Hibridación Iterativa de DE con Búsqueda Local con reinicio para problemas de alta dimensionalidad

Daniel Molina[1] and Francisco Herrera[2]

[1] Universidad de Cádiz, daniel.molina@uca.es
[2] Universidad de Granada, herrera@decsai.ugr.es

Resumen En este trabajo presentamos un nuevo algoritmo para problemas de alta dimensionalidad mediante la hibridación de un *Differential Evolution* con una aplicación de Búsqueda Local (alternando entre dos métodos diferentes). De esta manera se combina la exploración del *Differential Evolution* con el grado de explotación de la Búsqueda Local, lo cual es especialmente importante en problemas de alta dimensionalidad, ya que permite acelerar la convergencia en problemas que por su dimensionalidad ofrecen un espacio de búsqueda muy amplio.
Una versión preliminar de esta propuesta fue presentada en el Congreso IEEE de Computación Evolutiva de 2015, la principal novedad en este trabajo es la incorporación de un método efectivo de reinicio, y una más adecuada alternancia entre los métodos de Búsqueda Local. Se han realizado comparativas utilizando el *benchmark* propuesto para dicha competición [3]. Los resultados muestran que los cambios propuestos mejoran los resultados de forma muy significativa. También el algoritmo es comparado con el mejor algoritmo para alta dimensionalidad, MOS, mostrándose muy competitivo, e incluso mejor que MOS en las funciones más complejas.

Keywords: differential evolution, metaheurísticas, optimización continua, alta dimensionalidad, algoritmos híbridos

1. Introducción

Muchos problemas reales de optimización de distintos dominios (ingeniería, data mining, etc.) requieren una optimización de una función continua. Para resolver este tipo de problemas destacan los algoritmos evolutivos (AEs) [1], ya que son algoritmos muy eficientes capaces de obtener soluciones precisas en problemas complejos sin necesidad de información específica sobre dichos problemas, algo especialmente importante en problemas reales.

En los últimos años han surgido algoritmos híbridos que se mantienen como los mejores algoritmos en optimización continua, como ICMAES-ILS [4] en problemas de dimensionalidad media y MOS [2] en problemas de alta dimensionalidad. Estas propuestas se caracterizan por alternar en su ejecución distintos

algoritmos (usualmente con un mecanismo auto-adaptativo) para obtener buenos resultados, y en los que al menos ello de ellos es un método de Búsqueda Local (BL) capaz de explorar de forma bastante exhaustiva el entorno alrededor de las mejores soluciones. Este tipo de diseño es especialmente útil en problemas de alta dimensionalidad, ya que poseen un espacio de búsqueda muy amplio que requiere un grado de explotación de las soluciones encontradas que un método de BL puede ofrecer.

En un trabajo anterior se presentó un nuevo algoritmo con optimización continua, el *Iterative Hybrid DE with LS*, (IHDELS) [5]. Dicho algoritmo combina de forma iterativa un Differential Evolution (DE) [8] y un método de BL. En cada iteración se ejecutan ambos con la idea de que se complementen entre sí. Además, ambos componentes mantienen su estado entre iteraciones: En el caso del DE, la población, y en los métodos de BL, los parámetros adaptativos.

El algoritmo anterior obtenía buenos resultados, pero incorporaba un modelo de reinicio inadecuado, ya que apenas se ejecutaba, y cuando lo hacía no servía para mejorar los resultados. Esto suponía que aún obteniendo resultados competitivos, había un margen de mejora, especialmente en los problemas más complejos. En este trabajo presentamos una modificación del IHDELS que arregla dichos aspectos. La mejora se compone de dos partes: Por un lado, un nuevo mecanismo de reinicio que se ejecuta tras varias iteraciones sin una mejora significativa. Por otro lado, se ha mejorado el mecanismo de elección de la BL a aplicar. Se han comparado el algoritmo modificado y el modelo original usando el *benchmark* específico de alta dimensionalidad [3], y los resultados muestran que las modificaciones mejoran los resultados, especialmente en los problemas más complejos.

Este trabajo está estructurado de la siguiente forma: En la Sección 2 se introduce el IHDELS mejorado, remarcando las diferencias entre el algoritmo original y el nuevo algoritmo propuesto. En la Sección 3, se describe brevemente el *benchmark* usado, y se comparan los resultados obtenidos por nuestra propuesta con el algoritmo original, y otro algoritmo de referencia. Finalmente, en la Sección 4, se resumen las principales conclusiones obtenidas.

2. Propuesta: IHDELS modificado

En esta sección describimos la propuesta presentada, destacando los cambios respecto al algoritmo original IHDELS. Para mayor información, se puede consultar [5].

El Algoritmo 1 muestra el esquema general. Se puede ver que es un algoritmo que aplica de forma iterativa un DE y un método de BL, en el que se exploran todas las variables al mismo tiempo. Se mantienen entre las distintas iteraciones la población del DE, y también los parámetros de la Búsqueda Local (excepto cuando se reinicia el algoritmo).

Como algoritmo exploratorio utiliza el clásico SaDE[7], ya que es un algoritmo sencillo que auto-adapta sus parámetros. A la hora de aplicar la BL en cada iteración escoge aplicar uno de entre dos métodos distintos: Uno es el algoritmo

Algoritmo 1 Esquema general del IHDELS

1: $population \leftarrow random(dim, popsize)$
2: $initial\_solution \leftarrow (upper + lower)/2$
3: $current\_best \leftarrow BL(initial\_solution)$
4: $best\_solution \leftarrow current\_best$
5: **mientras** $totalevals < maxevals$ **hacer**
6: $current\_best \leftarrow SaDE(population, current\_best)$
7: $previous \leftarrow current\_best.fitness$
8: $improvement \leftarrow previous - current\_best.fitness.$
9: Selecciona la BL a aplicar en esta iteración.
10: $current\_best \leftarrow BL(population, current\_best)$
11: Actualiza probabilidad de aplicar la BL para siguientes iteraciones.
12: **si** $better(current\_best, best\_solution)$ **entonces**
13: $current\_best \leftarrow best\_solution.$
14: **fin si**
15: **si** Debe de reiniciar **entonces**
16: Reinicia y actualiza $current\_best$.
17: **fin si**
18: **fin mientras**

MTS LS-1 [9], un algoritmo especialmente diseñado para alta dimensionalidad, y el otro es el clásico L-BFGS-B [6] que utiliza una aproximación del gradiente para mejorar la búsqueda. Estos métodos son complementarios: El MTS es muy rápido y adecuado para problemas separables, pero es sensible a rotaciones. En cambio, el L-BFGS-B es menos potente, pero menos sensible a rotaciones.

Este esquema es común tanto al algoritmo original como al algoritmo modificado. Se distinguen uno y otro en dos aspectos: en el criterio de selección de la BL (líneas 9 y 11), y en el mecanismo de reinicio (líneas 15-17). A continuación detallamos las características del IHDELS original y de la propuesta, el IHDELS mejorado.

2.1. IHDELS original

En el IHDELS original la selección del método de BL a aplicar en cada iteración se establece usando un conjunto de BLs en las que para cada BL se guarda una determinada probabilidad, y en cada iteración se elige aleatoriamente cuál utilizar según dichas probabilidades (paso 9). Inicialmente, la Probabilidad de cada BL (P_{BL}) es igual para cada una ($P_{BL} = \frac{1}{|BL|}$), y por cada iteración (línea 11) se almacena la mejora obtenida en *fitness* por la BL. Después de $Frec_{BL}$ iteraciones (se usa $Frec_{BL} = 10$), se recalculan las probabilidades conforme a la ecuación siguiente:

$$P_{BL_M} = \frac{I_{BL_M}}{\sum_{m \in BL} I_{BL_m}}$$

En donde I_{BL_M} se define como

$$I_{BL_M} = \sum_{i=1}^{Frec_{BL}} \text{Mejora obtenida por } BL_M$$

En el modelo original, se desarrolló un mecanismo de reinicio (línea 16) que se ejecutaba en el caso de que en una iteración no se consiguiese ninguna mejora, en cuyo caso:

- Se seleccionaba aleatoriamente una solución de la población como la nueva solución *current_best*.
- Se reiniciaban aleatoriamente la población de la DE.
- Se reiniciaban los parámetros adaptativos de las BLs a sus valores originales.

Sin embargo, se comprobó durante la parte experimental que este modelo de reinicio no mejoraba los resultados. El motivo es que se requería que en una iteración la mejor solución no mejorase nada y se comprobó que casi siempre se producía una mejora aunque muy reducida, por lo que no se ejecutaba prácticamente nunca el mecanismo de reinicio. Además, en los pocos casos que se aplicó nunca sirvió para mejorar los resultados.

2.2. IHDELS mejorado propuesto

El modelo propuesto de selección de BL es el siguiente: Inicialmente siempre ejecuta una vez cada BL, y guarda para cada una de ellas el ratio de mejora $Ratio_{BL}$ definido según la ecuación siguiente:

$$Ratio_{BL_M} = \frac{Fitness_{anterior} - Fitness_{nuevo}}{Fitness_{anterior}}$$

En cada iteración se aplica aquel método de BL que haya conseguido un ratio $Ratio_{BL_M}$ mayor en su última aplicación (en cada aplicación se actualiza dicho ratio). Este modelo es más sencillo, y eficiente que el anterior.

Como mecanismo de reinicio, se define un valor umbral entre 0 y 1, que recibe como parámetro, y que sirve para identificar cuándo una mejora es suficiente o no: para que sea considerada suficiente el ratio de mejora debe de ser superior o igual que dicho valor umbral.

- Si tras aplicar el DE y la BL durante las $Reinicio_N$ iteraciones anteriores no se ha mejorado lo suficiente se reinicia el algoritmo.
- Si en una iteración el DE no mejora nada, se reinicia la población del DE.
- Si en una ejecución un método de BL no mejora nada, se reinician sus parámetros adaptativos.

El reinicio completo del algoritmo (en el primer caso anterior) supone que:

- $current\_best[i] \leftarrow best\_found\_solution[i] + rand_i \cdot 0{,}1 \cdot (b - a)$ en donde $rand_i$ devuelve un número aleatorio entre [-0.05, 0.05], y $[a, b]$ es el dominio de búsqueda.

- Se reinicia aleatoriamente la población del DE.
- Se reinician los parámetros adaptativos de todas los métodos de BL a sus valores originales.

3. Estudio Experimental

La experimentación se han realizado usando el *benchmark* y las condiciones experimentales indicadas en la sesión especial de optimización global de alta dimensionalidad, *Large Scale Global Optimization*, [3]. Este *benchmark* está compuesto por 15 funciones de optimización continua, de dimensión 1000 y con diferentes grados de separabilidad, desde funciones totalmente separables a funciones completamente no separables:

- Funciones totalmente separables: $(f_1 - f_3)$.
- Funciones parcialmente separables: Con un componentes separable $(f_4 - f_7)$ o sin componentes separables $(f_8 - f_{11})$.
- Funciones con solapamiento: $(f_{12} - f_{14})$.
- Funciones no separables: (f_{15}).

Para obtener mayor información sobre las funciones, se puede consultar [3].

Cada algoritmo es ejecutado sobre la misma función 25 veces, y cada ejecución termina cuando se alcanza un numero máximo de evaluaciones, *fitness evaluations, FEs*, igual a $3 \cdot 10^6$. Además, se mide el mejor *fitness* para los siguientes valores de FEs: $1{,}2 \cdot 10^5$; $3{,}0 \cdot 10^5$; y $6{,}0 \cdot 10^6$.

Los parámetros utilizados se indican en la Tabla 1. Se puede observar que en cada iteración se realizan 50000 evaluaciones (25000 para el DE y otras 25000 para el método de BL). Los parámetros usados son los mismos tanto para el algoritmo IHDELS como para el propuesto, por lo que no se han ajustado en cada caso (posiblemente se podría obtener mejores valores adaptándolo).

Tabla 1. Valores de los parámetros usados en IHDELS

Algoritmo	Parámetro	Descripción	valor
Parámetros Comunes	DE popsize	Tamaño de la población	100
	FE_{DE}	FEs de DE para cada iteración	25000
	FE_{BL}	FEs de la BL para cada iteración	25000
	MTS_{Istep}	Salto inicial del MTS	20
Original IHDELS	Frec_{BL}	Frecuencia de actualización de las probabilidades de BL	10
Mejorado IHDELS	$Reinicio_N$	Número de ejecuciones sin mejora	3
	$Threshold$	Ratio de mejora mínimo	1 %

Tabla 2. Valores medios de IHDELS original y IHDELS mejorado para FEs=$3 \cdot 10^6$

Grupo de Funciones	Función	Original	Propuesta
Separables	F_1	**4.80e-29**	4.53e-24
	F_2	1.27e+03	**1.26e+03**
	F_3	**2.00e+01**	2.01e+01
Parcialmente Separables	F_4	3.09e+08	**2.55e+08**
	F_5	**9.68e+06**	1.18e+07
	F_6	1.03e+06	1.03e+06
	F_7	3.18e+04	**5.83e+02**
Parcialmente Separables II	F_8	**1.36e+12**	2.19e+12
	F_9	7.12e+08	**5.14e+08**
	F_{10}	9.19e+07	**9.16e+07**
	F_{11}	9.87e+06	**2.77e+06**
Con solapamiento	F_{12}	5.16e+02	**1.32e+02**
	F_{13}	4.02e+06	**6.29e+05**
	F_{14}	1.48e+07	**1.01e+07**
No separable	F_{15}	3.13e+06	**1.35e+06**

3.1. Comparando IHDELS original con IHDELS mejorado

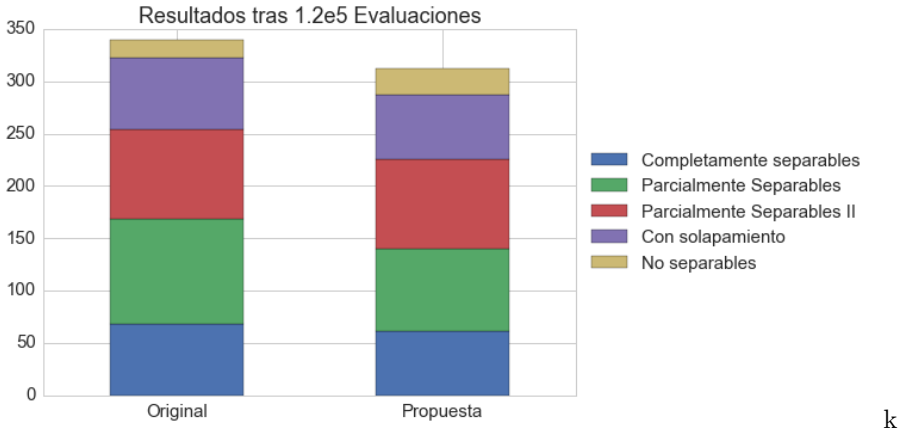
En primer lugar, comparamos los valores medios finales tanto del algoritmo original como del algoritmo propuesto. Como se puede ver en la Tabla 2, el algoritmo propuesto ofrece mejores resultados en diez de las quince funciones. Además, la propuesta ofrece claramente mejores resultados en las últimas 6 funciones, que son las más complejas.

Parar mostrar cómo evoluciona para cada tipo de función y nivel de evaluaciones, aplicamos el criterio usado en las comparativas de las sesiones especiales de alta dimensionalidad [3], en el que para cada algoritmo y función se ordenan los algoritmos por su *fitness* medio, y se le asigna a cada uno una puntuación en base a su ranking (siguiendo el criterio de la Fórmula 1: 21 para el primero, 18 para el segundo, etc.). La Figura 1 muestra en diagramas de barras acumulados las puntuaciones para distintos valores de *FEs*: $1{,}2 \cdot 10^5$, $6 \cdot 10^5$, $3 \cdot 10^6$. Los resultados son mejores cuanto mayor sean las puntuaciones. Se puede observar que la propuesta mejora: Aunque inicialmente converge más lentamente (en particular, en las funciones separables), acaba obteniendo mejores resultados en la práctica mayoría de funciones, en particular en las funciones más complejas.

3.2. Comparando la propuesta con algoritmo MOS

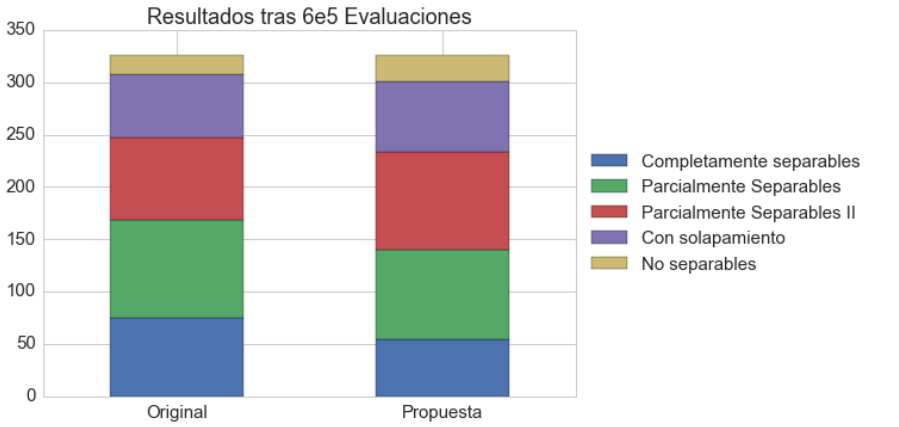
Comparamos con el algoritmo MOS [2], ya que considerado el mejor algoritmo para alta dimensionalidad, al ser el ganador de las distintas competiciones de alta dimensionalidad del *IEEE Congress on Evolutionary Computation*.

Figura 1. Comparando el IHDELS original y el mejorado para distintos valores de FEs

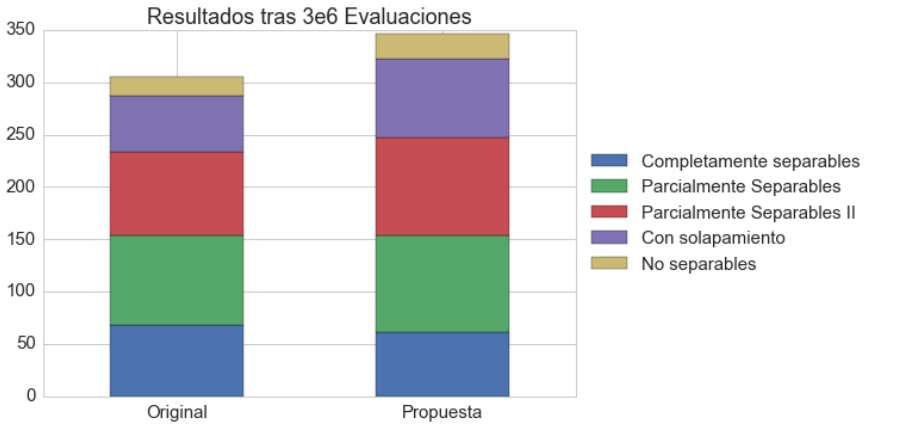


k

(a) FEs=$1{,}2 \cdot 10^5$



(b) FEs=$6{,}0 \cdot 10^5$



(c) FEs=$3{,}0 \cdot 10^6$

Tabla 3. Valores medios de IHDELS mejorado y MOS para FEs=$3 \cdot 10^6$

Grupo de Funciones	Función	Propuesta	MOS
Separables	F_1	4.53e-24	**0.00e+00**
	F_2	1.26e+03	**8.36e+02**
	F_3	2.01e+01	**9.10e-13**
Parcialmente Separables	F_4	2.55e+08	**1.56e+08**
	F_5	1.18e+07	**6.79e+06**
	F_6	1.03e+06	**1.39e+05**
	F_7	**5.83e+02**	1.62e+04
Parcialmente Separables II	F_8	**2.19e+12**	8.08e+12
	F_9	5.14e+08	**3.87e+08**
	F_{10}	9.16e+07	**1.18e+06**
	F_{11}	**2.77e+06**	4.48e+07
Con solapamiento	F_{12}	**1.32e+02**	2.46e+02
	F_{13}	**6.29e+05**	3.30e+06
	F_{14}	**1.01e+07**	2.42e+07
No separable	F_{15}	**1.35e+06**	2.38e+06

En la Tabla 3 se muestran los resultados obtenidos. Se observa que para las funciones más complejas, con solapamiento y no separables, nuestra propuesta no sólo mejora al IHDELS original, si no también a MOS en F_8 y en $F_{11} - F_{15}$.

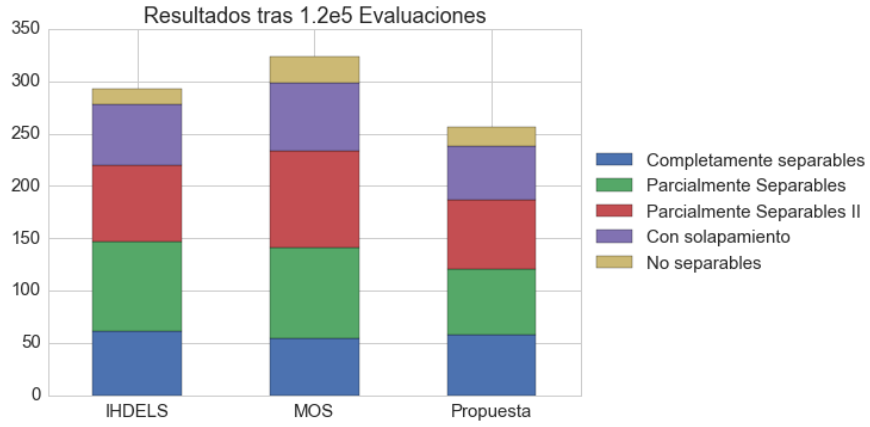
Visualmente, la Figura 2 muestra el resultado de comparar conjuntamente la mejora propuesta con el IHDELS original y MOS. Se puede observar que, a diferencia del algoritmo original, que conforme avanzaba el algoritmo obtenía peores resultados respecto a MOS, el modelo propuesto se mantiene mucho más cercano en resultados al MOS hasta el final. Además, en problemas con solapamiento o no separables es claramente mejor. Por tanto, IHDELS con reinicio es mejor en problemas con mayor grado de solapamiento, mientras que MOS se mantiene como mejor en problemas con mayor grado de separabilidad.

4. Conclusiones

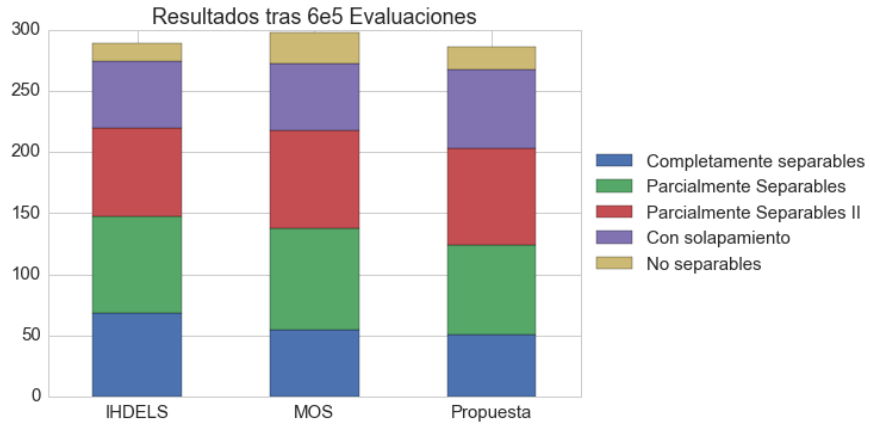
En este trabajo proponemos un nuevo algoritmo especialmente para alta dimensionalidad, mediante la hibridación iterativa de un DE y una BL, combinando la capacidad explorativa del DE con la capacidad de explotación de la BL para mejorar los resultados.

Este trabajo se basa en un trabajo previo que se ha mejorado modificando el mecanismo de selección de la BL a aplicar en cada caso, y en el mecanismo de reinicio. En cada iteración se elige el método de BL de entre dos, el MTS, y el L-BFGS-B, aplicando en cada iteración aquél que consiguiese un mejor ratio de mejora en su última ejecución. El mecanismo de reinicio se activa cuando

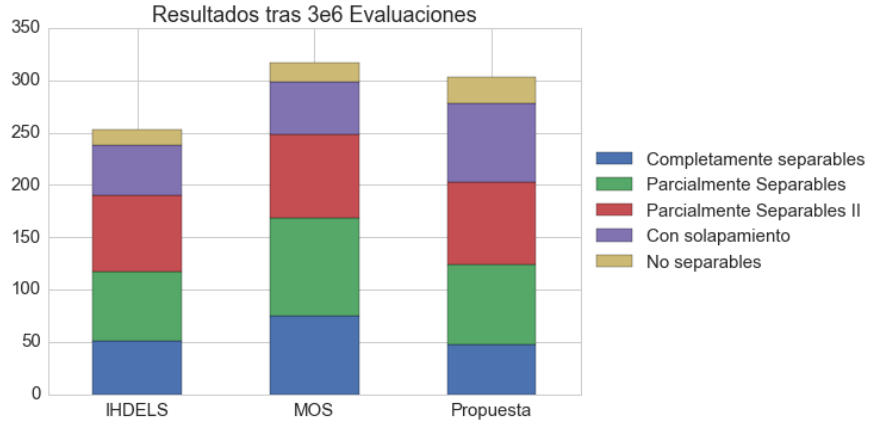
Figura 2. Comparando el IHDELS original y MOS con la propuesta, para distintos valores de FEs



(a) FEs=$1{,}2 \cdot 10^5$



(b) FEs=$6{,}0 \cdot 10^5$



(c) FEs=$3{,}0 \cdot 10^6$

durante varias iteraciones el ratio global de mejora obtenido en dicha iteración es peor o igual que un ratio predefinido.

Hemos evaluado la propuesta usando el *benchmark* propuesto para la Sesión Especial de Alta dimensionalidad del *IEEE Congress on Evolutionary Computation* [3], comparando tanto con el algoritmo original como con MOS, el actual estado del arte para optimización de alta dimensionalidad. Los resultados no sólo prueban que los cambios suponen mejoras significativas en la mayoría de las funciones, si no también que la propuesta es competitiva con respeto a MOS. Además, para las funciones más complejas, funciones con solapamiento y no separables, nuestra propuesta no sólo mejora al IHDELS original, si no también a MOS. Se plantea como trabajo futuro mejorarlo en problemas con mayor grado de separabilidad.

Agradecimientos

Este trabajo ha sido apoyado por los proyectos nacionales TIN2012-37930-C02-01, TIN2013-47210-P y los proyectos regionales P08-TIC-04173, P12-TIC-2958.

Referencias

1. T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, 1997.
2. A. LaTorre, S. Muelas, and J.-M. Pena. Large scale global optimization: Experimental results with mos-based hybrid algorithms. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2742–2749, June 2013.
3. X. Li, K. Tang, M. Omidvar, Z. Yang, and K. Qin. Benchmark functions for the cec'2013 special session and competition on large scale global optimization. Technical report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia, 2013.
4. Tianjun Liao and T. Stutzle. Benchmark results for a simple hybrid algorithm on the cec 2013 benchmark set for real-parameter optimization. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1938–1944, June 2013.
5. D. Molina and F. Herrera. Iterative hybridization of de with local search for the cec'2015 special session on large scale global optimization. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 1974–1978, June 2015.
6. José Luis Morales and Jorge Nocedal. Remark on algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization. *ACM Trans. Math. Softw.*, 38(1):7:1–7:4, December 2011.
7. A.K. Qin, V.L. Huang, and P.N. Suganthan. Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization. *IEEE Transactions on Evolutionary Computation*, 13(2):398–417, April 2009.
8. R. Storn and K. Price. Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11:341–359, 1997.
9. Lin-Yu Tseng and Chun Chen. Multiple trajectory search for large scale global optimization. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3052–3059, June 2008.