mejora la realizo sobre VESA y no sobre ESA porque como veremos en los siguientes apartado, obtuve mejor resultados con VESA.

2.4. CMAES+J-VESA

Para terminar, visualicé que CMAES es un algoritmo que tiene mucha potencia en estos tipo de problemas pero que al incorporarlo como búsqueda local del propio algoritmo memético no daba buenos resultados ya que consume demasiadas evaluaciones y no llega a converger, por lo que lo que realicé fue dividir las iteraciones para seguir teniendo una proporción como en los anteriores casos (que veremos más adelante) y comenzar el algoritmo con una solución de calidad, es decir, utilizo CMAES dandole un 10 % de las evaluaciones para obtener una solucion de calidad en vez de una aleatoria y esta solución pasará a ser nuestra líder de la población. Con esto, el resto de evaluaciones (90 %) se ejecuta el propio algoritmo teniendo en cuenta como he comentado que la proporción de exploración y explotación esté mas o menos equilibrada, en mi caso siempre he dado un porcentaje ligeramente mas grande a la exploración que a la explotación.

Como antelación a los resultados, he de comentar que este algoritmo es el que mejor resultados me ha dado, tambien comentar que CMAES es la potencia más significativa ya que si en vez de darle un 10 % le damos un 20 %, mejora aún mas.

3. Descripción para ejecución

Los ficheros entregados se encuentran dentro de un directorio llamado ESA, lo único que hay que realizar es **cmake .** para generar el makefile, y seguidamente hacer **make**. Con esto tenemos el ejecutable llamado example_ls para poder ejecutar el algoritmo. Los distintos parámetros y mejoras se pueden editar tanto desde example.cc como desde ESA.cpp.

4. Análisis de los resultados

4.1. Descripción

- **Elephant Search (ESA)**

  Para este algoritmo tengo una población de **60 elefantes**, de los cuales utilizando el ratio por defecto tengo 12:48 siendo machos:hembras respectivamente. El número de evaluaciones que se realizan **para la exploración son 60 por generacion** ya que es el numero que tenemos de población, por lo que he definido que por generación el numero de iteraciones que hará la **busqueda local será de 36**, teniendo entonces 96 evaluaciones por generación. Esto hace que hagamos unas 1041 generaciones y que el número total de evaluaciones dedicadas para exploración sea $\approx$ 62500 y para la explotación $\approx$ 37500 en el caso de 10 dimensiones. Para 30 dimensiones, con los mismos valores, utilizamos $\approx$ 187500 para exploración y $\approx$ 112500 para explotación.

- **Jump Vitality Elephant Search Algorithm (J-VESA)**

  Conservamos los valores del ESA y añadimos como comenté en apartados anteriores que el nuevo parámetro utilizado para indicar cuando salto a un elefante aunque sea peor será $2 \cdot nDimensiones$.

- **ESA MIN-MAX**

  Conservamos los parametros de ESA ha excepción del ratio, ya que en este caso se calcula automáticamente. No sabía muy bien como distribuir esto ya que como tengo que ejecutar el propio algoritmo para comprobar los rangos, decidí utilizar evaluaciones de las max_eval de cada función, por lo que utilicé un 30 % de las evaluaciones para calcular el mejor ratio y el resto 70 % para la ejecución del algoritmo.

- **CMAES+J-VESA**

  Se mantienen los mismos valores que para ESA, cambiando obviamente las evaluaciones ya que ahora tenemos al comienzo un 10 % de evaluaciones para CMAES y el resto para el algoritmo en sí. Por lo que tenemos que modificar y disminuir las búsquedas locales del algoritmo para que globalmente siga utilizando el mismo ratio que al principio. Tenemos que **CMAES hace 10000 evaluaciones** , y sabamos que **en exploración hacemos 60 evaluaciones por generacion** como anteriormente por lo que el cambio ahora es que **la búsqueda local hará 24 evaluaciones por generación**, obteniendo $\approx$ 64300 para exploración y $approx$25700 para explotación que mas las 10000 de CMAES hacen $\approx$ 35700 en dimensión 10. Para dimensión 30 utilizamos otros valores para la explotación, haciendo **30**

evaluaciones por generacion, por lo que obtenemos que tendremos ≈ 180000 evaluaciones para exploración y ≈ 90000 para explotación que sumadas a las 30000 que hace CMAES, tenemos ≈ 120000.

4.2. Resultados obtenidos

4.2.1. Comparativa ESA D10

Tabla 1: Resultados obtenidos por todas las versiones de ESA en 10 dimensiones

	ESA SW	ESA SIMPLEX	J-ESA-SW	ESA-SW MIN-MAX	J-VESA-SW	J-VESA-SW MIN-MAX	CMAES+J-VESA-SW
F1	4,85E+04	2,27E+06	4,48E+04	1,18E+05	3,65E+04	1,06E+05	1,00E+02
F2	3,57E+03	1,48E+05	2,67E+02	1,97E+03	2,65E+02	2,95E+02	2,00E+02
F3	6,12E+03	1,28E+04	6,37E+03	9,20E+03	6,67E+03	9,23E+03	3,00E+02
F4	4,31E+02	4,39E+02	4,25E+02	4,29E+02	4,23E+02	4,27E+02	4,00E+02
F5	5,20E+02	5,20E+02	5,20E+02	5,20E+02	5,20E+02	5,20E+02	5,20E+02
F6	6,10E+02	6,07E+02	6,10E+02	6,11E+02	6,10E+02	6,10E+02	6,01E+02
F7	7,01E+02	7,00E+02	7,00E+02	7,02E+02	7,00E+02	7,00E+02	7,00E+02
F8	8,66E+02	8,42E+02	8,39E+02	8,53E+02	8,38E+02	8,35E+02	8,14E+02
F9	9,54E+02	9,55E+02	9,44E+02	9,49E+02	9,45E+02	9,43E+02	9,16E+02
F10	2,12E+03	1,60E+03	1,81E+03	2,08E+03	1,73E+03	1,80E+03	1,68E+03
F11	2,32E+03	2,40E+03	1,98E+03	2,25E+03	1,99E+03	2,07E+03	1,86E+03
F12	1,20E+03	1,20E+03	1,20E+03	1,20E+03	1,20E+03	1,20E+03	1,20E+03
F13	1,30E+03	1,30E+03	1,30E+03	1,30E+03	1,30E+03	1,30E+03	1,30E+03
F14	1,40E+03	1,40E+03	1,40E+03	1,40E+03	1,40E+03	1,40E+03	1,40E+03
F15	1,53E+03	1,50E+03	1,51E+03	1,53E+03	1,51E+03	1,51E+03	1,50E+03
F16	1,60E+03	1,60E+03	1,60E+03	1,60E+03	1,60E+03	1,60E+03	1,60E+03
F17	2,94E+03	1,05E+05	3,79E+03	3,69E+03	3,05E+03	3,23E+03	2,24E+03
F18	1,19E+04	9,76E+03	6,79E+03	1,05E+04	6,47E+03	1,02E+04	1,87E+03
F19	1,93E+03	1,90E+03	1,92E+03	1,92E+03	1,92E+03	1,92E+03	1,90E+03
F20	6,81E+03	9,79E+03	6,72E+03	7,92E+03	7,57E+03	7,95E+03	2,09E+03

4.2.2. Comparativa ESA D30

Tabla 2: Resultados obtenidos por todas las versiones de ESA en 30 dimensiones

	ESA SW	ESA SIMPLEX	J-ESA-SW	ESA-SW MIN-MAX	J-VESA-SW	J-VESA-SW MIN-MAX	CMAES+J-VESA-SW
F1	3,02E+05	2,92E+08	3,46E+05	5,57E+05	3,56E+05	5,07E+05	5,86E+03
F2	1,02E+04	8,86E+09	4,19E+03	5,18E+03	4,36E+03	3,99E+03	2,00E+02
F3	1,20E+04	4,99E+04	1,16E+04	2,86E+04	1,19E+04	2,84E+04	3,00E+02
F4	4,25E+02	1,64E+03	4,11E+02	4,36E+02	4,28E+02	4,37E+02	4,04E+02
F5	5,20E+02	5,21E+02	5,20E+02	5,20E+02	5,20E+02	5,20E+02	5,20E+02
F6	6,41E+02	6,36E+02	6,41E+02	6,40E+02	6,40E+02	6,40E+02	6,06E+02
F7	7,00E+02	7,10E+02	7,00E+02	7,00E+02	7,00E+02	7,00E+02	7,00E+02
F8	1,04E+03	1,03E+03	9,47E+02	9,48E+02	9,49E+02	9,47E+02	8,53E+02
F9	1,09E+03	1,21E+03	1,08E+03	1,08E+03	1,07E+03	1,08E+03	9,52E+02
F10	5,62E+03	6,44E+03	4,56E+03	4,65E+03	4,64E+03	4,82E+03	3,75E+03
F11	5,92E+03	8,21E+03	5,02E+03	5,17E+03	5,07E+03	4,92E+03	4,18E+03
F12	1,20E+03	1,20E+03	1,20E+03	1,20E+03	1,20E+03	1,20E+03	1,20E+03
F13	1,30E+03	1,30E+03	1,30E+03	1,30E+03	1,30E+03	1,30E+03	1,30E+03
F14	1,40E+03	1,40E+03	1,40E+03	1,40E+03	1,40E+03	1,40E+03	1,40E+03
F15	1,71E+03	1,58E+03	1,64E+03	1,65E+03	1,64E+03	1,65E+03	1,50E+03
F16	1,61E+03	1,61E+03	1,61E+03	1,61E+03	1,61E+03	1,61E+03	1,61E+03
F17	4,61E+04	7,19E+07	3,85E+04	5,52E+04	3,73E+04	4,79E+04	3,60E+03
F18	8,76E+03	9,00E+07	2,08E+03	2,09E+03	2,07E+03	2,08E+03	2,08E+03
F19	1,98E+03	2,08E+03	1,93E+03	1,94E+03	1,93E+03	1,94E+03	1,92E+03
F20	1,22E+04	9,72E+04	1,35E+04	2,03E+04	1,36E+04	3,16E+04	2,51E+03

4.2.3. CMAES+J-VESA vs DE D10

Tabla 3: Comparativa entre ESA mejorado y DE en 10 dimensiones

	CMAES+J-VESA-SW	Debin	Deexp	CMAES+J-VESA-SW – Debin	CMAES+J-VESA-SW – Deexp
F1	1,00E+02	1,00E+002	1,00E+002	0,00E+00	0,00E+00
F2	2,00E+02	2,00E+002	2,00E+002	0,00E+00	0,00E+00
F3	3,00E+02	3,00E+002	3,00E+002	0,00E+00	0,00E+00
F4	4,00E+02	4,22E+002	4,22E+002	-2,11E+01	-2,14E+01
F5	5,20E+02	5,20E+002	5,20E+002	-2,14E-01	-1,38E-01
F6	6,01E+02	6,01E+002	6,00E+002	5,64E-01	9,63E-01
F7	7,00E+02	7,00E+002	7,00E+002	-2,82E-02	-2,74E-02
F8	8,14E+02	8,04E+002	8,00E+002	9,30E+00	1,36E+01
F9	9,16E+02	9,12E+002	9,09E+002	4,13E+00	7,32E+00
F10	1,68E+03	1,05E+003	1,01E+003	6,29E+02	6,70E+02
F11	1,86E+03	1,54E+003	1,66E+003	3,12E+02	2,01E+02
F12	1,20E+03	1,20E+003	1,20E+003	-2,57E-01	-2,70E-01
F13	1,30E+03	1,30E+003	1,30E+003	-2,10E-02	-1,90E-02
F14	1,40E+03	1,40E+003	1,40E+003	9,90E-02	1,12E-01
F15	1,50E+03	1,50E+003	1,50E+003	-6,61E-01	-1,28E-01
F16	1,60E+03	1,60E+003	1,60E+003	1,39E+00	1,37E+00
F17	2,24E+03	1,72E+003	1,71E+003	5,20E+02	5,28E+02
F18	1,87E+03	1,80E+003	1,80E+003	7,37E+01	7,37E+01
F19	1,90E+03	1,90E+003	1,90E+003	2,56E+00	2,42E+00
F20	2,09E+03	2,00E+003	2,00E+003	9,05E+01	9,06E+01
MEDIA				8,11E+01	7,83E+01

4.2.4. CMAES+J-VESA vs DE D30

Tabla 4: Comparativa entre ESA mejorado y DE en 30 dimensiones

	CMAES+VESA-SW+SALTO	Debin	Deexp	VESA-SW CMAES – Dfbin	VESA-SW CMAES – Dfexp
F1	5,86E+03	8,88E+004	2,95E+005	-8,29E+04	-2,89E+05
F2	2,00E+02	2,00E+002	2,00E+002	0,00E+00	0,00E+00
F3	3,00E+02	3,00E+002	3,00E+002	0,00E+00	0,00E+00
F4	4,04E+02	4,06E+002	4,33E+002	-2,60E+00	-2,95E+01
F5	5,20E+02	5,21E+002	5,20E+002	-8,87E-01	-4,34E-01
F6	6,06E+02	6,04E+002	6,18E+002	1,80E+00	-1,21E+01
F7	7,00E+02	7,00E+002	7,00E+002	-1,80E-03	1,00E-04
F8	8,53E+02	8,14E+002	8,00E+002	3,95E+01	5,34E+01
F9	9,52E+02	1,03E+003	9,82E+002	-8,19E+01	-2,94E+01
F10	3,75E+03	1,49E+003	1,00E+003	2,26E+03	2,74E+03
F11	4,18E+03	7,21E+003	4,86E+003	-3,02E+03	-6,81E+02
F12	1,20E+03	1,20E+003	1,20E+003	-1,80E+00	-9,00E-03
F13	1,30E+03	1,30E+003	1,30E+003	-6,80E-02	-1,46E-01
F14	1,40E+03	1,40E+003	1,40E+003	-1,34E-01	-1,28E-01
F15	1,50E+03	1,51E+003	1,51E+003	-1,10E+01	-5,18E+00
F16	1,61E+03	1,61E+003	1,61E+003	6,37E-01	2,03E+00
F17	3,60E+03	4,97E+003	3,94E+003	-1,37E+03	-3,44E+02
F18	2,08E+03	1,82E+003	1,84E+003	2,65E+02	2,41E+02
F19	1,92E+03	1,90E+003	1,91E+003	1,21E+01	9,85E+00
F20	2,51E+03	2,01E+003	2,03E+003	4,96E+02	4,83E+02
MEDIA				-4,22E+03	-1,43E+04

4.2.5. CEC2014 D10

Tabla 5: Comparativa con los algoritmos del CEC2014 en dimension 10

	CoDE	D-SHADE	EPSDE	JADE	L-SHADE	NBIPOP-aCMA-ES	SHADE11	SaDE	dynNP-jDE	iCMAES-ILS	CMAES+J-VESA-SW
F1	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	2,55E+00	2,17E-07	0,00E+00	0,00E+00
F2	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00
F3	0,00E+00	0,00E+00	0,00E+00	6,17E-03	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00
F4	1,05E+01	3,08E+01	0,00E+00	2,76E+01	2,94E+01	2,82E+00	2,95E+01	1,81E+01	3,32E+00	1,44E+01	4,78E-01
F5	1,84E+01	1,77E+01	2,00E+01	1,73E+01	1,41E+01	1,81E+01	1,80E+01	1,58E+01	1,60E+01	1,47E+01	2,00E+01
F6	1,65E-06	0,00E+00	3,04E+00	1,76E-01	1,75E-02	3,31E-01	0,00E+00	0,00E+00	0,00E+00	0,00E+00	1,15E+00
F7	3,76E-02	5,31E-03	1,76E-02	1,19E-02	3,04E-03	0,00E+00	9,78E-03	7,24E-03	4,97E-03	0,00E+00	8,40E-03
F8	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	3,70E+00	0,00E+00	0,00E+00	0,00E+00	2,54E-01	1,37E+01
F9	3,88E+00	3,08E+00	3,69E+00	3,51E+00	2,34E+00	3,26E-01	3,14E+00	3,58E+00	3,86E+00	9,75E-02	1,61E+01
F10	3,55E-02	4,90E-02	4,41E-02	6,12E-03	8,57E-03	9,16E+01	1,22E-02	1,96E-02	2,45E-03	1,22E+02	6,81E+02
F11	7,60E+01	5,49E+01	3,23E+02	8,37E+01	3,21E+01	1,17E+02	6,32E+01	1,96E+02	1,36E+02	8,59E+00	7,56E+02
F12	4,34E-02	5,29E-02	3,21E-01	2,50E-01	6,82E-02	1,01E-02	1,36E-01	4,35E-01	3,11E-01	6,50E-02	1,69E-01
F13	7,98E-02	4,89E-02	1,22E-01	8,40E-02	5,16E-02	1,09E-02	7,40E-02	1,25E-01	1,19E-01	9,11E-03	9,30E-02
F14	1,07E-01	9,01E-02	1,36E-01	1,11E-01	8,14E-02	2,82E-01	1,06E-01	1,86E-01	1,35E-01	1,55E-01	2,75E-01
F15	6,52E-01	4,03E-01	7,54E-01	5,78E-01	3,66E-01	5,47E-01	5,05E-01	7,90E-01	7,82E-01	7,23E-01	1,11E+00
F16	1,13E+00	1,34E+00	2,54E+00	1,65E+00	1,24E+00	2,53E+00	1,56E+00	1,97E+00	1,59E+00	1,91E+00	3,73E+00
F17	2,66E+00	3,38E+00	5,33E+01	3,09E+01	9,77E-01	3,89E+01	1,56E+00	2,83E+01	2,62E+00	2,10E+01	5,37E+02
F18	4,31E-01	4,75E-01	1,20E+00	2,39E-01	2,44E-01	3,58E+00	2,37E-01	1,65E+00	4,41E-01	5,26E-01	7,42E+01
F19	7,45E-02	2,05E-01	1,43E+00	2,55E-01	7,73E-02	8,28E-01	1,92E-01	6,69E-02	1,22E-01	7,08E-01	2,88E+00
F20	2,39E-02	2,73E-01	1,65E-01	3,24E-01	1,85E-01	1,32E+00	2,43E-01	1,08E-01	4,15E-02	8,04E-01	9,07E+01

4.2.6. CEC2014 D30

Tabla 6: Comparativa con los algoritmos del CEC2014 en dimension 30

	CoDE	D-SHADE	EPSDE	JADE	L-SHADE	NBIPOP-aCMA-ES	SHADE11	SaDE	dynNP-jDE	iCMAES-ILS	CMAES+J-VESA-SW
F1	2,63E+04	5,04E-03	2,42E+04	4,48E+02	0,00E+00	0,00E+00	4,81E+02	2,99E+05	4,65E+04	0,00E+00	5,76E+03
F2	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00
F3	0,00E+00	0,00E+00	0,00E+00	5,63E-04	0,00E+00	0,00E+00	0,00E+00	1,43E+01	0,00E+00	0,00E+00	0,00E+00
F4	2,52E+00	5,03E-09	3,21E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	3,72E+01	2,04E+00	0,00E+00	3,53E+00
F5	2,01E+01	2,00E+01	2,03E+01	2,03E+01	2,01E+01	2,05E+01	2,01E+01	2,05E+01	2,03E+01	2,00E+01	2,00E+01
F6	1,99E+00	5,92E-02	1,89E+01	9,42E+00	1,38E-07	7,14E-01	5,29E-01	5,46E+00	1,20E+00	4,00E-03	5,54E+00
F7	1,45E-04	0,00E+00	2,08E-03	0,00E+00	0,00E+00	0,00E+00	4,83E-04	1,23E-02	0,00E+00	0,00E+00	1,00E-04
F8	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	9,98E+00	0,00E+00	7,80E-02	0,00E+00	2,42E+00	5,34E+01
F9	4,04E+01	8,70E+00	4,44E+01	2,62E+01	6,78E+00	3,24E+00	1,58E+01	3,81E+01	3,39E+01	2,57E+00	5,23E+01
F10	5,00E-01	3,51E-02	2,01E-01	5,31E-03	1,63E-02	6,36E+02	1,27E-02	2,69E-01	4,08E-03	1,45E+02	2,75E+03
F11	1,95E+03	1,30E+03	3,56E+03	1,64E+03	1,23E+03	7,31E+02	1,40E+03	3,15E+03	1,95E+03	7,38E+01	3,08E+03
F12	6,00E-02	9,67E-02	5,25E-01	2,71E-01	1,61E-01	1,32E-02	1,62E-01	7,94E-01	3,62E-01	2,83E-02	6,51E-01
F13	2,31E-01	1,34E-01	2,43E-01	2,20E-01	1,24E-01	3,89E-02	2,04E-01	2,52E-01	2,53E-01	2,95E-02	2,32E-01
F14	2,39E-01	2,32E-01	2,78E-01	2,34E-01	2,42E-01	3,28E-01	2,25E-01	2,29E-01	2,66E-01	1,70E-01	1,72E-01
F15	3,18E+00	1,89E+00	5,67E+00	3,10E+00	2,15E+00	2,14E+00	2,56E+00	4,14E+00	4,76E+00	2,51E+00	3,51E+00
F16	9,26E+00	8,52E+00	1,11E+01	9,37E+00	8,50E+00	1,06E+01	9,15E+00	1,09E+01	9,22E+00	1,09E+01	1,29E+01
F17	1,45E+03	2,10E+02	4,61E+04	9,67E+03	1,88E+02	8,52E+02	1,06E+03	1,15E+04	9,58E+02	1,05E+03	1,90E+03
F18	1,34E+01	1,04E+01	3,32E+02	3,58E+02	5,91E+00	1,15E+02	4,99E+01	4,44E+02	2,10E+01	9,61E+01	2,83E+02
F19	2,70E+00	3,53E+00	1,33E+01	4,44E+00	3,68E+00	5,70E+00	4,31E+00	4,00E+00	3,91E+00	6,46E+00	1,61E+01
F20	1,09E+01	4,20E+00	5,00E+01	2,89E+03	3,08E+00	2,40E+01	1,26E+01	1,25E+02	8,53E+00	3,35E+01	5,10E+02

4.3. Análisis de resultados

Después de todas las mejoras y todas las posibles ejecuciones, viendo la tabla 1 y 2 en la página 7 y 7 respectivamente en la que vemos las diferentes versiones para ESA dependiendo de la dimensión, vemos que finalmente si que se consiguió una buena mejora respecto al algoritmo básico que se implementó la primera vez, por lo que si que podemos decir que gracias a las mejoras hemos conseguido un buen resultado entre ellos. He de comentar que el algoritmo consigue unos mejores resultado al aumentar las dimensiones, por lo que para dimension 30 suele trabajar mejor que para dimension 10 respecto a otros algoritmos.

En la tabla 3 y 4 de la pagina 8 y 9 respectivamente, vemos los resultados de comparar el mejor resultado de mi algoritmo con el de Evolucion Diferencial. Como indico anteriormente, mi algoritmo trabaja mejor en mas dimensiones, por lo que se puede ver en las tablas que para 10 dimensines no mejora a DE en media aunque si en algunas funciones puntuales, sin embargo, en 30 dimensiones mejora a DE globalmente en media aunque principalmente esta mejora viene dada por la función 1.

Respecto a las tablas 5 y 6 que hacen referencia a las comparativas con los algoritmos del CEC2014, podemos visualizar que por lo general, el algoritmo no es muy competitivo respecto a los demás, obtiene unos resultados que están casi en la media pero no supera al resto como para decir que es mejor que los demás, no obstante, como he dicho, obtenemos unos resultados practicamente en la media menos en algunas funciones que si que se dispara un poco más y esto hace que sea el algoritmo con mas error medio de los del concurso en el caso de 10 dimensiones, aunque en 30 mejore a algunos.

Como resumen, hemos visto y comprobado que este algoritmo puede competir con el DE que es uno de los algoritmos punteros hoy en día, sin embargo, comparandolo con los algoritmos de la competicion, obtenemos el mayor error respecto a los demás en 10 dimensiones, sin embargo, en 30 dimensiones estaríamos en 6 posicion respecto 11 algoritmos, que no es un buen resultado pero por lo menos estamos en la media cosa que en 10 dimensiones estabamos mucho peor.

Referencias

[1] Elephant Search Algorithm for optimization problems, Suash Deb; Simon Fong; Zhonghuan Tian, ieeexplore.ieee.org, https://ieeexplore.ieee.org/document/7381893/, Accedido el 20 de julio de 2018.

[2] Vitality-based Elephant Search Algorithm, Suash Deb; Simon Fong; Zhonghuan Tian; Rui Tang; Raymond Wong, ieeexplore.ieee.org, https://ieeexplore.ieee.org/document/7743275/, Accedido el 20 de julio de 2018.

[3] Optimizing self-adaptive gender ratio of elephant search algorithm by min-max strategy, Simon Fong; Zhonghuan Tian; Richard Millham; Raymond Wong, ieeexplore.ieee.org, https://ieeexplore.ieee.org/document/7603161/, Accedido el 20 de julio de 2018.