SHADE con Búsqueda Local Iterativa para Optimización Global de Alta Dimensionalidad

1º Daniel Molina
DaSCI Instituto Andaluz en Data Science and Computational Intelligence
Universidad de Granada
Granada, España
dmolina@decsai.ugr.es

2º Antonio LaTorre
DATSI, ETSIINF
Centro de Simulación Computacional
Universidad Politécnica de Madrid
Madrid, España
atorre@fi.upm.es

3º Francisco Herrera
DaSCI Instituto Andaluz en Data Science and Computational Intelligence
Universidad de Granada
Granada, España
herrera@decsai.ugr.es

***Resumen*—La optimización global es un campo de investigación de gran interés dado el gran número de problemas de la ciencia y la ingeniería que pueden formularse en dichos términos. Uno de los grandes retos en este tipo de problemas es el incremento de la dimensionalidad, ya que supone un aumento en la complejidad de los mismos. Este hecho hace de la optimización global de alta dimensionalidad un campo especialmente atractivo en nuestros días. En esta contribución proponemos un nuevo algoritmo híbrido especialmente diseñado para tratar con este tipo de problemas. La propuesta combina, de manera iterativa, una reciente variante del algoritmo de Evolución Diferencial con una búsqueda local que se escoge de entre varias estrategias disponibles. Dicha selección es dinámica y se lleva a cabo teniendo en cuenta la contribución de cada una de ellas en las anteriores fases de intensificación, de tal modo que se use la más apropiada en cada una de las fases de búsqueda del algoritmo. La experimentación se ha llevado a cabo usando el benchmark de alta dimensionalidad del CEC'2013 y los resultados demuestran que la sinergia existente entre las distintas componentes del algoritmo permite obtener unos resultados que mejoran los del actual ganador de las últimas competiciones de optimización global de alta dimensionalidad, *Multiple Offsrping Sampling*, MOS, con mejoras especialmente reseñables en los problemas de mayor complejidad.**

***Index Terms*—Optimización Global de Alta Dimensionalidad, Evolución Diferencial, Computación Memética, Hibridación.**

I. Introducción

La optimización continua es un campo de investigación muy relevante ya que muchos problemas de diversas áreas de conocimiento pueden formularse en estos términos. En este tipo de problemas, la solución puede modelarse como un vector de una determinada longitud de variables continuas en un dominio de búsqueda. Los Algoritmos Evolutivos, AEs, tales como la Evolución Diferencial, ED [1], [2], son muy útiles en la resolución de este tipo de problemas, ya que son capaces de encontrar soluciones precisas a problemas complejos sin ninguna información acerca de estos, lo cual es de gran utilidad en problemas reales [3]. Sin embargo, este tipo de algoritmos suele ser muy sensible al tamaño del problema, dado que el dominio de búsqueda aumenta exponencialmente con el número de dimensiones.

La Optimización Global de Alta Dimensionalidad, LSGO por sus siglas en inglés, es una variante concreta de este tipo de problemas en la que el tamaño del mismo suele estar por encima del millar de variables. En este contexto, la eficiencia de las técnicas de búsqueda es crucial, debido al enorme espacio de búsqueda que debe cubrirse. En los últimos años, se han organizado varias sesiones especiales en LSGO donde se han propuesto varios algoritmos específicamente diseñados para este tipo de problemas. Una de las estrategias que está recibiendo más atención consiste en particionar el problema en subproblemas de menor tamaño. Un ejemplo de esto son las diferentes estrategias de agrupamiento de variables propuestas por algunos autores [4], [5]. No obstante, la actual referencia, y ganador de las competiciones de alta dimensionalidad desde 2013, *Multiple Offspring Sampling*, MOS [6], [7], sigue una aproximación diferente: combina dinámicamente múltiples técnicas de búsqueda que se usan de manera simultánea y cuya participación en el proceso de búsqueda se ajusta de acuerdo a su rendimiento.

En esta contribución proponemos un nuevo algoritmo, SHADE con una búsqueda local iterativa, SHADE-ILS, que combina la capacidad exploratoria de una reciente variante de la ED con la potencia de intensificación de varias Búsquedas Locales, BL. En cada iteración del algoritmo, la ED se aplica para evolucionar la población de soluciones candidatas y la BL se usa para mejorar la mejor solución encontrada hasta ese momento. Las técnicas de BL se seleccionan en cada iteración de acuerdo al rendimiento relativo obtenido previamente.

Este algoritmo se basa en el que se propuso en [8], IHDELS, pero con un importante número de diferencias: la selección de la BL a aplicar en cada generación se ha mejorado. Además, se ha introducido un mecanismo de reinicio capaz de detectar el estancamiento. Por último, en esta propuesta se ha reemplazado la ED utilizada anteriormente, SaDE [9], por otra más potente, Success-History based Adaptive DE, SHADE [10].

Los resultados de este nuevo método se han comparado con los de IHDELS, mejorando los que éste obtenía. Además, SHADE-ILS mejora los resultados del ganador de las anteriores competiciones de alta dimensionalidad, MOS [6], [7] que, no había sido mejorado desde su publicación, lo cual lo convertía en el actual *estado del arte*. De esta manera, el algoritmo propuesto, SHADE-ILS, puede ser considerado el nuevo *estado del arte* en optimización continua.

El resto del documento está organizado del siguiente modo: en la sección II, el algoritmo SHADE-ILS se describe en detalle, resaltando las principales diferencias con respecto a IHDELS. En la sección III, analizamos los resultados obtenidos por nuestro algoritmo y los comparamos con los de MOS. Por último, la sección IV presenta las principales conclusiones y líneas futuras del trabajo.

II. Propuesta

En esta sección vamos a describir en detalle el algoritmo propuesto, SHADE-ILS, resaltando los cambios realizados con respecto a IHDELS. Una descripción detallada de este último puede ser consultada en [8].

Algoritmo 1 SHADE-ILS

1: $population \leftarrow random(dim, popsize)$
2: $solucion\_inicial \leftarrow (upper + lower)/2$
3: $actual\_mejor \leftarrow BL(solucion\_inicial)$
4: $mejor\_solucion \leftarrow actual\_mejor$
5: **while** $totalevals < maxevals$ **do**
6: $previo \leftarrow actual\_mejor.fitness$
7: $actual\_mejor \leftarrow SHADE(population, actual\_mejor)$
8: $improvement \leftarrow previo - actual\_mejor.fitness.$
9: *Escoge el método de BL a aplicar en esta iteración.*
10: $actual\_mejor \leftarrow BL(population, actual\_mejor)$
11: *Actualiza probabilidad de aplicar* BL.
12: **if** $mejor(actual\_mejor, mejor\_solucion)$ **then**
13: $actual\_mejor \leftarrow mejor\_solucion.$
14: **end if**
15: **if** *Debe reiniciar* **then**
16: *Reinicia y actualiza* $actual\_mejor$.
17: **end if**
18: **end while**

El algoritmo 1 muestra el esquema general de la propuesta. Como se puede observar, el algoritmo aplica, iterativamente, los métodos de ED y BL, explorando todas las variables al mismo tiempo (una diferencia importante con respecto a otros algoritmos que agrupan variables). Otra característica relevante es que el algoritmo mantiene la misma población entre fases de aplicación de la ED. Además, los parámetros del método de BL también son persistentes entre llamadas sucesivas sobre la misma solución (exceptuando tras los reinicios). Por lo tanto, el flujo del algoritmo podría resumirse en dos pasos:

- Inicialmente, se usa una técnica exploratoria para explorar el espacio de búsqueda. En nuestro caso, hemos seleccionado SHADE [10] por su simplicidad y porque auto-ajusta sus parámetros. Existe una versión que reduce el tamaño de la población, L-SHADE [11], muy popular en el campo de la optimización continua. Sin embargo, en este caso, el ajuste del tamaño de la población reduciría demasiado rápidamente sus capacidades de exploración.
- Al final de cada iteración, se escoge una BL, de entre dos métodos, para la fase de intensificación. Una de ellas es el algoritmo MTS LS-1 [12], específicamente diseñado para LSGO. La otra es el clásico L-BFGS-B [13], que usa una aproximación del gradiente para mejorar la búsqueda. Ambos métodos son complementarios: MTS LS-1 es muy rápido y apropiado para problemas separables, pero muy sensible a rotaciones. Por otro lado, L-BFGS-B es menos potente pero también menos sensible a estas.

Este *framework*, presentado en el algoritmo 1, es común a la anterior propuesta [8]. Sin embargo, como mencionamos antes, existen tres diferencias principales entre ambas propuestas: el algoritmo de ED utilizado (SHADE en lugar de SaDE) (línea 6), la selección de la BL (líneas 9 y 11) y el mecanismo de reinicio (líneas 15-17). En los párrafos siguientes vamos a detallar las principales características de la nueva propuesta.

II-A. Algoritmo exploratorio: SHADE

En este trabajo, aplicamos SHADE como el componente exploratorio. Este algoritmo presenta las siguientes ventajas:

- Tiene un mecanismo de auto-ajuste de los parámetros de la ED, CR (parámetro del cruce) y F (parámetro de la mutación), sofisticado, que permite un ajuste óptimo a cada problema. El único parámetro que hay que fijar es el tamaño de población.
- El operador de mutación tiene en cuenta soluciones anteriores almacenadas en un archivo, lo cual incrementa la diversidad de las nuevas soluciones.
- El operador de mutación está sesgado para no seleccionar siempre la mejor solución. En su lugar, selecciona aleatoriamente de entre las p mejores soluciones.

Una descripción más detallada del algoritmo SHADE se puede encontrar en [10].

En algunas versiones más recientes de SHADE, algunos autores incorporan una reducción lineal del tamaño de la población al considerar que es demasiado exploratorio [11], [14]. Sin embargo, en nuestro caso, es ésta precisamente la característica que más nos interesa, ya que disponemos de otros métodos de intensificación, por lo que no necesitamos aplicar la reducción del tamaño de la población. Es más, habiendo probado ambas versiones, los mejores resultados los hemos obtenidos con SHADE.

II-B. Selección de la BL

En IHDELS, la selección de la BL en cada iteración se llevaba a cabo con una determinada probabilidad (línea 9) P_{LS}. Este valor se inicializaba como $P_{LS} = \frac{1}{|LS|}$, donde $|LS|$ es el número de métodos de BL (2 en nuestro caso) disponibles. Además, en cada iteración (línea 11) la mejora de cada método de BL se calculaba como:

$$I_{LS} = \frac{fitness(Before_{LS}) - fitness(After_{LS})}{Before_{LS}} \quad (1)$$

A continuación, la probabilidad de seleccionar cada BL se ajustaba según el valor media de I_{LS} obtenido por cada método de BL.

En SHADE-ILS, la mejora de cada método de BL se calcula del mismo modo (ecuación 1). Sin embargo, en lugar de usar el I_{LS} medio para cada BL, selecciona la BL con mayor I_{LS}

durante la última aplicación. Este criterio es más simple y, en nuestros experimentos, más eficiente que el anterior. No sólo eso: cuando el rendimiento de un método de BL decrece rápidamente, el uso del I_{LS} medio requiere de más tiempo para detectar este cambio, mientras que el uso del I_{LS} de la anterior ejecución proporciona una adaptación más rápida.

II-C. Mecanismo de reinicio

En ocasiones, un proceso de optimización puede estancarse. En esas situaciones, una solución muy común consiste en incluir un mecanismo que detecte cuándo la mejor solución no puede ser mejorada y se encargue de reiniciar la población.

En IHDELS, el criterio de reinicio (línea 16) sólo se cumplía cuando no se conseguía mejorar durante una iteración completa. Sin embargo, en optimización continua es bastante frecuente que se siga mejorando ligeramente la mejor solución en el entorno de un atractor. Esto hacía que el mecanismo de reinicio inicialmente propuesto sólo se aplicase unas pocas veces, y en esos casos, con poco éxito.

En este trabajo, proponemos un mecanismo de reinicio que se aplica cuando, durante tres iteraciones consecutivas, el ratio de mejora (teniendo en cuenta tanto la ED como la BD) es menor del 5 %. En esos casos, el mecanismo de reinicio se aplica de la siguiente manera:

- Se selecciona, aleatoriamente, una solución sol.
- Se aplica una perturbación a sol que siga una distribución normal de media 0 y desviación típica un 10 % del dominio de búsqueda: $current_{best} = sol + rand_i \cdot 0{,}1 \cdot (b-a)$, donde $rand_i$ devuelve un número aleatorio $rand_i \in [0,1]$ y $[a,b]$ es el dominio de búsqueda.
- La población del algoritmo se reinicia aleatoriamente.
- Los parámetros adaptativos de los métodos de BL se reinician a sus valores por defecto.

Normalmente, las poblaciones se suelen reiniciar de manera aleatoria (o a partir de perturbaciones sobre la mejor solución encontrada hasta ese momento). Lo que nosotros proponemos es seleccionar una solución aleatoriamente de la población de la ED con el objetivo de que sea una solución razonablemente buena pero que no haya sido mejorada todavía por la BL. Sobre ésta, realizamos una pequeña perturbación para evitar que la siguiente población sea muy parecida a la actual.

Por otro lado, cuando la BL no es capaz de mejorar la mejor solución, también se reinician sus parámetros.

Permitir tres iteraciones completas antes de reiniciar completamente la población sigue la lógica de permitir al algoritmo que aplique ambas búsquedas locales y que éstas pueden, a su vez, reiniciar sus parámetros. Si no se producen mejoras significativas después de estas tres iteraciones, se aplica el reinicio de la población tal y como se describió con anterioridad.

III. Experimentación

Para la experimentación hemos seguido las directrices de la competición de LSGO del CEC 2013 [15]. Este benchmark se compone de 15 funciones de 1000 dimensiones y varios niveles de separabilidad, desde funciones completamente separables a completamente no-separables. La descripción detallada del benchmark puede encontrarse en [15].

La implementación del algoritmo se ha realizado en Python, y está libremente disponible[1]. Cada algoritmo se ha ejecutado 51 veces, y cada ejecución finaliza cuando se alcanza un número máximo de evaluaciones de fitness, *FEs* por sus siglas en inglés, que se fija en $3 \cdot 10^6$. Además, se registra el mejor *fitness* en distintos puntos de control (en términos de *FEs*, que son, en este caso, los siguientes: $\{1{,}2, 3{,}0, 6{,}0, 9{,}0, 12, 15, 18, 21, 24, 27, 30\} \cdot 10^5$. Sin embargo, como en anteriores competiciones sólo se tenían en cuenta tres puntos de control ($1{,}2 \cdot 10^5$, $6{,}0 \cdot 10^5$, $3{,}0 \cdot 10^6$), usaremos éstos para las comparativas con algoritmos del estado del arte.

La Tabla I recoge la configuración de parámetros usados en esta experimentación. En dicha tabla se puede comprobar cómo se ha establecido un número de evaluaciones de 50000 para cada iteración (25000 de las cuales son usadas por la ED y las otras 15000 por la BL). El resto de parámetros, compartidos entre IHDELS y SHADE-ILS, se mantienen sin cambios.

Tabla I: Configuración de parámetros

Algoritmo	Parámetro	Descripción	valor
Parámetros Compartidos	ED popsize	Tamaño de población	100
	FE_{ED}	FEs para la ED	25000
	FE_{BL}	FEs para la BL	25000
	MTS_{Istep}	Paso inicial para MTS-LS1	20
IHDELS	Freq_{BL}	Frec. actualización de prob.	10
SHADE-ILS IHDELS	$Restart_N$	Iteraciones sin mejora	3
	$Threshold$	Ratio de mejora mínimo	5 %

Los resultados de nuestra propuesta quedan recogidos en la Tabla II. Las siguientes secciones de esta contribución discuten acerca de estos resultados. En primer lugar, analizamos la contribución de las distintas componentes de nuestra propuesta, con especial interés en el mecanismo de reinicio. A continuación, comparamos los resultados con los del algoritmo IHDELS, en el que se basa esta propuesta. Por último, compararemos los resultados con los del actual ganador de las últimas competiciones de LSGO, MOS.

III-A. Influencia de las distintas componentes

En esta sección analizamos la contribución de cada una de las subcomponentes de SHADE-ILS a los resultados globales.

En primer lugar, vamos a estudiar la contribución del nuevo mecanismo de reinicio y a compararlo con el método anterior a través del análisis de varias funciones representativas (Figura 1). Hay algunas funciones, como F_4, donde el nuevo mecanismo de reinicio mejora al anterior pero que presentan curvas de convergencia con una tendencia similar. En otros casos, como en F_5, el nuevo mecanismo de reinicio permite al algoritmo una exploración completa de la zona del óptimo, lo cual facilita una mejora rápida y continuada de la mejor solución. Por último, hemos observado, para varias funciones (como

[1] https://github.com/dmolina/shadeils

Tabla II: Resultados de SHADE-ILS en el benchmark del CEC 2013

Punto de Control	Estadístico	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
1.20e+05	Mejor	1.55e+04	2.25e+03	2.01e+01	1.53e+10	1.53e+06	1.04e+06	1.75e+08	3.22e+13
	Mediana	6.07e+04	2.57e+03	2.01e+01	2.60e+10	2.50e+06	1.05e+06	3.61e+08	1.62e+14
	Peor	1.42e+05	3.07e+03	2.12e+01	7.50e+10	3.28e+06	1.06e+06	9.46e+08	1.06e+15
	Media	6.10e+04	2.65e+03	2.03e+01	3.13e+10	2.50e+06	1.05e+06	3.95e+08	2.12e+14
	Desv. Típica	3.03e+04	2.17e+02	3.92e-01	1.43e+10	3.85e+05	6.68e+03	1.54e+08	2.04e+14
6.00e+05	Mejor	0.00e+00	1.43e+03	2.00e+01	7.60e+08	1.29e+06	1.02e+06	4.72e+05	8.82e+11
	Mediana	5.07e-25	1.74e+03	2.01e+01	1.26e+09	2.28e+06	1.04e+06	7.66e+05	6.17e+12
	Peor	2.27e-22	2.44e+03	2.01e+01	4.26e+09	3.08e+06	1.05e+06	2.70e+06	1.94e+13
	Media	3.71e-23	1.80e+03	2.01e+01	1.54e+09	2.29e+06	1.04e+06	9.25e+05	6.93e+12
	Desv. Típica	6.42e-23	2.93e+02	2.24e-02	8.22e+08	3.67e+05	8.35e+03	4.73e+05	4.28e+12
3.00e+06	Mejor	0.00e+00	8.53e+02	2.00e+01	5.95e+07	1.09e+06	1.00e+06	7.65e+00	1.81e+10
	Mediana	0.00e+00	9.88e+02	2.01e+01	1.15e+08	1.41e+06	1.02e+06	5.46e+01	2.78e+11
	Peor	6.73e-23	1.21e+03	2.01e+01	3.86e+08	1.86e+06	1.04e+06	1.99e+02	1.33e+12
	Media	2.69e-24	1.00e+03	2.01e+01	1.48e+08	1.39e+06	1.02e+06	7.41e+01	3.17e+11
	Desv. Típica	1.35e-23	8.90e+01	1.12e-02	8.72e+07	2.03e+05	1.19e+04	5.46e+01	3.06e+11

Punto de Control	Estadístico	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
1.20e+05	Mejor	2.15e+08	9.32e+07	1.98e+09	2.09e+03	4.74e+09	5.15e+10	6.12e+07
	Mediana	2.88e+08	9.43e+07	4.23e+09	2.57e+03	1.30e+10	1.14e+11	8.11e+07
	Peor	3.52e+08	9.50e+07	3.46e+10	4.08e+03	2.57e+10	5.12e+11	1.58e+08
	Media	2.88e+08	9.43e+07	6.55e+09	2.67e+03	1.29e+10	1.62e+11	9.12e+07
	Desv. Típica	3.48e+07	4.90e+05	7.26e+09	4.86e+02	5.97e+09	1.08e+11	2.57e+07
6.00e+05	Mejor	1.93e+08	9.08e+07	7.85e+07	5.97e+02	1.72e+07	4.62e+07	5.76e+06
	Mediana	2.51e+08	9.30e+07	1.24e+08	1.33e+03	4.10e+07	6.42e+07	1.29e+07
	Peor	3.07e+08	9.39e+07	2.73e+08	1.89e+03	3.84e+08	1.24e+08	3.10e+07
	Media	2.50e+08	9.29e+07	1.37e+08	1.28e+03	5.68e+07	6.97e+07	1.22e+07
	Desv. Típica	3.08e+07	7.64e+05	4.74e+07	2.98e+02	7.27e+07	1.82e+07	6.30e+06
3.00e+06	Mejor	1.30e+08	9.06e+07	3.10e+05	8.53e-20	3.52e+04	4.93e+06	3.91e+05
	Mediana	1.63e+08	9.20e+07	4.50e+05	3.99e+00	8.28e+04	5.75e+06	6.09e+05
	Peor	1.94e+08	9.31e+07	1.43e+06	2.99e+02	3.31e+05	6.39e+06	1.56e+06
	Media	1.64e+08	9.18e+07	5.11e+05	6.18e+01	1.00e+05	5.76e+06	6.25e+05
	Desv. Típica	1.57e+07	6.93e+05	2.25e+05	1.04e+02	7.19e+04	3.76e+05	2.40e+05

F_9 o F_{12}) que el nuevo mecanismo de reinicio es capaz de incrementar significativamente la velocidad de convergencia.

A continuación, vamos a comparar la influencia de cada una de las subcomponentes de manera individual. La Tabla III presenta los resultados partiendo del algoritmo IHDELS y aplicando los distintos cambios descritos en la Sección II:

- IHDELS: Versión original de IHDELS propuesta en 2015 [8], que usa SaDE y el mecanismo de reinicio original.
- IHDELS-S: Algoritmo que reemplaza SaDE por SHADE pero manteniendo el mismo mecanismo de reinicio.
- IHDELS-R: Algoritmo con el nuevo mecanismo de reinicio pero manteniendo SaDE en lugar de SHADE.
- IHDELS-SR: Propuesta final, usando SHADE en lugar de SaDE y el nuevo mecanismo de reinicio.

En el análisis anterior no incluimos el nuevo mecanismo de selección de la BL porque las diferencias no eran evidentes. No obstante, decidimos mantener dicho cambio ya que reduce la complejidad del algoritmo al eliminar el parámetro $Freq_{BL}$ sin deterioro de la calidad de las soluciones.

Analizando los resultados de la Tabla III se pueden extraer las siguientes conclusiones:

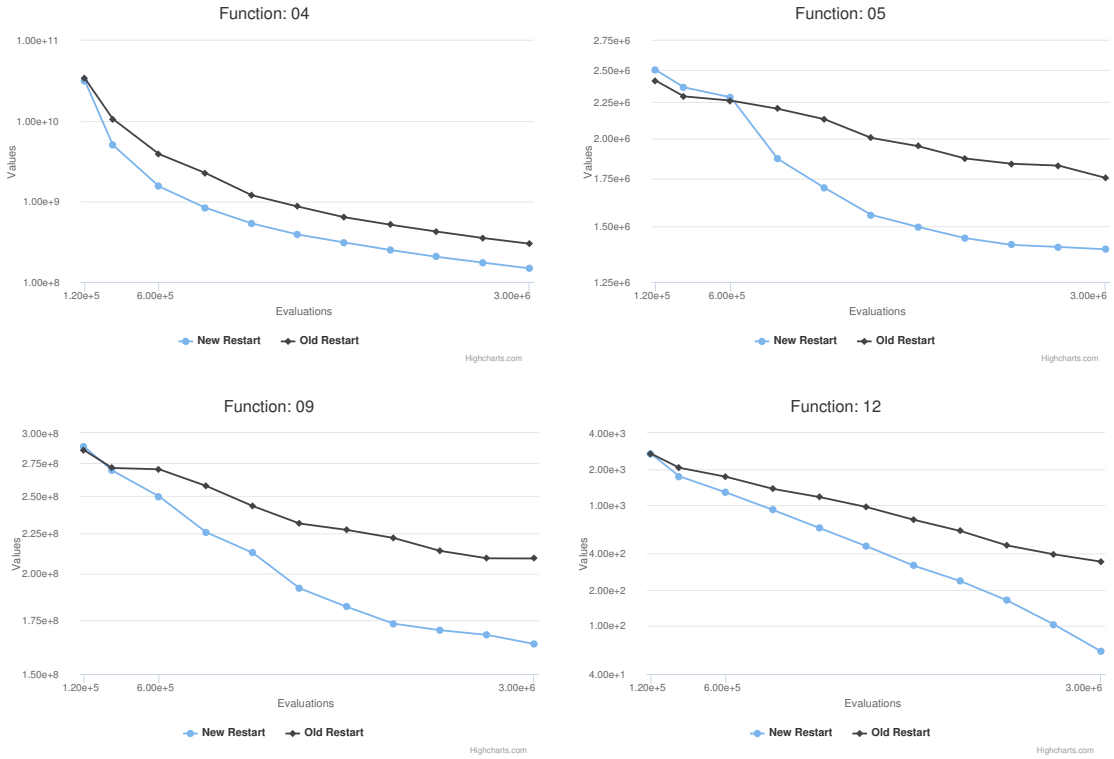
- El cambio en el mecanismo de reemplazo tiene un efecto mayor que el de la componente de ED.
- El uso de SHADE en lugar de SaDE mejora los resultados significativamente, independientemente del mecanismo de reinicio utilizado, especialmente en el caso de las funciones no-separables y con solapamiento.

Tabla III: Resultados para las diferentes combinaciones de las componentes del algoritmo tras $3 \cdot 10^6$ FEs

Func.	IHDELS-SR (SHADE-ILS)	IHDELS-R	IHDELS-S	IHDELS original
F_1	2.69e-24	1.21e-24	1.76e-28	**4.80e-29**
F_2	**1.00e+03**	1.26e+03	1.40e+03	1.27e+03
F_3	2.01e+01	2.01e+01	2.01e+01	**2.00e+01**
F_4	**1.48e+08**	1.58e+08	2.99e+08	3.09e+08
F_5	**1.39e+06**	3.07e+06	1.76e+06	9.68e+06
F_6	**1.02e+06**	1.03e+06	1.03e+06	1.03e+06
F_7	**7.41e+01**	8.35e+01	2.44e+02	3.18e+04
F_8	**3.17e+11**	3.59e+11	8.55e+11	1.36e+12
F_9	**1.64e+08**	2.48e+08	2.09e+08	7.12e+08
F_{10}	**9.18e+07**	9.19e+07	9.25e+07	9.19e+07
F_{11}	5.11e+05	**4.76e+05**	5.20e+05	9.87e+06
F_{12}	**6.18e+01**	1.10e+02	3.42e+02	5.16e+02
F_{13}	**1.00e+05**	1.34e+05	9.61e+05	4.02e+06
F_{14}	**5.76e+06**	6.14e+06	7.40e+06	1.48e+07
F_{15}	**6.25e+05**	8.69e+05	1.01e+06	3.13e+06
Mejor	12	1	0	2

- La combinación de SHADE y el nuevo mecanismo de reinicio es significativamente mejor que cualquiera de las otras combinaciones, lo cual pone de manifiesto que existe una interacción positiva entre ambas mejoras.
- El algoritmo propuesto obtiene los mejores resultados en 12 de las 15 funciones del benchmark, siendo las

Figura 1: Comparativa del error medio para el mecanismo de reinicio original y el nuevo



diferencias muy pequeñas en las otras 3.

En resumen, las diferentes componentes de SHADE-ILS, especialmente el nuevo mecanismo de reinicio y el uso de SHADE en lugar de SaDE, son capaces de mejorar claramente los resultados de IHDELS. Es más, la combinación de todos estos nuevos componentes obtiene los mejores resultados, que son significativamente mejores que los de cualquier otra combinación posible.

III-B. Comparación de SHADE-ILS e IHDELS

En la Tabla III se puede observar que SHADE-ILS es mejor que IHDELS en 13 de las 15 funciones del benchmark (con diferencias mínimas en las otras dos). No sólo eso: las mejoras de SHADE-ILS son especialmente importantes en las funciones más complejas, donde obtiene un error al menos un orden de magnitud más pequeño en muchas de ellas (por ejemplo, en las funciones $F_5, F_7, F_8, F_{11}, F_{12}, F_{13}, F_{14}$, o F_{15}). Como acabamos de discutir, las mejoras son debidas a la combinación de ambas componentes. Por último, el cambio en el mecanismo de selección de la BL elimina un parámetro del algoritmo (Feq_{LS}) y lo hace, por tanto, más sencillo, mientras sigue manteniendo un comportamiento robusto.

III-C. Comparación de SHADE-ILS y MOS

En esta sección vamos a comparar SHADE-ILS con MOS, el algoritmo de referencia en LSGO y, desde 2013, el ganador de todas las competiciones en alta dimensionalidad. Ningún otro algoritmo ha sido capaz de mejorar sus resultados hasta ahora, por lo que se puede considerar el actual *estado-del-arte* y, por tanto, la referencia clara que aspiramos a superar.

En la Tabla IV se presentan los resultados de SHADE-ILS y MOS para los distintos puntos de control ($1{,}2\cdot10^5$, $6\cdot10^5$, y $3\cdot10^6$), de los que podemos extraer las siguientes conclusiones:

- MOS obtiene los mejores resultados tras $1{,}2\cdot10^5$ FEs, éstos se igualan tras $6\cdot10^5$ FEs y, para el máximo número de FEs, $3\cdot10^6$, SHADE-ILS obtiene los mejores resultados en 10 de las 15 funciones.
- MOS sigue obteniendo los mejores resultados en las funciones separables (f_1-f_3), mientras que SHADE-ILS es mejor en funciones más complejas: con la excepción de las funciones f_6 y f_{10}, SHADE-ILS es significativamente mejor en el resto de las funciones.
- SHADE-ILS es muy competitivo en un número mayor de funciones, especialmente en las solapadas y no-separables. No sólo mejora los resultados de MOS en muchos casos, sino que sus resultados son, a menudo, al menos un orden de magnitud más pequeños.

IV. Conclusiones

En este artículo hemos propuesto un nuevo algoritmo de optimización especialmente diseñado para Optimización Global de Alta Dimensionalidad, SHADE-ILS, que combina la capacidad exploratoria de una ED adaptativa con la capacidad de intensificación de dos métodos de BL. En cada iteración del algoritmo, éste evoluciona una población con SHADE y, a continuación, selecciona la BL con el mejor rendimiento

Tabla IV: Resultados obtenidos por SHADE-ILS y MOS para los distintos puntos de control: FEs=$1{,}2 \cdot 10^5$, FEs=$5{,}0 \cdot 10^5$ y FEs=$3{,}0 \cdot 10^6$.

	1.20e+05		6.00e+05		3.00e+06	
Función	SHADE-ILS	MOS	SHADE-ILS	MOS	SHADE-ILS	MOS
F_1	**6.10e+04**	2.71e+07	**3.71e-23**	3.48e+00	2.69e-24	**0.00e+00**
F_2	2.65e+03	**2.64e+03**	1.80e+03	**1.78e+03**	1.00e+03	**8.32e+02**
F_3	2.03e+01	**7.85e+00**	2.01e+01	**1.33e-10**	2.01e+01	**9.17e-13**
F_4	**3.13e+10**	3.47e+10	**1.54e+09**	2.56e+09	**1.48e+08**	1.74e+08
F_5	**2.50e+06**	6.96e+06	**2.29e+06**	6.95e+06	**1.39e+06**	6.94e+06
F_6	1.05e+06	**3.11e+05**	1.04e+06	**1.48e+05**	1.02e+06	**1.48e+05**
F_7	3.95e+08	**3.46e+08**	**9.25e+05**	8.19e+06	**7.41e+01**	1.62e+04
F_8	**2.12e+14**	3.72e+14	**6.93e+12**	8.41e+13	**3.17e+11**	8.00e+12
F_9	**2.88e+08**	4.29e+08	**2.50e+08**	3.84e+08	**1.64e+08**	3.83e+08
F_{10}	9.43e+07	**1.16e+06**	9.29e+07	**9.03e+05**	9.18e+07	**9.02e+05**
F_{11}	6.55e+09	**3.13e+09**	**1.37e+08**	8.05e+08	**5.11e+05**	5.22e+07
F_{12}	**2.67e+03**	1.16e+04	**1.28e+03**	2.20e+03	**6.18e+01**	2.47e+02
F_{13}	1.29e+10	**8.37e+09**	**5.68e+07**	8.10e+08	**1.00e+05**	3.40e+06
F_{14}	1.62e+11	**4.61e+10**	**6.97e+07**	2.03e+08	**5.76e+06**	2.56e+07
F_{15}	9.12e+07	**1.45e+07**	1.22e+07	**6.26e+06**	**6.25e+05**	2.35e+06

relativo durante la última fase de activación para que mejora la mejor solución encontrada por el algoritmo poblacional. Además, se ha incorporado un mecanismo de reinicio que permite al algoritmo explorar otras regiones del espacio de búsqueda cuando ésta se queda estancada (cuando la mejora relativa es pequeña durante varias iteraciones consecutivas).

En la sección de experimentación, hemos puesto a prueba SHADE-ILS usando el benchmark propuesto para la competición de alta dimensionalidad del CEC 2013. En primer lugar, hemos comprobado que cada una de las modificaciones llevadas a cabo sobre IHDELS contribuye significativamente a la mejora de los resultados. A continuación, hemos comparado la propuesta con el algoritmo de referencia en alta dimensionalidad, MOS. En esta comparación, SHADE-ILS consigue los mejores resultados, batiendo a MOS por primera vez desde 2013. Esta comparación revela que SHADE-ILS es especialmente competitivo en las funciones más complejas, con componentes solapadas y no separables, donde no sólo obtiene el error más pequeño en la mayoría de las funciones, sino que también consigue reducir el error en al menos un orden de magnitud en muchas de ellas.

Como trabajo futuro, vamos a probar otras técnicas para acelerar la convergencia del algoritmo y obtener mejores resultados en más funciones no-separables.

Agradecimientos

Este trabajo ha sido financiado por ayudas del Ministerio de Ciencia, Innovación y Universidades (TIN2014-57481-C2-2-R, TIN2016-8113-R, TIN2017-83132-C2-2-R y TIN2017-89517-P) y de la Junta de Andalucía (P12-TIC-2958).

Referencias

[1] K. V. Price, R. M. Rainer, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, 2005.

[2] S. Dasa, S. Maity, B.-Y. Qu, and P. Suganthan, "Real-parameter evolutionary multimodal optimization-a survey of the state-of-the-art," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 71–78, 2011.

[3] N. Xiong, D. Molina, M. L. Ortiz, and F. Herrera, "A walk into metaheuristics for engineering optimization: principles, methods and recent trends," *International Journal of Computational Intelligence Systems, 2015*, vol. 8, no. 4, pp. 606–636, June 2015.

[4] M. N. Omidvar, X. Li, and X. Yao, "Cooperative Co-evolution with delta grouping for large scale non-separable function optimization," in *2010 IEEE Congress on Evolutionary Computation (CEC 2010)*. IEEE, 2010, pp. 1762–1769.

[5] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 378–393, June 2014.

[6] A. LaTorre, S. Muelas, and J. M. Peña, "Large Scale Global Optimization: Experimental Results with MOS-based Hybrid Algorithms," in *2013 IEEE Congress on Evolutionary Computation (CEC 2013)*, Cancún, Mexico, 2013, pp. 2742–2749.

[7] A. LaTorre, S. Muelas, and J. Peña, "A Comprehensive Comparison of Large Scale Global Optimizers," *Information Sciences*, vol. 316, pp. 517–549, 2014.

[8] D. Molina and F. Herrera, "Iterative hybridization of DE with local search for the CEC'2015 special session on large scale global optimization," in *2015 IEEE Congress on Evolutionary Computation (CEC 2015)*. IEEE, 2015, pp. 1974–1978.

[9] A. Qin, V. Huang, and P. Suganthan, "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

[10] R. Tanabe and A. Fukunaga, "Evaluating the performance of shade on cec 2013 benchmark problems," in *2013 IEEE Congress on Evolutionary Computation*, June 2013, pp. 1952–1959.

[11] R. Poláková, J. Tvrdík, and P. Bujok, "L-shade with competing strategies applied to cec2015 learning-based test suite," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 4790–4796.

[12] L. Y. Tseng and C. Chen, "Multiple Trajectory Search for Large Scale Global Optimization," in *2008 IEEE Congress on Evolutionary Computation (CEC 2008)*. IEEE Press, Jun. 2008, pp. 3052–3059.

[13] J. L. Morales and J. Nocedal, "Remark on algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization," *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 7:1–7:4, Dec. 2011.

[14] J.Brest and B. M.Sepesy Mačec, "iL-SHADE: Improved L-SHADE Algorithm for Single Objective Real-Parameter OptimizationTesting United Multi-operator Evolutionary Algorithms-II on Single Objective," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 1188–1195.

[15] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Quin, "Benchmark functions for the CEC'2013 special session and competition on large scale global optimization," RMIT University, Tech. Rep., 2013.