Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Mejora en el proceso de búsqueda en Evolución Diferencial

Estudiante: Iago Uhía Otero
Dirección: José Santos Reyes
Dirección: Javier Parapar López

A Coruña, junio de 2020.

A los que nunca abandonan

Agradecimientos

Me gustaría agradecer especialmente a los directores de este proyecto, José Santos Reyes y Javier Parapar López, por su inestimable ayuda para poder llevar a cabo este proyecto. Agradecer también el apoyo de mi familia y amigos durante todos estos años.

Resumen

En este proyecto se ha abordado una hibridación entre la búsqueda global de un método evolutivo (Evolución Diferencial) y un método de búsqueda local (*Line Search*), tratando de aunar las ventajas de cada método y proponiendo nuevos esquemas de explotación variable en cada gen (parámetro del genotipo) que codifique cada solución.

El objetivo del proyecto es identificar cuál de estas variantes es más adecuada para resolver un problema de optimización, dependiendo de las características de cada problema. También se busca identificar qué variante se adapta mejor a cualquier problema que se presente, independientemente de las características de este.

Para cumplir con los objetivos se ha realizado la implementación de dos algoritmos híbridos de ED-LS y se ha analizado su rendimiento junto al de sus progenitores en 13 escenarios (funciones *benchmark*) de distintas características.

Con el fin de realizar un desarrollo satisfactorio del proyecto, se ha decidido utilizar la metodología ágil Scrum, adaptándola a las peculiaridades de este proyecto de investigación. Esta metodología se ha regido por ciclos de desarrollo cortos, permitiendo de esta manera ajustar con facilidad la dirección de la investigación, así como recabar información para planificar de la mejor manera posible las siguientes tareas a realizar.

Palabras clave:

- Evolución Diferencial
- *Line Search*
- Método de optimización

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este trabajo se ha realizado una implementación de una solución híbrida entre la búsqueda global de Evolución Diferencial (ED) y la búsqueda local de *Line Search* (LS). Posteriormente se ha realizado una comparativa de calidades obtenidas, utilizando un mismo número de evaluaciones necesarias de calidad, entre las diferentes soluciones. Para ello se ha utilizado una variedad de funciones *benchmark* ampliamente utilizadas en computación evolutiva para la comparativa de algoritmos (unimodales, multimodales, separables, no separables, ...). Por último se ha realizado también un análisis de los resultados obtenidos teniendo en cuenta la naturaleza de las distintas funciones *benchmark*, así como la configuración de las distintas soluciones propuestas.

Los algoritmos de búsqueda local como *Line Search* (LS) son métodos que mantienen una solución actual, sobre la cual se realizan optimizaciones de forma iterativa realizando una búsqueda sobre su vecindad. El principal interés en este tipo de algoritmos proviene de su capacidad para encontrar soluciones óptimas de manera rápida y efectiva, utilizando para ello un número muy reducido de evaluaciones. El mayor problema de los algoritmos de búsqueda local es su carencia de mecanismos para alcanzar la solución óptima en espacios de búsqueda complejos.

Por otro lado, los algoritmos evolutivos como, por ejemplo, *Evolución Diferencial* (ED), utilizan una población de soluciones candidatas, la cual es evolucionada con el fin de explorar el espacio de búsqueda a través de cuatro pasos: selección de los padres, cruce, mutación y reemplazo. Los algoritmos evolutivos han demostrado ser muy eficientes en espacios de búsqueda complejos. Sin embargo, en soluciones que requieran un ajuste fino estos distan de ser tan eficientes al compararlos con los grados de eficiencia que pueden lograr los algoritmos de búsqueda local.

El principal objetivo de este proyecto es el de realizar una hibridación entre ambas estrategias favoreciendo la búsqueda de una solución óptima global, al mismo tiempo que se intenta acelerar la obtención de la misma. Es importante tener en cuenta que, en ningún caso este

proyecto pretende demostrar qué algoritmo es mejor que otro. Lo que se pretende es identificar los puntos fuertes y débiles de las distintas estrategias en los distintos escenarios, con el fin de que esta investigación pudiera ser un buen punto de partida en un futuro para crear un híbrido ED-LS que potencie los puntos fuertes y minimice los puntos débiles detectados en este proyecto. El trabajo realizado en este proyecto podría ser, por ejemplo, utilizada en la creación de un híbrido que se especialice en un tipo de problema concreto, o en un híbrido de propósito más general.

El trabajo de Whitley y col. [1] establece los esquemas de combinación clásicos entre búsqueda global poblacional y búsqueda local, distinguiendo entre *Lamarckian strategies* y *Baldwinian strategies*. En el primer caso significa que lo refinado por la búsqueda local en cada individuo de la población genética revierte en o modifica al genotipo inicial. En el segundo caso, la búsqueda local solo modifica el *fitness*, sin modificar el genotipo. El primer caso (*Lamarck*) generalmente implica una búsqueda más rápida pero con la posibilidad de un rápido decremento en la variabilidad de la población genética (y consecuente estancamiento en la búsqueda). Las estrategias de *Lamarck* también se usan generalmente en los "algoritmos meméticos", cuando la búsqueda genética global se combina con diferentes metaheurísticas [2].

Ejemplos de hibridación entre ED y algoritmos de búsqueda local se pueden encontrar en [3][4]. En los trabajos [5][6] se pueden encontrar hibridaciones entre PSO (Particle Swarm Optimization) y el algoritmo *Line Search*.

Para este trabajo se ha considerado la implementación de dos hibridaciones, ambas pertenecientes a la variante lamarckiana. La diferencia entre ambas hibridaciones radica en el grado de explotación aplicado en cada gen/parámetro del genotipo. Mientras que la primera versión (Sección 2.3) realiza el mismo nivel de explotación en cada gen/parámetro, la segunda versión (Sección 2.4) realiza un nivel de explotación dependiente de cuán bien esté establecido un gen/parámetro en la población.

Con el fin de lograr los objetivos de este proyecto se han diseñado 6 escenarios de configuración para los distintos algoritmos propuestos (Capítulo 2). Estas configuraciones se han utilizado en hasta 13 funciones *benchmark* (Apéndice A.1), las cuales pretenden cubrir escenarios de diversas características.

En el capítulo 2 de esta memoria se puede encontrar una descripción del funcionamiento de LS (Sección 2.2), de ED (Sección 2.1), así como una descripción de los dos algoritmos híbridos implementados para cumplir los objetivos de este proyecto (Secciones 2.3, 2.4). En la sección 2.5 se describen las principales características a tener en cuenta de las funciones *benchmark* utilizadas. También se incluye una tabla clasificatoria (Tabla 2.1) sobre la base de esas características. La configuración utilizada (Sección 3.1) y los resultados obtenidos son descritos en el capitulo 3. Por último, en el capítulo 5, se presentan las conclusiones y traba-

jo futuro del proyecto. En los apéndices se puede encontrar la definición de las 13 funciones *benchmark* (Apéndice A.1), así como una imagen gráfica de las mismas (Apéndice A.2).

Capítulo 2

Métodos

En este capítulo se puede encontrar una descripción del funcionamiento de LS, de ED, así como una descripción de los dos algoritmos híbridos implementados para cumplir los objetivos de este proyecto. Posteriormente se describen las principales características a tener en cuenta de las funciones *benchmark* utilizadas. Por último se incluye una tabla clasificatoria sobre la base de esas características.

2.1 Evolución Diferencial

Evolución Diferencial (ED) [7, 8, 9, 10, 11] es un algoritmo evolutivo. Su objetivo, por lo tanto, es la optimización y búsqueda de soluciones en problemas de distinta índole, utilizando como base las ideas de la evolución propuestas por Charles Darwin, así como los descubrimientos en genética realizados por Gregor Mendel.

Dada una población inicial aleatoria, ED, a través de un proceso iterativo, trata de mejorar la solución candidata con respecto a una medida de calidad (*fitness*). Las soluciones candidatas se obtienen mediante la combinación de soluciones anteriores, utilizando operaciones de cruce vectorial y mutación. Más concretamente, estas soluciones candidatas, se forman añadiendo a un vector (elegido aleatoriamente) de un miembro de la población (X_1), las diferencias ponderadas entre otros dos miembros $(X_2 - X_3)$, también elegidos aleatoriamente. Al resultado de esta operación lo llamaremos "vector donador".

$$X_{donador} = X_1 + F(X_2 - X_3) \tag{2.1}$$

El algoritmo básico de ED se presenta en el pseudocódigo de 2.1.1, mientras que en la figura 2.1 se muestran las operaciones sobre los vectores implicados.

Una de las principales ventajas de ED frente a otros algoritmos evolutivos clásicos es que reduce considerablemente el número de parámetros que necesitan ajustarse. Los parámetros a considerar son:

- ***CR*** controla la probabilidad de cruce
- ***F*** Pondera la diferencia entre los 2 miembros seleccionados para calcular el vector "donador"

El algoritmo ED, posee cuatro etapas bien definidas:

1. La ***inicialización***, que se ejecuta una única vez al comienzo del algoritmo. Su objetivo es darle valores iniciales a los parámetros de cada uno de los individuos de la población. Estos parámetros habitualmente suelen acotarse al rango del problema a resolver.

2. La ***mutación***, se encarga de generar los vectores donadores, expandiendo así el espacio de búsqueda.

3. La ***recombinación*** (crossover), se encarga de obtener los vectores de prueba $y_{p,m}$, mezclando los parámetros del vector inicial $v_{p,m}$ y el vector donador $x_{p,m}$, sobre la base de un número aleatorio que se compara con el factor CR. Esto se realiza de la siguiente manera:

$$y_{p,m} \begin{cases} x_{p,m} \text{ si } rand_{p,m(float(0,1))} < CR \text{ o si } m = rand_{p,m(integer(0,D-1))} \\ v_{p,m} \text{ en caso contrario} \end{cases}$$

   siendo **D**, **p** y **m**, el número de variables del vector, el índice del individuo dentro de la población y la variable del vector $p(m = 0...(D-1))$

4. La ***selección***, se encarga de formar la población objetivo, es decir, la nueva población de vectores iniciales para la siguiente generación $g+1$. Para ello se inserta en el vector inicial de la nueva generación v_p^{g+1} al vector v_p^g o al vector de prueba y_p^g, según el que produzca un mejor valor en la función de *fitness*.

$$v_p^{g+1} \begin{cases} y_p^g \text{ si } f_{obj}(y_p^g) \leq f_{obj}(v_p^g) \\ v_p^g \text{ en caso contrario} \end{cases}$$

Las últimas tres etapas se repiten generación tras generación en el mismo orden que han sido expuestas hasta satisfacer un criterio de parada establecido.

La idea fundamental del algoritmo es adaptar la magnitud de la diferencia de vectores $(F(x2–x3))$ a lo largo del proceso evolutivo. En las primeras generaciones la diferencia de vectores será grande, ya que probablemente los individuos inicialmente disten mucho unos de otros, pero a medida que avanza la evolución tenderá a producirse una convergencia de

la población, con lo que la diferencia se vuelve cada vez menor. Esto puede apreciarse en la figura 2.1. De este modo, el algoritmo presenta un balance automático entre exploración y explotación.

ED puede utilizarse con distintos esquemas que representan distintos variantes del algoritmo, centrados principalmente en especificar cómo se llevan a cabo las operaciones de mutación y de cruce. Para distinguir estas variantes, la notación "DE/α/β/γ" es comúnmente utilizada, dónde α indica cuál es el vector objetivo de la mutación (X_1), β es el número de pares de soluciones utilizadas para calcular los vectores diferencia (X_2 y X_3) y γ indica el esquema de cruce. Uno de los esquemas más populares es llamado "DE/rand/1/bin", donde "DE" significa Evolución diferencial, la palabra "rand" indica que el individuo base (X_1) para formar el vector donador es elegido de modo aleatorio de la población, "1" 1 es el número de vectores diferencia considerados para formar el vector donador (X_2 y X_3) y, por último, "bin" indica que el cruce se aplica a través de una recombinación binaria (recombinación binaria indicada en el pseudo-código y en el paso 3 de ED anteriormente descrito).

Algorithm 2.1.1: EVOLUCIÓN DIFERENCIAL($Poblacion$)

for each $Individuo \in Poblacion$
do $\{Individuo \leftarrow$ INICIALIZARPOSICIONESALEATORIAS$()$
repeat
for each $Individuo\ \ x \in Poblacion$
do
- $x_1, x_2, x_3 \leftarrow$ TOMARINDIVIDUOSALEATORIOS$(Poblacion)$
- // x_1, x_2, x_3 deben ser distintos entre ellos y de x
- $R \leftarrow$ TOMARALEATORIO$(1, D)$// D es la dimensión del problema
- **for each** $i \in 1 : D$
- // Calcula la nueva posición potencial del individuo $y = [y_1, ..., y_n]$ (vector de prueba)
- **do**
  - $r_i \leftarrow$ TOMARALEATORIO$(0, 1)$ // de manera uniforme en el intervalo (0,1)
  - **if** $((i = R)\ ||\ (r_i < CR))$ // CR - probabilidad de cruce
    - $y_i = x_{1_i} + F(x_{2_i} - x_{3_i})$ // F - factor de peso
  - **else** $y_i = x_i$
- **if** $(f(y) \leq f(x))$ $x = y$ // Si y tiene mejor *fitness*, se remplaza x con y

until CRITERIODEPARADA$()$
return (TOMARMENORFITNESS$(Poblacion)$) // devolver mejor solución candidata

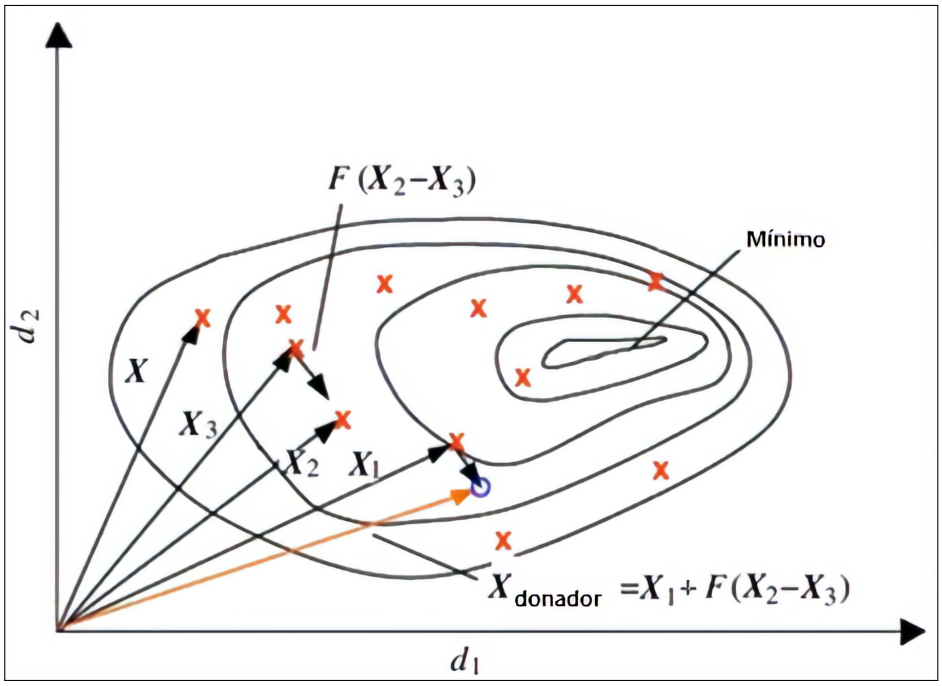


Figura 2.1: Generación de vector donador en Evolución Diferencial.

2.2 Line Search

Line Search (LS) [12, 13, 14] es un algoritmo de búsqueda u optimización local. Su idea básica es, a partir de una solución inicial en el espacio de búsqueda, muestrear en su vecindad y en cada dimensión de modo independiente para buscar una mejor solución.

En *Line Search* se parte desde un punto aleatorio en el espacio, se realiza una búsqueda en cada dimensión, moviendo cada vez el valor del parámetro en una dimensión mientras el resto de valores en otras dimensiones permanecen fijos. El procedimiento está resumido en el algoritmo 2.2.1. Por cada dimensión, se seleccionan N puntos equidistantes en cada eje (dimensión) alrededor del valor inicial (teniendo en cuenta los límites del parámetro en el espacio). Cada uno de los puntos es evaluado con la función de *fitness*, guardando el punto con el mejor *fitness*. La función de *fitness* dependerá del problema y mide la calidad de un individuo para una tarea dada. Este procedimiento se repite por cada una de las dimensiones o parámetros del problema (Paso 1 del algoritmo).

En un segundo paso del algoritmo, se define una línea entre el punto original y el nuevo punto. Este nuevo punto es el resultado del primer paso, en el cual cada valor en cada dimensión se define como el mejor valor encontrado en el muestreado en esa dimensión. Esta línea representa la dirección "prometedora". Una vez obtenida esta línea, el proceso se repite de nuevo seleccionando un número de puntos equidistantes entre los dos extremos. El punto con mejor *fitness* es seleccionado, y si este nuevo punto es mejor que el original, este pasa a ser el nuevo punto de origen en la siguiente iteración del algoritmo. Este proceso de muestreo puede apreciarse con mayor detalle en la figura 2.2.

De esta manera, una "iteración" se define como un ciclo a través de todos los parámetros, donde finalmente se realiza una búsqueda en la dirección prometedora. Por lo tanto, si hay P dimensiones, entonces se realizaran $P + 1$ operaciones de búsqueda *line search* por iteración. Además, la escala entre los puntos seleccionados se reduce mediante un factor de reducción al comienzo de cada nueva iteración, permitiendo una mayor exploración en las primeras iteraciones y una mayor explotación en las iteraciones finales.

Algorithm 2.2.1 Line Search.

N=Número de puntos de muestra en cada dimensión, D=Número de dimensiones, I=Intervalo entre puntos de muestra.
Seleccionar un punto aleatorio inicial.
Paso 1 del algoritmo (para la dimensión d)
$min \leftarrow\ max(0, posicion\_inicial[d] - \frac{I}{2}); (0\ es\ el\ limite\ inferior)$
$max \leftarrow\ min(1, posicion\_inicial[d] + \frac{I}{2}); (1\ es\ el\ limite\ superior)$
$incremento \leftarrow \frac{(max-min)}{N}$;
$mejor\_posicion[d] \leftarrow\ min; (Mejor\ posicion\ inicial)$
for $n \leftarrow 1$ ***to*** N **do**
$p \leftarrow min + incremento * n$;

$nueva\_posicion[d] \leftarrow\ p$;
if $(fitness(nueva\_posicion[d]) < fitness(mejor\_posicion[d]))$ **then**
$mejor\_posicion[d] \leftarrow\ nueva\_posicion[d]$;
end if
end for
return mejor_posicion[d]

Paso 2 del algoritmo
for $d \leftarrow 1$ **to** D **do**
$max\_dim[d] \leftarrow\ max(posicion\_inicial[d], mejor\_posicion[d])$;

$min\_dim[d] \leftarrow\ min(posicion\_inicial[d], mejor\_posicion[d])$;

$incremento[d] \leftarrow\ \frac{(max\_dim[d]-min\_dim[d])}{N}$;
end for
$mejor\_posicion \leftarrow\ posicion\_inicial$

for $n \leftarrow 1$ ***to*** N **do**
for $d \leftarrow 1$ ***to*** D **do**
$nueva\_posicion[d] \leftarrow\ min\_dim[d] + incremento[d] * n$;

end for
if $(fitness(nueva\_posicion) < fitness(mejor\_posicion))$ **then**
$mejor\_posicion \leftarrow nueva\_posicion$

end if
end for
return $mejor\_posicion$

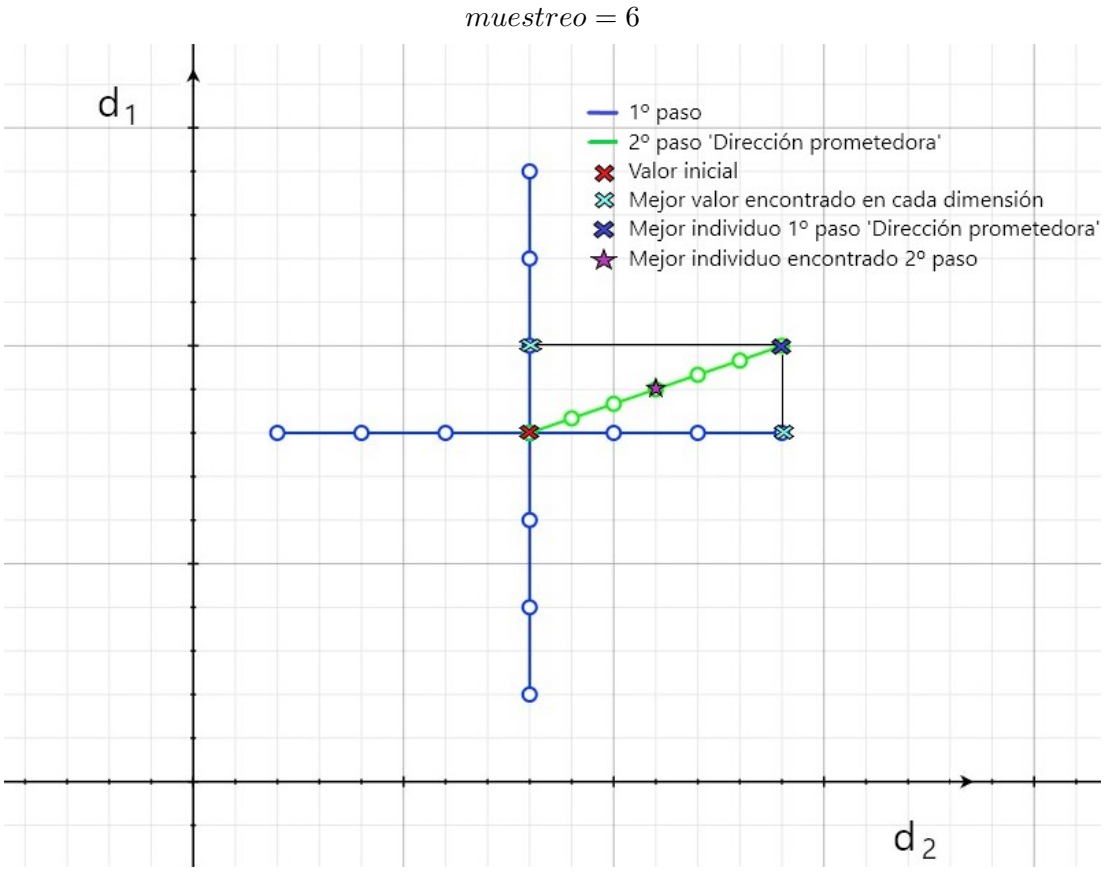


Figura 2.2: Proceso de muestreo del espacio de búsqueda en *Line search* con el fin de obtener un punto con una mejor calidad.

2.3 Algoritmo híbrido ED - Line Search

Con el objetivo de mejorar el proceso de búsqueda de ED, en este trabajo se ha realizado la implementación de un algoritmo híbrido que se aproveche tanto de las bondades de ED como de LS (de ahora en adelante se denominará como Híbrido).

LS es un algoritmo sencillo y rápido a la hora de moverse por el espacio de búsqueda para realizar explotación, pero a costa de una pobre exploración en fases más avanzadas, lo que a menudo provoca que este se quede atascado en mínimos locales (máximos si el problema es de maximización). En cambio, ED, como método de búsqueda global, posee mayor capacidad de exploración, lo que hace difícil que este se estanque en mínimos locales, a costa del mayor cómputo necesario de realizar la búsqueda simultánea en las soluciones de la población genética. Debido a esto, dependiendo del tipo de problema al que nos enfrentemos, elegir entre ED o LS puede ser muy determinante, ya sea a la hora de requerir una solución mejor o mayor velocidad en la ejecución. Por resumir, podríamos decir que, en general, sin estudiar la naturaleza del problema, ED actuará lento pero seguro y LS rápido pero de manera poco fiable.

Este Híbrido pretende mejorar el balance entre explotación y exploración, posibilitando la existencia de un algoritmo que obtenga buenos resultados independientemente de la naturaleza del problema, e incluso llegando a superar a sus progenitores (ED y LS) en algunos problemas cuya naturaleza premie este balance.

La idea detrás de este algoritmo es sencilla, consiste en ejecutar ED pero añadiendo una 5º etapa al proceso. Después de que ED realice la selección de individuos para la siguiente generación, se toma al individuo con mejor *fitness* de la población como semilla para LS, y posteriormente se aplica un paso de LS. Finalmente, el individuo o solución resultante reemplazará al mejor individuo, para que ese mejor individuo refinado por LS forme parte de la nueva generación.

Por tanto, es una hibridación típica lamarckiana entre una búsqueda poblacional y una búsqueda local, en el sentido de que los individuos refinados por la búsqueda local reemplazan a los individuos de la población [1]. La diferencia es que, en este caso, restringimos la hibridación únicamente al mejor individuo de la población, lógicamente con el fin de no degradar en gran medida el tiempo requerido en la búsqueda, con respecto a la hibridación en la que se aplica la búsqueda local a cada individuo, además de aplicar solo una iteración de LS en el mejor individuo.

2.4 Algoritmo híbrido ED - Line Search modificado

En esta segunda revisión de hibridación se va a realizar una modificación, considerando un grado de explotación diferente (en el mejor individuo) en cada gen/parámetro del genotipo. La idea es que ese grado de explotación depende de cuán bien esté establecido un gen/parámetro en la población.

Para ello se chequeará el nivel de variabilidad de un gen en la población. Esto se hará de modo muy sencillo, considerando el valor absoluto de la diferencia $X_2 - X_3$ usada en la generación del individuo donador (sección 2.1). La variabilidad en los genes se discrimina en cuartiles, y en cada cuartil (en los genes de cada cuartil) se utilizará diferente cantidad de explotación con *Local Search*.

El objetivo de esta operación es el de poder refinar las soluciones de un modo más efectivo, realizando menos evaluaciones de *fitness* en aquellos genes (parámetros del problema) que pueden estar mejor determinados en la población, mientras se potencia la búsqueda en aquellos genes con más variabilidad.

El nivel de explotación en cada gen, dependiendo del cuartil asignado, vendrá dado por un número de puntos o muestras equidistantes a evaluar (1º paso de LS, sección 2.2). Para ello, ese número de puntos a evaluar se establece de la siguiente manera, con el siguiente orden de prioridad (de mayor a menor variabilidad de los genes):

- Q_4 ($\geq 75\%$ *de variabilidad)* : $n\_muestras\_base + (n\_muestras\_base/2)$
- Q_3 ($\geq 50\%$ *de variabilidad)* : $n\_muestras\_base$
- Q_2 ($\geq 25\%$ *de variabilidad)* : $n\_muestras\_base - (n\_muestras\_base/2)$
- Q_1 ($\geq 0\%$ *de variabilidad)* : 0 (no se realiza explotación)

donde $n\_muestras\_base \geq 2$ y par, siendo $n\_muestras\_base$, un parámetro de configuración inicial que determina un número de muestras promedio en el proceso de LS (en $Q_2 - Q_4$).

Por tanto, los genes con mayor variabilidad estarán asignados a Q_4 y los de menor variabilidad a Q_1.

Se priorizará que cada uno de los 4 niveles de explotación tenga el mismo número de genes asignados. En caso de que esto no sea posible, debido a que № parámetros mod $4 \neq 0$, estos genes restantes serán asignados a Q_1, o lo que es lo mismo, no se realizará explotación en estos genes.

Si cada uno de los 4 niveles de explotación tiene el mismo número de genes, con el proceso descrito anteriormente, el número de evaluaciones de *fitness* en este 1º paso de LS se reduce en un 25% con respecto al número de evaluaciones que se realizarían sin este proceso. Este

porcentaje se deduce de la distribución de los genes en los diferentes cuartiles ya que, si por ejemplo tomamos 4 genes ($dim = 4$) con $n\_muestras\_base = 2$ cada gen respectivo realizaría 3, 2, 1 y 0 evaluaciones, correspondiendo a los cuartiles Q_4 Q_3 Q_2 y Q_1, pero sin aplicar este proceso se realizarían $n\_muestras\_base * gen = 8$ evaluaciones.

Al aplicar este proceso también se estaría potenciando la explotación en el espacio cercano a los genes peor determinados en la población actual (genes con mayor variabilidad).

2.5 Funciones de benchmark utilizadas

Dependiendo de las características de una función a optimizar, el rendimiento de los algoritmos puede variar de forma muy notable. Por ejemplo, un algoritmo de búsqueda local como LS, ante problemas con mucho ruido tenderá a estancarse en puntos "valle".

Con el objetivo de mostrar en este proyecto el comportamiento de los algoritmos en distintos escenarios, se han seleccionado para ello 13 funciones con distintas características (tabla 2.1). Entre los rasgos característicos de una función particularmente importantes en optimización, nos encontramos con la modalidad y la separabilidad.

- **Modalidad:** se determina en función del número de mínimos (locales o global) que presenta la función.
  - *Unimodal: se refiere a funciones que solo poseen un mínimo (global).*
  - *Multimodal: se refiere a funciones que poseen dos o más mínimos locales..*

  En una función multimodal, un algoritmo como LS puede quedarse estancado durante el proceso de búsqueda, fallando a la hora de encontrar el mínimo global. Por lo tanto, el proceso de búsqueda se ralentiza, dificultando el encontrar verdaderas soluciones óptimas.

- **Separabilidad:** La separabilidad está relacionada con la interrelación entre las variables de una función. La función de calidad en una función separable indica que el valor de calidad depende de la calidad que defina cada variable por separado. Es decir, no hay interdependencia entre variables a la hora de determinar la calidad, mientras que en una función no-separable lo contrario, la calidad depende de la combinación de valores entre las variables. Es decir, en este sentido, existe una interdependencia entre las variables a la hora de determinar la calidad. Por lo tanto, una función con n variables se dice que es separable, si puede ser expresada como una suma de n funciones cada una con una única variable.

  $f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} f(x_i)$

Las funciones no-separables presentan lógicamente mayores dificultades a la hora de ser optimizadas.

	Modalidad	
Separable	*Unimodal*	*Multimodal*
SÍ	• f01 - Sphere Model • f02 - Schwefel's Problem 2.22 • f04 - Schwefel's Problem 2.21 • f06 - Step Function • f07 - Quartic Function with Noise	• f08 - Generalized Schwefel's Problem 2.26 • f09 - Generalized Rastrigin's Function
NO	• f03 - Schwefel's Problem 1.2	• f05 - Generalized Rosenbrock's Function • f10 - Ackley's Function • f11 - Generalized Griewank's Function • f12 - Generalized Penalized Function • f13 - Generalized Penalized Function

Tabla 2.1: Características principales de las 13 funciones *benchmark* utilizadas en este proyecto.

2.6 Detalles de implementación

Para la implementación de los cuatro algoritmos descritos en este capítulo (2.1, 2.2, 2.3 y 2.4) se ha utilizado el lenguaje de programación *Java*. También cabe remarcar que cada uno de estos algoritmos ha sido paralelizado con hilos.

Capítulo 3

Experimentos y resultados

Las pruebas realizadas en este proyecto se han diseñado con el fin de poner a prueba el comportamiento de los algoritmos previamente descritos en el capítulo 2, en condiciones similares, en un conjunto de problemas de alta dimensionalidad y diferentes características. Los 13 problemas seleccionados se presentan en la tabla 2.1, también se puede encontrar su definición, así como su correspondiente gráfica en el anexo A.1.

En este capítulo se encuentran las gráficas comparativas del comportamiento de los distintos algoritmos, así como las tablas mostrando los resultados finales. En estas tablas y gráficas se indicarán los valores de los parámetros de configuración establecidos en cada experimento ($muestras$ y $dimensiones$), y se deberá entender que los parámetros no indicados se mantienen sin cambios en todos los experimentos.

3.1 Configuración

En esta sección se describen los parámetros de configuración utilizados, así como el valor utilizado en los experimentos realizados.

3.1.1 Parámetros de configuración comunes a todos los algoritmos

- **Dimensiones del problema**: Se han realizado experimentos con **30 y 100 dimensiones** en cada uno de los 13 problemas. Estas dimensiones son estándar en computación evolutiva al comparar diferentes algoritmos con funciones de *benchmark*.

- **№ de ejecuciones**: Se presentarán los datos sobre la base de la media de **1000 ejecuciones**. Ello es necesario dada la estocasticidad de los algoritmos. Este es un número muy alto (1000) comparado con el número estándar (50) usado en computación evolutiva para ejecuciones independientes.

- **№ de evaluaciones de calidad**: El número máximo de evaluaciones de calidad en cada ejecución se ha establecido sobre la base de la siguiente formula:

$$evaluaciones = dimensiones * 10000 \tag{3.1}$$

  Esta formula se ha establecido de esta manera por ser comúnmente utilizada por diversos autores cuando comparan algoritmos con funciones de *benchmark* de diferente dimensionalidad.

  Algunos de los algoritmos sobrepasan lo mínimo indispensable este valor. Esto se debe a que cada uno de los algoritmos tiene un número de evaluaciones distinto en cada iteración del mismo, lo cual imposibilita que dispongan del mismo número exacto de evaluaciones objetivo. En todo caso, las gráficas principales que se mostrarán, harán hincapié en el comportamiento de los algoritmos frente al mismo número de evaluaciones de calidad.

- **Dominio del problema**: Los genotipos codifican las soluciones en el rango **-1 y 1**. De esta manera, los distintos algoritmos utilizan en sus operadores genéticos esas soluciones normalizadas, consiguiendo ser independientes de la función de *benchmark* utilizada. Únicamente, en el momento de calcular el *fitness*, estas soluciones se decodifican al rango original de la función correspondiente (anexo A.1).

  El objetivo de esta configuración es la de poder mostrar y comparar el comportamiento de los distintos algoritmos, independientemente del dominio del problema.

3.1.2 Parámetros de configuración de Evolución Diferencial

- **Esquema de Evolución diferencial**: Se ha establecido **rand/1/bin** como estrategia de ED (Sección 2.1). Se ha seleccionado por ser el esquema más común y el que proporciona menor presión selectiva.
- **Población**: Se ha establecido una población de **100 individuos**.
- **F**: Se ha establecido un valor de **0.6**.
- **CR**: Se ha establecido un valor de **0.9**.

Se puede encontrar una descripción de F y CR en la sección 2.1.

Los valores de F y CR, con valores en rangos estándar, se han establecido de modo experimental, siendo valores que proporcionan un adecuado balance entre exploración y explotación en las diferentes funciones de *benchmark*. El tamaño de la población es un valor común en otros trabajos previos con DE y las funciones de *benchmark*.

3.1.3 Parámetros de configuración de *Line search*

- **Muestras**: Se han realizado experimentos con **2, 6 y 12 muestras** , es decir, el número de puntos equidistantes que se utilizan en cada dimensión en el proceso de búsqueda con LS. El objetivo es el de poder mostrar cómo se comporta el algoritmo cuando este valor se reduce o aumenta en los distintos escenarios.

- **Tamaño inicial cubierto por el muestreo con LS**: Se ha establecido en la cuarta parte del rango en cada dimensión. Es decir, para cada punto inicial considerado en LS, el muestreo inicial considerará ese rango para definir los puntos extremos a chequear en cada dimensión (independientemente del número de muestras).

- **Factor de reducción del tamaño cubierto por el muestreo con LS**: Este factor reduce el tamaño del intervalo entre muestras. Para ello, después de cada iteración del algoritmo, se establece el nuevo tamaño del intervalo multiplicando el antiguo por este factor. Se ha establecido este factor en **0.9**.

3.1.4 Parámetros de configuración del algoritmo híbrido

- **Muestras**: El número de muestras establecido en LS se ha variado únicamente en el caso del híbrido modificado, y se ha establecido conforme a lo ya explicado en la sección 2.4.

- **Factor de reducción del tamaño cubierto por el muestreo con LS**: Los algoritmos híbridos, hasta alcanzar el número de evaluaciones de calidad establecidas, realizan un menor número de pasos/iteraciones de LS en comparación con los realizados por el algoritmo LS (estos dedican parte de sus evaluaciones a la estrategia evolutiva de ED). Por lo tanto, este factor de reducción se aplicará menos veces que en el algoritmo LS, provocando que el tamaño cubierto por el muestreo en la última evaluación con LS en las versiones híbridas, difiera en gran medida con el de la versión pura de LS. Con el objetivo de establecer una comparación más justa, este factor se ajusta para las versiones híbridas, en cada uno de los escenarios planteados, de manera que el tamaño cubierto por el muestreo en la última evaluación coincida con el alcanzado en la versión pura de LS.

3.2 F01 - Sphere Model

La figura 3.1 muestra la evolución de los distintos algoritmos. En esta figura se muestran 6 escenarios, cada uno de estos escenarios se determina sobre la base de un número de muestras y dimensiones establecido. A la izquierda se pueden ver los escenarios con 30 dimensiones mientras que a la derecha se encuentran los escenarios con 100 dimensiones. La primera fila se corresponde con los escenarios con 2 muestras en LS, la segunda con 6 y la tercera con 12. Las líneas continuas muestran la evolución de la calidad del mejor individuo de la población (promediadas además por las 1000 ejecuciones independientes), mientras que las líneas discontinuas muestran la media de la calidad de la población (promediando adicionalmente esa calidad media de las 1000 ejecuciones del correspondiente algoritmo). Las tablas, desde 3.1 hasta 3.6, muestran la media del mejor *fitness* obtenido en las 1000 ejecuciones independientes del correspondiente algoritmo. Tanto en las gráficas como en las tablas el Hibrido1 se corresponde con el Algoritmo híbrido DE - Line Search (2.3), mientras que Hibrido2 se corresponde con el Algoritmo híbrido DE - Line Search modificado (2.4).

Cabe remarcar que se ha decidido utilizar en las gráficas la evolución de calidad frente al número de evaluaciones en lugar de frente al número de generaciones. Debido a que los híbridos, por su definición, realizan un número de evaluaciones superior a ED por generación, al comparar la evolución de calidad frente al número de generaciones resultaría una comparación injusta para ED en favor de los híbridos. Por ejemplo, en la generación 100, ED ha realizado 10000 evaluaciones ($población = 100 * generaciones = 100$) mientras que el Hibrido1 ha realizado 16039.

Las evaluaciones del Hibrido1 se obtienen sumando las evaluaciones que este dedica a ED (Ecuación 3.2) y las evaluaciones que este dedica a LS (Ecuación 3.3). El híbrido dedica a ED una evaluación por cada miembro de la población en cada generación, mientras que a LS dedica por generación (exceptuando la 1ª) para su 1ª fase "$muestras * dimensiones$" y en su 2ª fase "$muestras - 1$" número de evaluaciones. En la primera generación, el híbrido solo realiza evaluaciones de ED (fase de inicialización). Por este motivo, en la ecuación 3.3 el número de evaluaciones dedicadas a LS por generación se multiplica por "$generaciones - 1$".

$$evaluaciones\_ED = población * generaciones \tag{3.2}$$

$$evaluaciones\_LS = (muestras*dimensiones+muestras-1)*(generaciones-1) \tag{3.3}$$

$$evaluaciones\_totales\_Hibrido = evaluaciones\_ED + evaluaciones\_LS \tag{3.4}$$

En la figura 3.2 se puede apreciar una comparativa del mismo escenario de la evolución de calidad frente al número de evaluaciones vs. frente al número de generaciones. En ella se puede observar cómo la media de ED pasa de estar siempre por encima de las medias de los híbridos (calidad vs. generaciones) a estar prácticamente siempre por debajo (calidad vs. evaluaciones). Por tanto, en la gráfica de calidad vs. generaciones, puede parecer que los híbridos son más eficientes que ED cuando en realidad no es el caso.

En general, todos los algoritmos se desenvuelven bien en todos los escenarios planteados ante este problema, como se denota en los resultados mostrados en la tablas. Todos los algoritmos evolucionan de forma muy rápida y consiguen acercarse mucho a la solución óptima. Esto no sorprende debido a la relativa facilidad de F01.

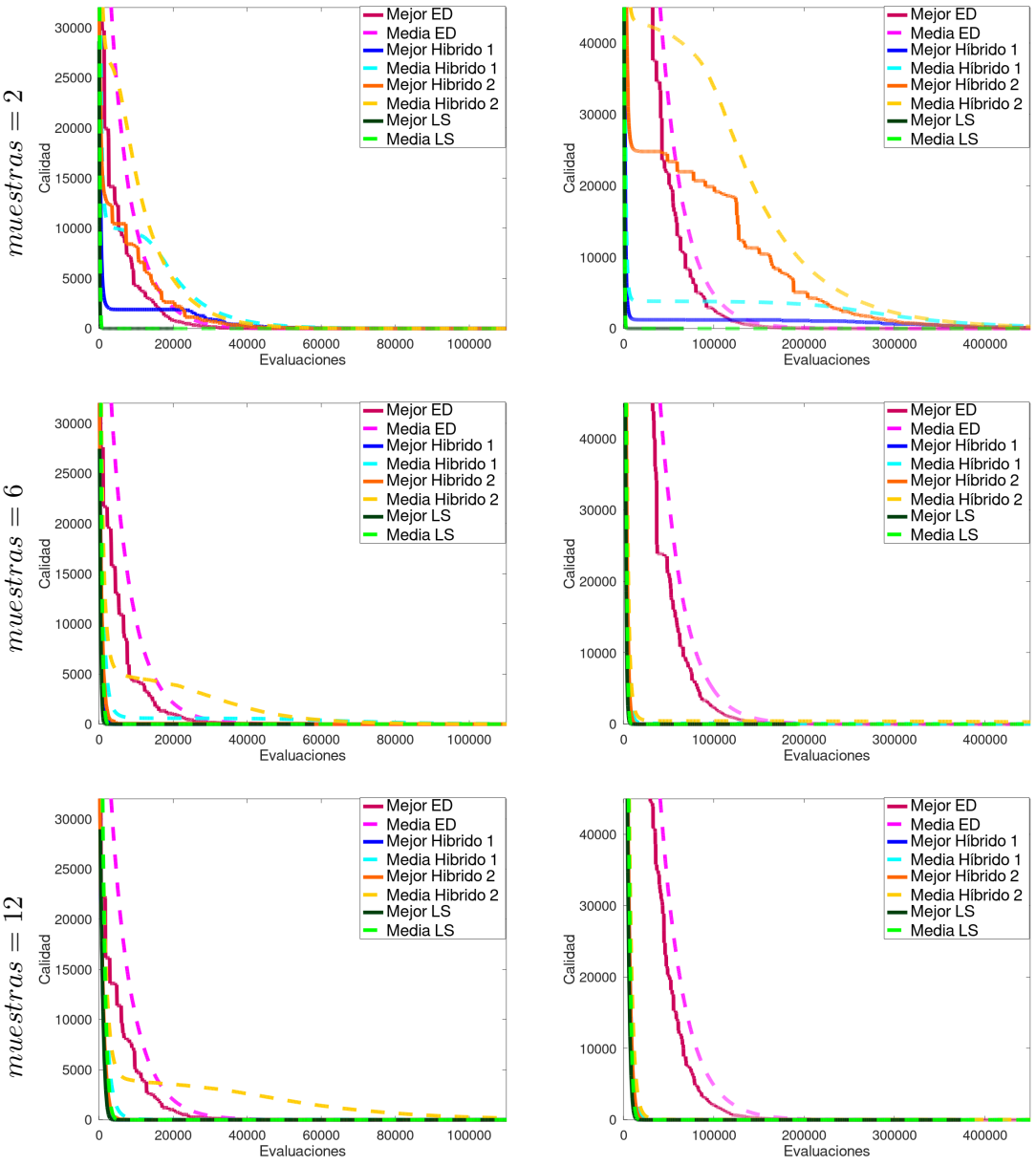
Destaca sobre los demás algoritmos LS, como era de esperar en este tipo de problema (unimodal y separable). LS es claramente el algoritmo que más rápido evoluciona. Es el ganador en todos los escenarios planteados exceptuando en la tabla 3.5 (seguramente porque alguna ejecución de LS cae en un mínimo local), cuyo vencedor es ED, aunque LS no se queda lejos de la solución. Además, LS consigue llegar siempre a la solución óptima en 3 de los 6 escenarios planteados.

Aunque las características de F01 favorecen, en gran medida, a un método de búsqueda local como LS, se puede observar cómo ED obtiene un 2º lugar mostrando también unos muy buenos resultados. Esto puede ser explicado por la sencillez del problema, lo que provoca que la población de ED converja muy rápidamente hacia la solución.

En cuanto a los métodos híbridos, sus resultados son claramente peores a ED y LS. Esto puede deberse a que las evaluaciones realizadas basadas en LS para refinar al mejor individuo de la población, no están siendo bien aprovechadas. Esto a primera vista puede parecer contradictorio, puesto que el método LS es el que obtiene los mejores resultados, pero para explicar esto se deben tener en cuenta 3 factores: se está utilizando como estrategia de ED rand/1/bin, los métodos híbridos solo refinan con LS al mejor individuo de la población y ED converge por sí mismo muy rápidamente hacia la solución en este problema. Con estos factores en mente, el peor funcionamiento de la hibridación puede explicarse porque la mejora

que introduce LS en el mejor individuo de la población no se usa (intensivamente) por parte de ED. Recordemos que ED elige el individuo base de modo aleatorio (esquema rand/1/bin), por lo que no es frecuente que el individuo mejorado por LS se use como vector base al definir el vector donador y al vector candidato. Además, LS implica un "gasto" de evaluaciones de calidad. Si ED encuentra fácilmente la solución, ese gasto extra lógicamente perjudica el algoritmo híbrido. En las tablas de resultados finales (con prácticamente el mismo número máximo de evaluaciones de calidad) se puede observar cómo efectivamente los valores tienden a degradarse a medida que se aumente el número de muestras consideradas en LS, ya que implica un número mayor de evaluaciones extra de calidad en cada generación del algoritmo evolutivo híbrido con respecto a ED sin hibridizar.

En las tablas podemos apreciar cómo, a diferencia de LS, los híbridos sí parecen sensibles a aumentar el nº de muestras. En general se aprecia que, a mayor número de muestras, la media empeora y la desviación típica aumenta, lo que podría indicar que se están desperdiciando evaluaciones al refinar individuos con LS. Este problema planteado es menos determinante en las primeras generaciones del proceso evolutivo, debido a que los primeros individuos refinados por LS tienen un enorme potencial a la hora de reducir el coste de la función de *fitness*, puesto que en F01, LS mueve rápidamente cualquier solución hacia la zona del mínimo global. Esto se puede apreciar en las gráficas viendo cómo los híbridos obtienen mejores resultados que ED en evaluaciones muy tempranas.



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.1: Gráfica de resultados: Función F01 - Sphere Model. Todas las gráficas son el promedio de 1000 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	8.336483370252633E-16	5.539263740695232E-15
LS	300060	0.0	0.0
Híbrido 1	300043	2.160585579018902E-9	1.9533067027311075E-9
Híbrido 2	300114	7.58179277210372E-11	5.644373267925272E-11

Tabla 3.1: Resultados F01 - 30 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	1.1736541129212682E-15	1.0310120361086305E-14
LS	300071	7.586920756021612E-18	1.8131332913868796E-16
Híbrido 1	300205	3.117141364778601E-4	1.8983410180700271E-4
Híbrido 2	300169	7.709698389061499E-6	5.411816123848402E-6

Tabla 3.2: Resultados F01 - 30 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	1.9974554461491455E-15	1.832423775057373E-14
LS	300140	0.0	0.0
Híbrido 1	300127	0.09815351182516999	0.15558361495816672
Híbrido 2	300301	0.020565200852508375	0.01057859197653748

Tabla 3.3: Resultados F01 - 30 dimensiones, $muestras = 12$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	1.7245159287138853E-11	1.279008191248143E-10
LS	1000177	0.0	0.0
Híbrido 1	1000022	0.6499681069003957	0.38059134316637566
Híbrido 2	1000084	0.13560414792340758	0.12024624586679589

Tabla 3.4: Resultados F01 - 100 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	7.604797529884352E-12	4.4190900128023326E-11
LS	1000066	0.004010021375490435	0.07720721423738473
Híbrido 1	1000495	8.530456727143735	22.75722445819124
Híbrido 2	1000210	21.46648874903995	13.894317819989515

Tabla 3.5: Resultados F01 - 100 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	1.3469919806867926E-11	1.7627139261292533E-10
LS	1000287	5.452606576887637E-29	1.2186630125312042E-27
Híbrido 1	1000393	0.005579548622580055	0.14395421810049092
Híbrido 2	1000990	18.515849532982948	57.24256766068135

Tabla 3.6: Resultados F01 - 100 dimensiones, $muestras = 12$ en LS.

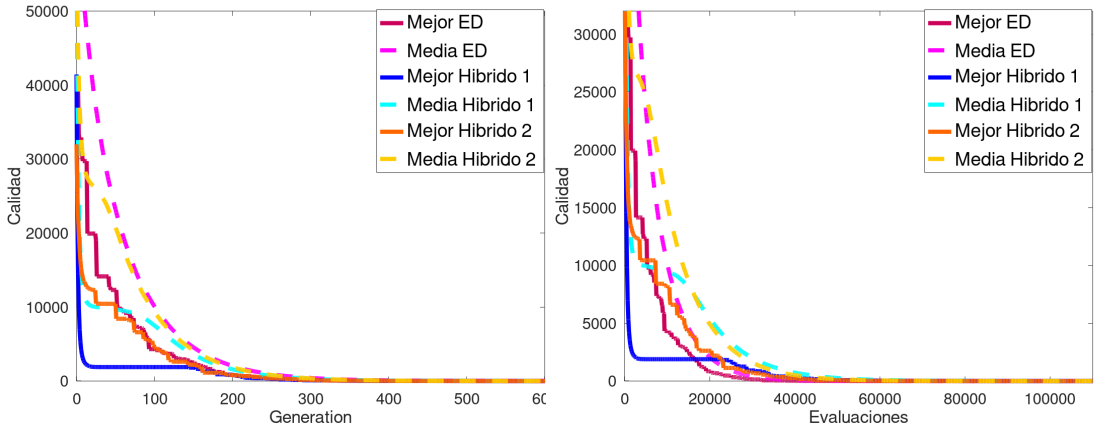


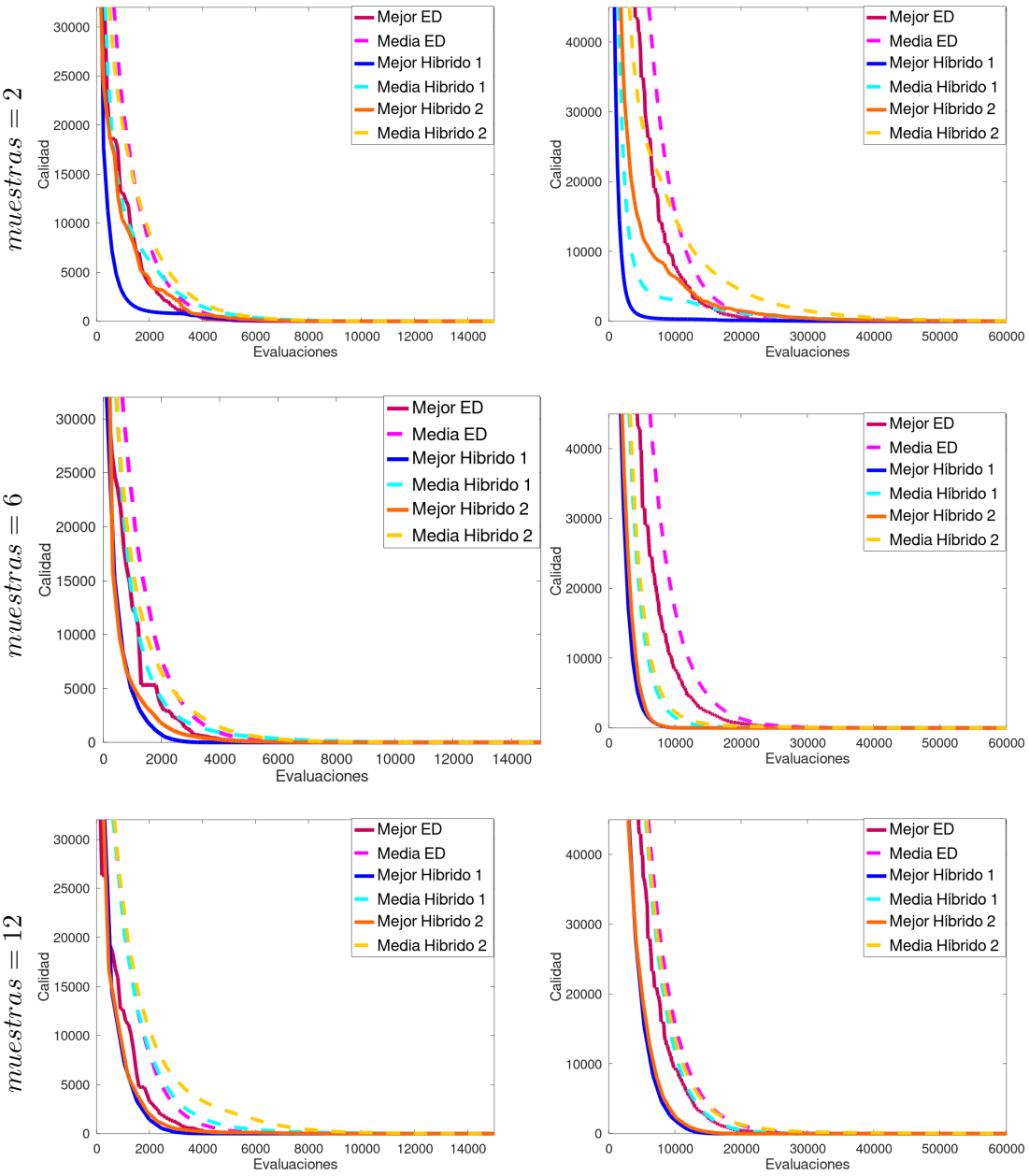
Figura 3.2: Función F01 - Sphere Model, 30 dimensiones con muestras = 2. A la izquierda calidad frente a generaciones. A la derecha calidad frente a evaluaciones.

En cuanto a los tiempos de ejecución de los distintos algoritmos, LS es el algoritmo más rápido, mientras que ED es el más lento, siempre considerando el mismo número de evaluaciones de calidad realizadas. Los algoritmos híbridos se encuentran entre LS y ED. La razón por la cual el Híbrido 2 es más lento que el Híbrido 1 es porque este dedica un mayor número de evaluaciones a ED (algoritmo más lento). En la tabla 3.7 se pueden encontrar los tiempos de ejecución en milisegundos de 1000 ejecuciones independientes para cada uno de los algoritmos.

Método	Tiempo de ejecución
ED	5126292ms
LS	2499474ms
Híbrido 1	2814095ms
Híbrido 2	4034529ms

Tabla 3.7: Tiempos de ejecución de F01 - 1000 ejecuciones, 100 dimensiones, $muestras = 6$ en LS.

Con el fin de mostrar el impacto de cambiar el esquema de ED se han recreado los escenarios previamente mostrados, pero esta vez utilizando el esquema de ED "DE/best/1/bin" en lugar de "DE/rand/1/bin" y utilizando el promedio de 100 ejecuciones independientes en lugar de 1000 (figura 3.3 y tablas de 3.8 a 3.13). Los resultados tanto de los híbridos como de ED mejoran notablemente. LS dejaría de ser el algoritmo mejor posicionado.



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.3: Gráfica de resultados: Función F01 - Sphere Model. Esquema de ED utilizado **DE/-best/1/bin**. Todas las gráficas son el promedio de 100 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	0.0	0.0
Híbrido 1	300043	1.8175355256292128E-29	1.8175355256292103E-28
Híbrido 2	300114	0.0	0.0

Tabla 3.8: Resultados F01 - 30 dimensiones, $muestras = 2$ en LS, esquema **DE/best/1/bin**.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	8.077935669463157E-30	8.077935669463158E-29
Híbrido 1	300205	2.827277484312105E-29	1.4067738959979192E-28
Híbrido 2	300169	0.0	0.0

Tabla 3.9: Resultados F01 - 30 dimensiones, $muestras = 6$ en LS, esquema **DE/best/1/bin**.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	0.0	0.0
Híbrido 1	300127	1.7569510081082386E-28	6.874849433236182E-28
Híbrido 2	300301	1.6155871338926317E-29	1.1366083143858393E-28

Tabla 3.10: Resultados F01 - 30 dimensiones, $muestras = 12$ en LS, esquema **DE/best/1/bin**.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.0	0.0
Híbrido 1	1000022	1.6438599087357534E-27	3.4585295813703314E-27
Híbrido 2	1000084	8.582806648804608E-28	3.562772010227954E-27

Tabla 3.11: Resultados F01 - 100 dimensiones, $muestras = 2$ en LS, esquema **DE/best/1/bin**.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.0	0.0
Híbrido 1	1000495	2.1734894176876504E-14	7.995368200021657E-14
Híbrido 2	1000210	4.1688491553630835E-18	1.74191418330403E-17

Tabla 3.12: Resultados F01 - 100 dimensiones, $muestras = 6$ en LS, esquema **DE/best/1/bin**.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.0	0.0
Híbrido 1	1000393	3.490808162682601E-9	3.292720088199208E-8
Híbrido 2	1000990	4.6345813218574935E-10	1.3777650499338402E-9

Tabla 3.13: Resultados F01 - 100 dimensiones, $muestras = 12$ en LS, esquema **DE/best/1/bin**.

3.3 F02 - Schwefel's Problem 2.22

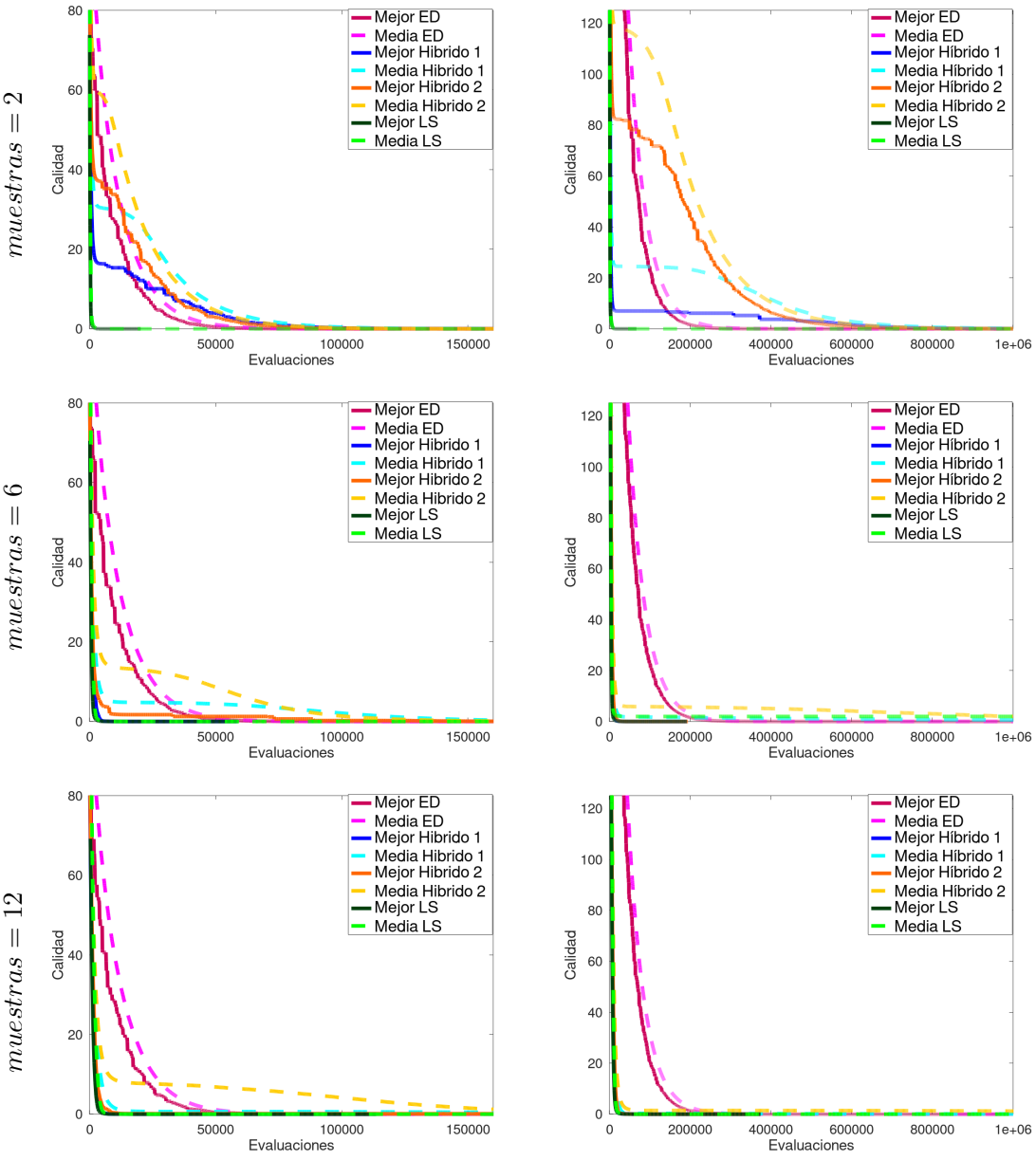
Este problema es similar a F01 en cuanto a sencillez. La principal diferencia de F02 con respecto a F01 es que F02 posee un valle mucho más amplio en el lugar donde se encuentra la solución óptima. Esto se puede observar si comparamos las gráficas de F01 y F02 (Apéndice A.2), al observar la parte baja veremos cómo es mucho más plana la de F02 que la de F01.

LS es claramente el algoritmo que más rápido evoluciona debido a que F02 premia mucho la explotación, al igual que en F01, pero la cosa cambia en fases más avanzadas, cuando los algoritmos se sitúan en el valle de la función. Este equilibrio es lo que explica por qué ED acaba obteniendo unos resultados similares a LS incluso llegando a quedar por encima en alguno de los casos planteados.

El número de $muestras = 6$ parece estar teniendo un impacto negativo en los resultados finales de LS. Probablemente esto se deba a que, aunque el valor inicial del cual parte el primer punto de muestreo es aleatorio, el número de muestras que se realicen aumentará o disminuirá la probabilidad de que el punto seleccionado se sitúe en una parte del valle más o menos cercana a la solución de F02.

Es de reseñar que, aunque LS obtiene mejores valores en varios casos respecto a ED, este último algoritmo de búsqueda global es más robusto, obteniendo muy buenos resultados en todos los casos. Sin embargo, LS presenta altas desviaciones estándar en varios casos con diferente número de muestras, lo que indica que LS se queda en un mínimo local en alguna de las ejecuciones independientes, a pesar de la naturaleza unimodal de la función.

Las versiones híbridas obtienen también buenos resultados pero continúan por detrás de sus progenitores, salvo algún caso en donde consiguen ganar a LS (Tabla 3.18). Por tanto, los híbridos parecen continuar teniendo los mismos problemas descritos en F01 (Sección 3.2)



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.4: Gráfica de resultados: Función F02 - Schwefel's Problem 2.22. Todas las gráficas son el promedio de 1000 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	7.175113534429978E-10	1.8222245763538227E-9
LS	300060	7.105427357600996E-18	1.7758233187877215E-16
Híbrido 1	300043	8.937627374010054E-6	3.095266641594542E-6
Híbrido 2	300114	1.3408034559869711E-6	8.339872822006885E-7

Tabla 3.14: Resultados F02 - 30 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	2.0934287192630992E-10	4.0567106628904647E-10
LS	300071	7.683895132606014E-4	0.012618283118294502
Híbrido 1	300205	0.00324388742710593	0.001099038252646976
Híbrido 2	300169	3.4400241693455236E-4	1.0937431104525477E-4

Tabla 3.15: Resultados F02 - 30 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	1.1381344471317149E-9	4.328389007996368E-9
LS	300140	2.8421709430403983E-17	8.987733679556482E-16
Híbrido 1	300127	0.07261881340306753	0.07202884201478653
Híbrido 2	300301	0.03400654767830376	0.008063878990746245

Tabla 3.16: Resultados F02 - 30 dimensiones, $muestras = 12$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	3.204571972226213E-8	1.6744736397332005E-7
LS	1000177	0.0	0.0
Híbrido 1	1000022	0.16292429015726473	0.07724178360661563
Híbrido 2	1000084	0.06017852013649675	0.02204961640636153

Tabla 3.17: Resultados F02 - 100 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	1.3838895666751221E-8	2.7933273302328512E-8
LS	1000066	1.9627324395934604	24.553211279516155
Híbrido 1	1000495	0.6340575072696654	1.21270878290056
Híbrido 2	1000210	1.9297291164149366	0.8471388126967989

Tabla 3.18: Resultados F02 - 100 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	1.107013329360024E-8	4.312837947277546E-8
LS	1000287	9.43989563111309E-4	0.028706282153530556
Híbrido 1	1000393	0.071535177878113	1.9085219293949356
Híbrido 2	1000990	1.1965075441977022	1.2285704898047096

Tabla 3.19: Resultados F02 - 100 dimensiones, $muestras = 12$ en LS.

3.4 F03 - Schwefel's Problem 1.2

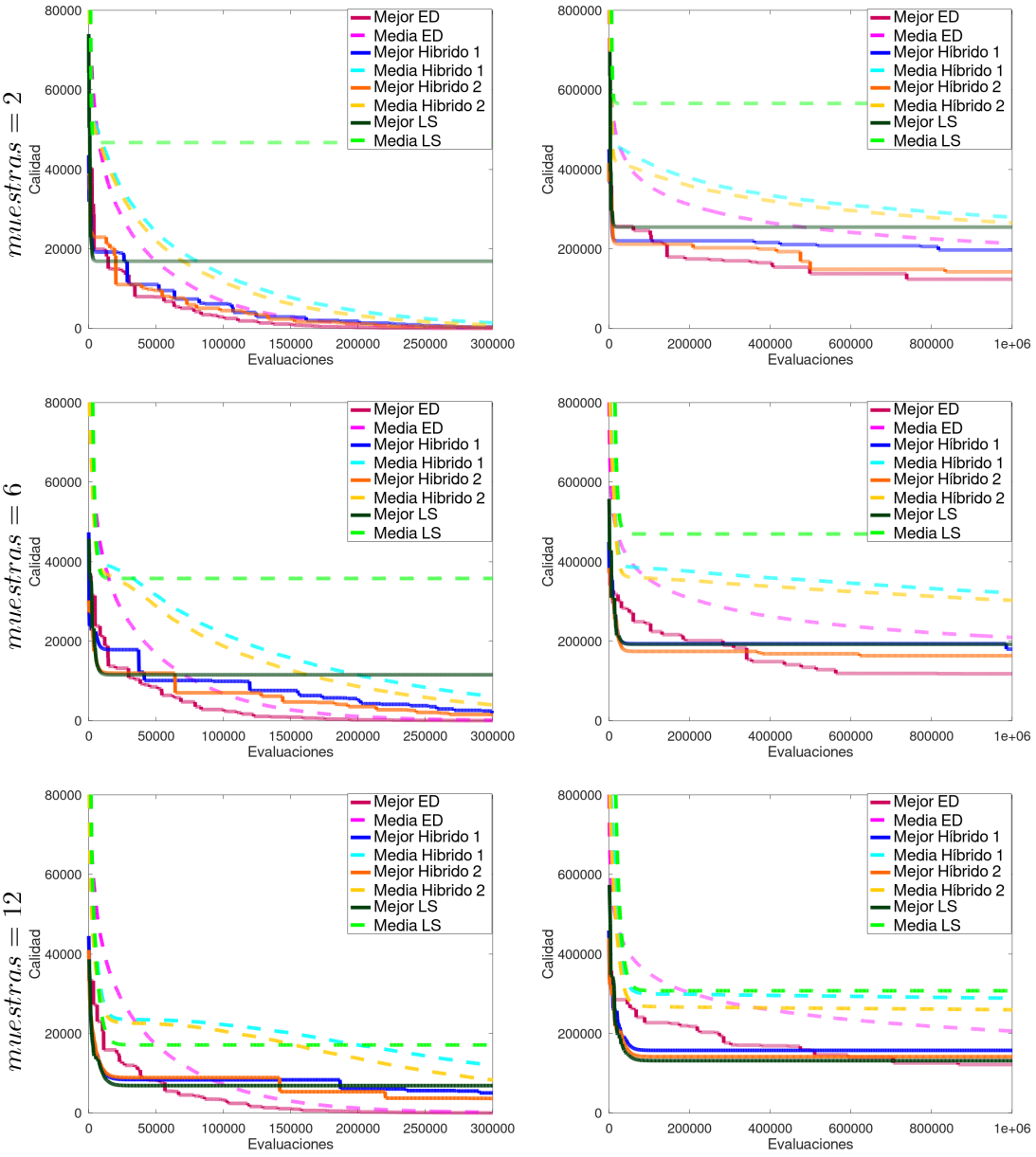
En esta ocasión la función F03 no es separable, concretamente el primer elemento del vector solución tiene mayor peso en el resultado de evaluar la función de *fitness* que el segundo elemento, el segundo elemento que el tercero y así de forma sucesiva. También cabe destacar que, dada la no separabilidad de la función, en muchas ocasiones, es posible que LS establezca un valor para una dimensión más alejado del mínimo óptimo (0 en este caso) que el propio valor inicial para esa dimensión. Esto se debe al mayor peso que pueden tener otras dimensiones en la función de *fitness*.

LS obtiene los peores resultados de forma clara en todos los escenarios, pese a evolucionar rápidamente en las primeras fases debido a la sencillez de realizar explotación en una función unimodal. En cambio, en fases más avanzadas, LS se estanca. Teniendo en cuenta las características de F03 explicadas previamente, es sencillo entender cuál es el problema. En la 1ª fase de LS, cada parámetro se evalúa de forma independiente, pero en la 2ª fase, los cambios realizados en cada parámetro de manera individual, se evalúan de forma conjunta, empeorando en muchas ocasiones el *fitness* obtenido en la anterior iteración de LS, debido a la no separabilidad de F03. Otro detalle de interés es observar la mejora en los resultados de LS al aumentar el número de muestras. Probablemente esto se deba a que aumentando el número de muestras, también aumentamos el número de vectores solución a evaluar en la 2ª fase de LS, haciendo más probable que al menos uno de estos consiga mejorar el *fitness* del vector inicial. De esta esta manera no se estarían ignorando las evaluaciones realizadas en la 1ª fase del algoritmo, donde cada parámetro se evalúa por separado.

ED obtiene los mejores resultados en todos los escenarios. La estrategia de ED se ve mucho menos perjudicada en los problemas descritos previamente para LS. Esto se debe principalmente a la facilidad de priorizar la selección de genes relevantes (parámetros con mayor peso en la función *fitness*).

En las versiones híbridas baja el rendimiento cuando el número de muestras aumenta. Al aumentar el número de muestras, también se dedica un número de evaluaciones mayor en una estrategia que no está funcionando bien (LS), por lo que no es raro esa pérdida de rendimiento.

Cabe destacar o reseñar que las versiones híbridas consiguen no estancarse a diferencia de LS. También se puede observar cómo la desviación típica, respecto a LS, se reduce en gran medida en todos los escenarios.



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.5: Gráfica de resultados: Función F03 - Schwefel's Problem 1.2. Todas las gráficas son el promedio de 1000 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	181.0639555983691	85.48264776370661
LS	300060	46754.118656094135	15185.815222291827
Híbrido 1	300043	1415.2923754185897	456.21625306021167
Híbrido 2	300114	888.7856352122811	296.030143101737

Tabla 3.20: Resultados F03 - 30 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	172.1374058243506	71.09605109103617
LS	300071	35793.69168988147	13570.275340187896
Híbrido 1	300205	6077.740001720598	1455.6946042195257
Híbrido 2	300169	3977.166690807197	1099.7375584802417

Tabla 3.21: Resultados F03 - 30 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	203.97529293073507	99.42110183017083
LS	300140	17134.396576737734	6649.564753897979
Híbrido 1	300127	11983.807751286313	2361.7131979973256
Híbrido 2	300301	8323.749934001135	1899.166667283923

Tabla 3.22: Resultados F03 - 30 dimensiones, $muestras = 12$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	213351.34566640423	21017.539610016986
LS	1000177	565985.9863227775	182825.0329693057
Híbrido 1	1000022	279615.532900222	25873.810671003383
Híbrido 2	1000084	265575.9652464955	25069.09250607599

Tabla 3.23: Resultados F03 - 100 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	209517.52277997337	21284.52757105569
LS	1000066	469746.7930255069	405364.43395248393
Híbrido 1	1000495	321567.39850847085	32489.97421283583
Híbrido 2	1000210	303056.7606965006	33000.75103384151

Tabla 3.24: Resultados F03 - 100 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	206338.47114239386	20443.585120125594
LS	1000287	307630.3614245502	284713.63003053906
Híbrido 1	1000393	288931.92312567105	52823.37924704615
Híbrido 2	1000990	259835.6385220362	51452.45119548238

Tabla 3.25: Resultados F03 - 100 dimensiones, $muestras = 12$ en LS.

3.5 F04 - Schwefel's Problem 2.21

Observando la gráfica de F04 (Apéndice A.2), puede parecer una función perfecta para un algoritmo de búsqueda local como LS, pero en los resultados se observa lo contrario. Esto se debe a que el coste de la función de *fitness* depende únicamente de un único parámetro, el más alejado (el que tiene el valor absoluto máximo) de su valor en la solución óptima, tal como se indica en la definición de la función en el Apéndice A.1.

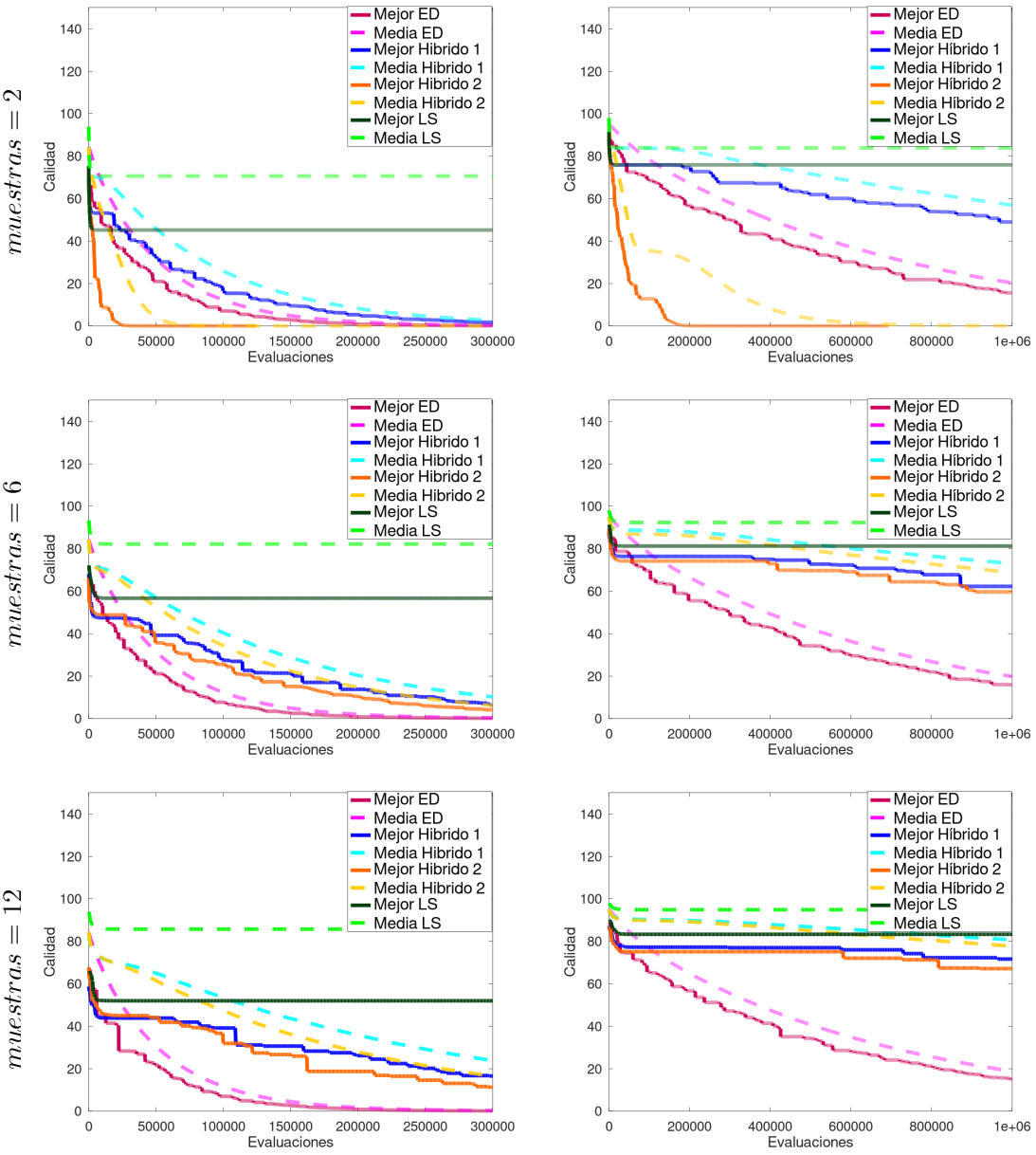
LS obtiene los peores resultados, el algoritmo se estanca y no consigue progresar. Los malos resultados pueden explicarse principalmente por dos razones. Primero, se desperdician muchas evaluaciones en el 1º paso de LS, puesto que la función de coste no devolverá otro resultado salvo que se realice un cambio en el parámetro peor posicionado (el que tiene el valor máximo en valor absoluto). Es decir, LS es claramente ineficiente con la función F04, ya que LS bosqueja en cada dimensión por separado, y la definición de F04 implica el máximo (en valor absoluto) en todas las dimensiones. Se puede observar en los resultados cómo el aumento del número de muestras (por tanto, aumento de número de evaluaciones por iteración) empeora los resultados. Esto a su vez provoca que entre iteraciones de LS solo un único parámetro pueda sufrir cambios. Segundo, en el más optimista de los casos todos los parámetros se habrán reducido al menos una vez en un número de iteraciones de LS igual al número de dimensiones del problema. Si, por ejemplo, tuviésemos una dimensionalidad igual a 100, en el más optimista de los casos, LS tendría que realizar 100 iteraciones para que la magnitud de todos los parámetros se haya reducido al menos 1 vez. En ese tiempo, el intervalo entre muestras a través del factor de reducción, ya se habría reducido 100 veces. Esto es precisamente lo que provoca que LS se estanque, ya que se estaría intentando desplazar un parámetro de una magnitud alta, una magnitud ridículamente pequeña. Por lo tanto, se deduce que ajustar el factor de reducción a un valor especifico en este problema, es determinante en los resultados.

ED consigue el mejor resultado en 4 de los 6 escenarios planteados, obteniendo un segundo lugar en los 2 restantes. La exploración de ED es claramente una estrategia más robusta a la búsqueda local de LS al enfrentar las dificultades que plantea F04.

En cuanto a las versiones híbridas, estas consiguen, a diferencia de LS, no estancarse. Con dimensionalidad 30, LS logra, al menos, unos resultados mucho más cercanos a ED que a LS. También se puede observar en las versiones híbridas cómo el aumento de número de muestras empeora sus resultados.

Cabe destacar los excelentes resultados logrados por el híbrido modificado en los escenarios con un número de muestras establecidas en 2. Este obtiene los mejores resultados en estos escenarios y unas soluciones muy cercanas a la solución óptima. Como se ha explicado anteriormente, únicamente las evaluaciones sobre el parámetro peor posicionado serán efectivas en el 1º paso de LS, esto explica por qué en todos los escenarios el híbrido modificado obtiene

mejores resultados que el no-modificado, puesto que este realiza menos evaluaciones en los genes que puedan estar mejor determinados en la población. Probablemente la hibridación esté funcionando en este caso porque esta esté garantizando a ED que, en cada generación, el peor gen del mejor individuo sea mejorado, a costa de un número de evaluaciones lo suficientemente bajo.



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.6: Gráfica de resultados: Función F04 - Schwefel's Problem 2.21. Todas las gráficas son el promedio de 1000 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	0.29701359991188864	0.08895330135014123
LS	300060	70.59474680922683	5.2591282814925195
Híbrido 1	300043	2.592991533194165	0.4065400168908524
Híbrido 2	300114	7.259729102315795E-10	7.616006752085484E-9

Tabla 3.26: Resultados F04 - 30 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	0.30437939430663496	0.11041272490059058
LS	300071	82.18757164860007	7.06509783778678
Híbrido 1	300205	10.17207403337342	1.2480656660174516
Híbrido 2	300169	6.181715467400919	0.8364526587467568

Tabla 3.27: Resultados F04 - 30 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	0.2491016782767655	0.07180781675211349
LS	300140	85.71712410550403	8.251131162679503
Híbrido 1	300127	23.805805763922972	2.2063791096598395
Híbrido 2	300301	16.427203637682112	1.6938715746901627

Tabla 3.28: Resultados F04 - 30 dimensiones, $muestras = 12$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	20.232495242827834	2.203409744488792
LS	1000177	83.88811234054936	2.1494743343346885
Híbrido 1	1000022	56.9164912370151	2.256485387973393
Híbrido 2	1000084	0.08199705578705214	0.3720637148856419

Tabla 3.29: Resultados F04 - 100 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	19.759778524271926	1.5942600515699341
LS	1000066	92.3448547767764	2.518301805173959
Híbrido 1	1000495	72.9244886220677	2.272141664835916
Híbrido 2	1000210	68.72655139910451	2.233069360684558

Tabla 3.30: Resultados F04 - 100 dimensiones, $muestras = 6$ en LS.

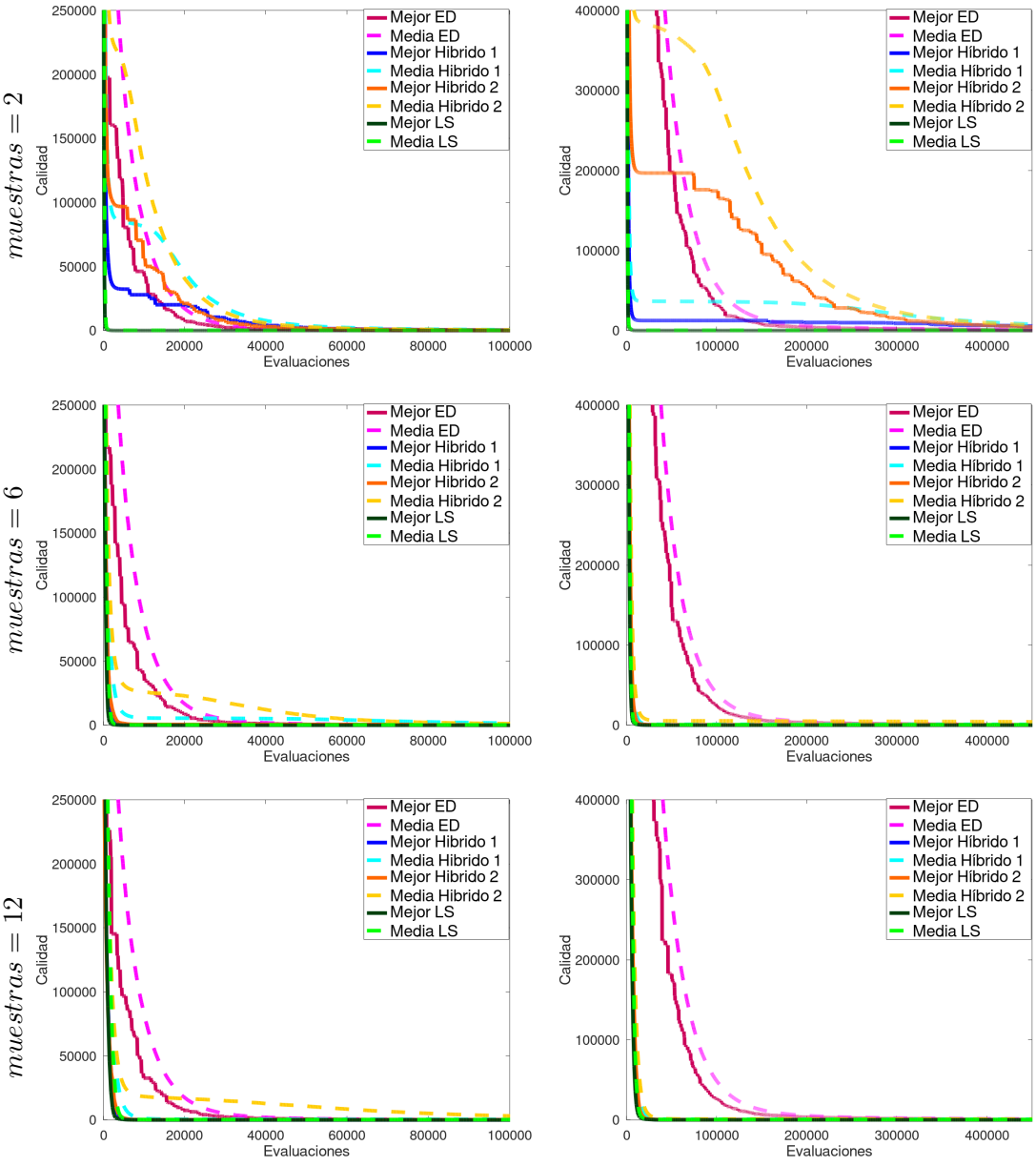
Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	18.732895553252312	1.8309427024385847
LS	1000287	94.86764559354862	2.6357420942365297
Híbrido 1	1000393	80.7028067294561	2.224128829323998
Híbrido 2	1000990	77.68988684011342	2.129227370396929

Tabla 3.31: Resultados F04 - 100 dimensiones, $muestras = 12$ en LS.

3.6 F05 - Generalized Rosenbrock's Function

A pesar de ser el algoritmo que más lento progresa en las primeras evaluaciones, ED es el algoritmo mejor posicionado en los resultados finales, en todos los escenarios. De nuevo, al igual que en el caso de F04, la propia definición de F05 dificulta la búsqueda mediante LS. La función F05 (Apéndice A.1), implica la minimización entre genes consecutivos ($x_{i+1} - x_i^2$). La búsqueda local en una dimensión "i" debe minimizar la distancia entre x_{i+1} y x_i^2. Esto es "engañoso" para la búsqueda local de LS, ya que se incrementará o disminuirá el valor óptimo de x_i (en cada búsqueda de LS en cada dimensión), dependiendo del valor que se encuentre en x_{i+1}, es decir, no moviendo x_i necesariamente hacia el valor óptimo en 1. Al contrario, DE, como búsqueda global, tiene la capacidad de lograr una combinación de genes todos lo más cercanos a 1.

LS progresa de una forma muy rápida en las primeras evaluaciones, pero en los resultados finales se queda lejos de los buenos resultados obtenidos por ED (por la dificultad en la propia definición de F05 explicada previamente). Esta situación probablemente está relacionada con que F05 esté premiando mucho la estrategia de explotación de LS, hasta llegar a cierto punto dónde la no-separabilidad de F05 comienza a premiar, en mayor medida, estrategias de exploración como la de ED.



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.7: Gráfica de resultados: Función F05 - Generalized Rosenbrock's Function. Todas las gráficas son el promedio de 1000 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	4.502809177547864E-4	6.025204912739704E-4
LS	300060	198.93475242535322	182.74546763744073
Híbrido 1	300043	0.2164382078027869	0.1143796927970594
Híbrido 2	300114	0.05718698247662096	0.036795797272333786

Tabla 3.32: Resultados F05 - 30 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	3.706022598609599E-4	5.662404724928879E-4
LS	300071	283.64555951212486	213.4724330577206
Híbrido 1	300205	27.85391820924537	13.361507684360472
Híbrido 2	300169	5.190910059686579	2.2027601770961867

Tabla 3.33: Resultados F05 - 30 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	6.164625367834083E-4	0.001188121605824191
LS	300140	265.91994147046506	216.00574355102276
Híbrido 1	300127	346.68202426657666	130.56209188253123
Híbrido 2	300301	156.31442029876237	51.28519469555503

Tabla 3.34: Resultados F05 - 30 dimensiones, $muestras = 12$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.24164325808766388	0.39060077484294736
LS	1000177	343.14547888620734	214.46024798220986
Híbrido 1	1000022	1938.3939412146049	189.51467226157934
Híbrido 2	1000084	1511.631619135431	206.75340979606793

Tabla 3.35: Resultados F05 - 100 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.07625764825557825	0.09561612637533534
LS	1000066	623.1252433547689	348.363583019355
Híbrido 1	1000495	1109.801420720205	307.2547716672876
Híbrido 2	1000210	2558.402293080738	1041.9505590128347

Tabla 3.36: Resultados F05 - 100 dimensiones, $muestras = 6$ en LS.

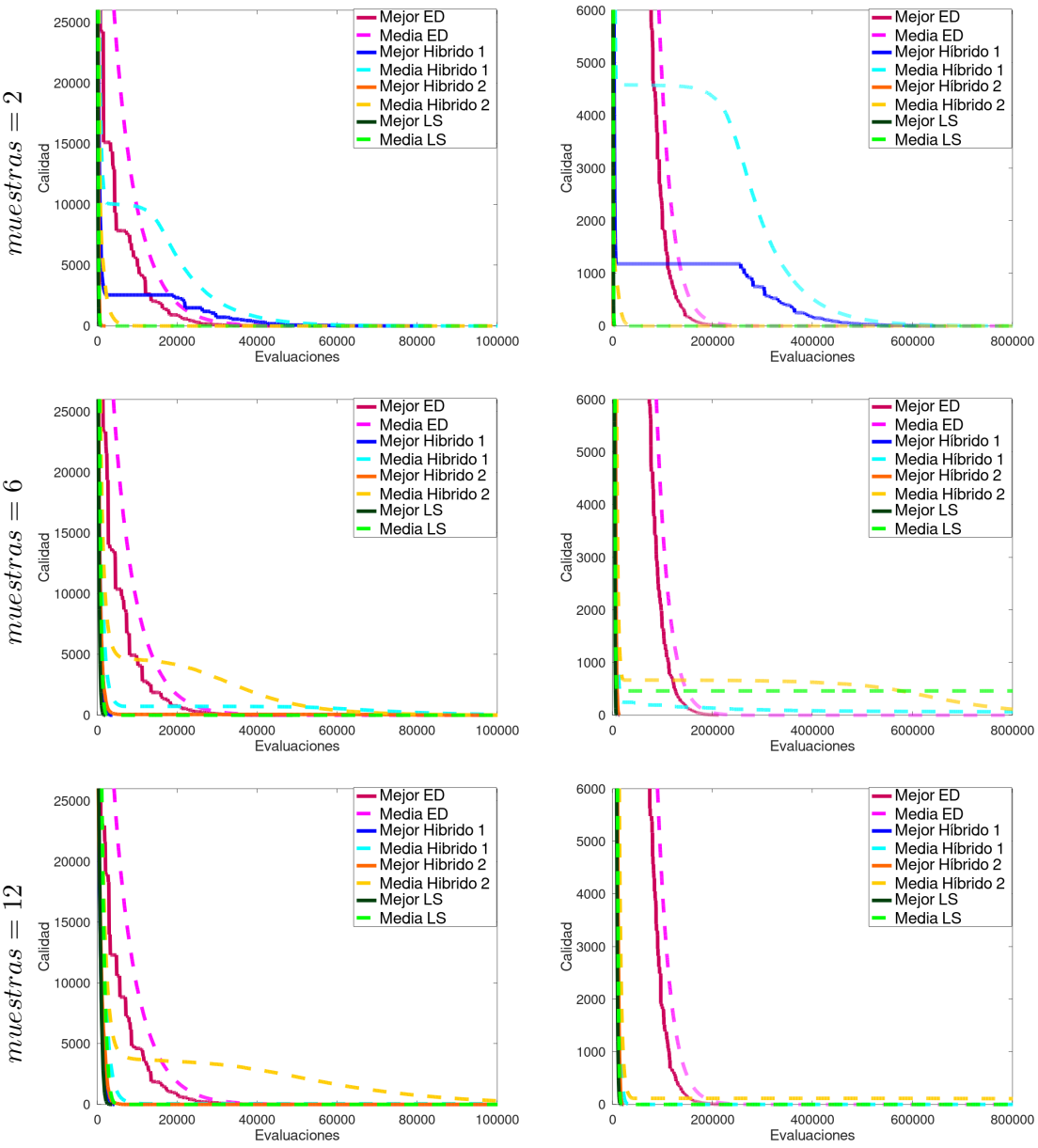
Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.1551265619106764	0.33755576793302305
LS	1000287	557.2136004279595	223.67283574644662
Híbrido 1	1000393	864.6342268527886	256.2659631906127
Híbrido 2	1000990	1061.817144668974	648.4311776767806

Tabla 3.37: Resultados F05 - 100 dimensiones, $muestras = 12$ en LS.

3.7 F06 - Step Function

La peculiaridad de F06 es que dispone de infinitos mínimos globales, cualquier valor de un parámetro o gen entre [−0.5,0.5) es un valor que puede formar parte de una de las múltiples soluciones óptimas posibles. Otra cosa importante a tener en cuenta en F06, es que si se quiere modificar un parámetro o gen añadiendo o restando una cantidad incluida en ese rango, la función de *fitness* no devolverá un coste distinto, porque no se estaría superando el "escalón" de la función. Esto resulta especialmente problemático en algoritmos de búsqueda local como LS, puesto que esto puede impedirles obtener la información necesaria para conocer si están encaminados en la dirección correcta. Esto es más acuciante a medida que se va decrementando el tamaño de paso en el muestreo realizado por LS en cada dimensión.

Todas las ejecuciones de ED consiguen llegar a una solución óptima, esta conclusión se puede obtener al observar una desviación típica igual a 0 en todos los casos. Su único inconveniente sería su lenta progresión en las primeras evaluaciones, en comparación con los demás algoritmos propuestos. Por contra, LS, por la razón indicada, presenta una rápida progresión en las primeras iteraciones, pero puede terminar en un valor subóptimo por lo anteriormente indicado de la disminución paulatina del tamaño de paso. De todos modos, este problema en LS solo se pone de manifiesto con alta dimensionalidad (100), con lo cual las versiones híbridas se resienten igualmente frente a DE. Hay que fijarse también, que con $muestras = 12$, el problema es menor, precisamente por estar utilizando tamaños de pasos menores al existir más muestras en cada dimensión.



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.8: Gráfica de resultados: Función F06 - Step Function. Todas las gráficas son el promedio de 1000 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	0.0	0.0
LS	300060	0.0	0.0
Híbrido 1	300043	0.0	0.0
Híbrido 2	300114	0.0	0.0

Tabla 3.38: Resultados F06 - 30 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	0.0	0.0
LS	300071	9.999999999999992E-4	0.03162277660168412
Híbrido 1	300205	0.0	0.0
Híbrido 2	300169	0.0	0.0

Tabla 3.39: Resultados F06 - 30 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	0.0	0.0
LS	300140	0.009999999999999997	0.09996996545960213
Híbrido 1	300127	0.0	0.0
Híbrido 2	300301	0.0	0.0

Tabla 3.40: Resultados F06 - 30 dimensiones, $muestras = 12$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.0	0.0
LS	1000177	0.0	0.0
Híbrido 1	1000022	0.0	0.0
Híbrido 2	1000084	0.0	0.0

Tabla 3.41: Resultados F06 - 100 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.0	0.0
LS	1000066	461.3180000000006	8963.407818044767
Híbrido 1	1000495	50.132999999999996	42.8510621166445
Híbrido 2	1000210	21.029000000000003	12.290370710935129

Tabla 3.42: Resultados F06 - 100 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.0	0.0
LS	1000287	0.007999999999999993	0.17879588353194092
Híbrido 1	1000393	0.5760000000000028	5.869128612551458
Híbrido 2	1000990	100.18200000000002	157.80879725950768

Tabla 3.43: Resultados F06 - 100 dimensiones, $muestras = 12$ en LS.

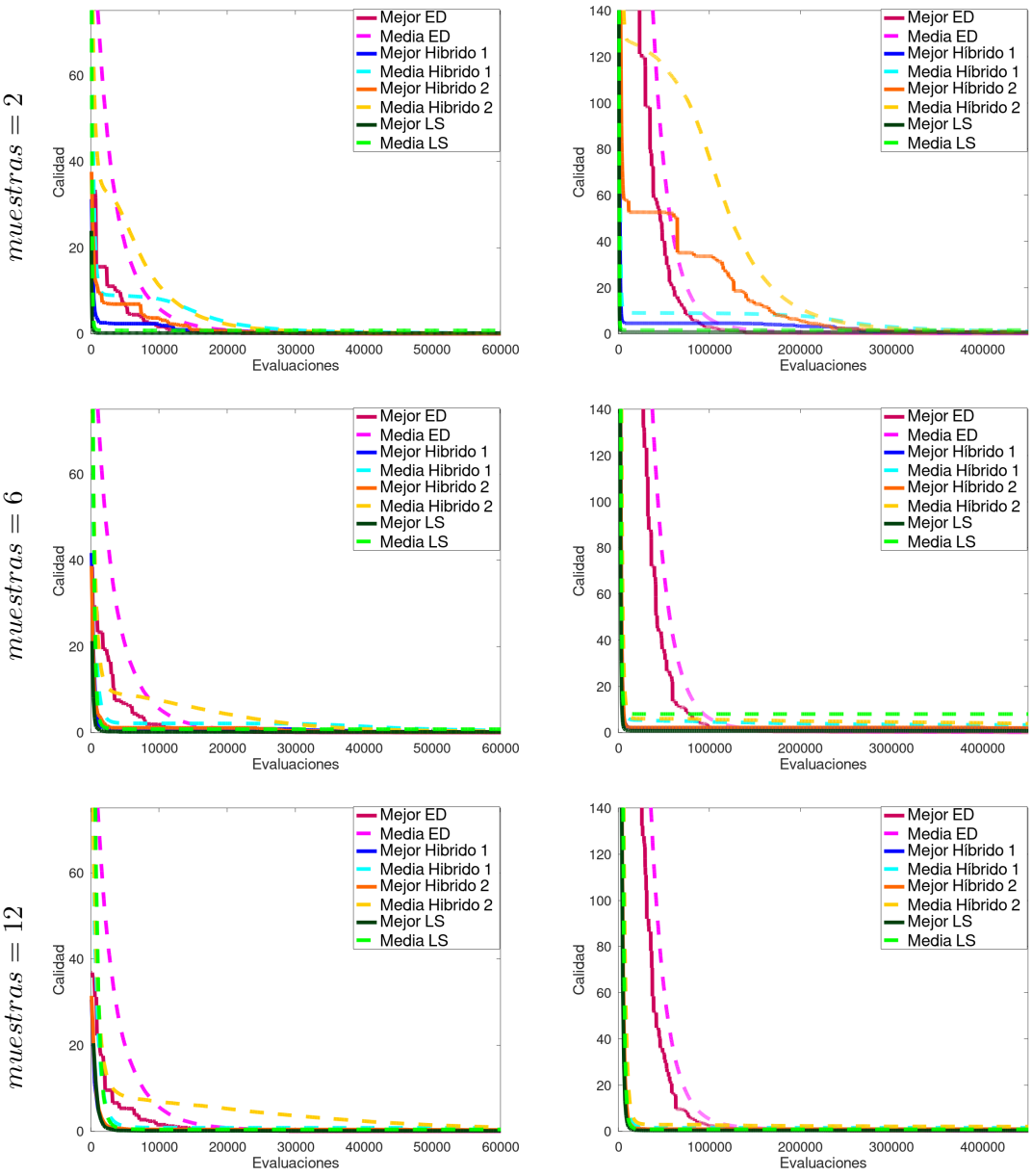
3.8 F07 - Quartic Function with Noise

Todos los algoritmos consiguen unos buenos resultados, exceptuando LS, en el escenario de 100 dimensiones con 6 muestras (tabla 3.48). Dada la alta desviación típica (más de 10 veces superior a la media), y que ésta solo aumenta de forma clara en el caso de pasar de 30 a 100 dimensiones con 6 muestras, no sería raro afirmar que, muy probablemente, esto se deba a un sesgo producido por algunas de las ejecuciones que han padecido excesiva desdicha.

ED obtiene los mejores resultados en todos los casos planteados. Esta función es muy clarificadora de la ventaja de la búsqueda global de ED, ya que, como todo algoritmo de búsqueda local, LS puede verse atascado en un mínimo local ante la presencia de ruido en el espacio de calidad, que hace más ”rugoso” ese espacio por el que bosqueja LS.

Cabe destacar la rápida convergencia de LS hacia la solución óptima, aunque el ruido de la función parece causarle muchos problemas a la hora de poder dirigirse hacia la solución óptima. El número de muestras parece no tener un impacto relevante para LS en este problema.

En cuanto a los híbridos, un número mayor de muestras si parece estar teniendo un impacto negativo en sus resultados. Por lo general los híbridos están consiguiendo ser mas robustos que LS, ya que su coste de *fitness* alcanzado, por lo general es mucho más cercano al coste alcanzado por ED que LS.



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.9: Gráfica de resultados: Función F07 - Quartic Function with Noise. Todas las gráficas son el promedio de 1000 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	0.01570878259519811	0.00339738954598381
LS	300060	0.7979404497092234	0.3214607571239054
Híbrido 1	300043	0.0235905025775971	0.005604366252926797
Híbrido 2	300114	0.020765328238793367	0.004778954459048073

Tabla 3.44: Resultados F07 - 30 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	0.015165456048506758	0.0032548944350875107
LS	300071	0.7940240285225566	0.23074133934418173
Híbrido 1	300205	0.04046272730151001	0.010630781455147433
Híbrido 2	300169	0.03191364536725038	0.008118129163030036

Tabla 3.45: Resultados F07 - 30 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	0.015145519967253262	0.00316876439401143
LS	300140	0.4478830716557113	0.12622435843178986
Híbrido 1	300127	0.07472803468115	0.01928565133290521
Híbrido 2	300301	0.05408083371443653	0.014042008406890764

Tabla 3.46: Resultados F07 - 30 dimensiones, $muestras = 12$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.10139755442911917	0.013015931039128117
LS	1000177	1.6609357392438264	0.3840134462709998
Híbrido 1	1000022	0.343073126269383	0.04348936176335875
Híbrido 2	1000084	0.2829465147724265	0.037094423700720566

Tabla 3.47: Resultados F07 - 100 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.09986386479441674	0.012374289641441147
LS	1000066	8.048184436805734	85.15259416654305
Híbrido 1	1000495	1.1140971950215814	0.1528495619547927
Híbrido 2	1000210	0.776700996575987	0.10584122583185746

Tabla 3.48: Resultados F07 - 100 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.09766158522386381	0.012019753818503544
LS	1000287	0.8807489746569405	0.16956385923782097
Híbrido 1	1000393	1.5318096126612655	0.2692775771228786
Híbrido 2	1000990	1.7676687172923804	0.34166610160464134

Tabla 3.49: Resultados F07 - 100 dimensiones, $muestras = 12$ en LS.

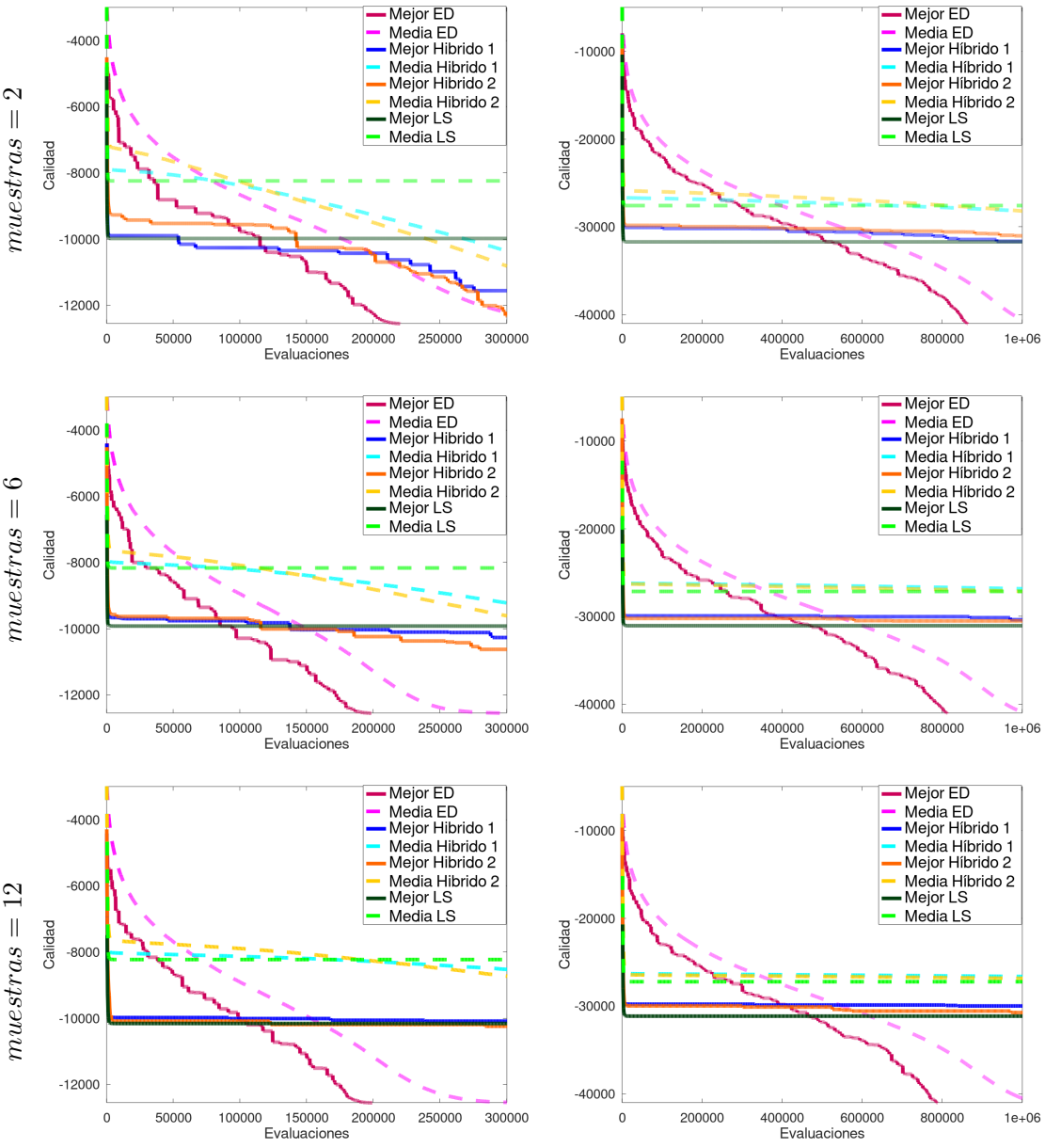
3.9 F08 - Generalized Schwefel's Problem 2.26

Si se observa gráficamente F08 (anexo A.1), se puede observar cómo F08 se presta a que algoritmos de búsqueda local como LS puedan reducir el *fitness* rápidamente, pero de igual manera condena a estos algoritmos a estancarse en mínimos locales de muy difícil salida. Precisamente esto puede observarse en el comportamiento de LS en todos los escenarios planteados.

En las gráficas se puede apreciar claramente en qué momento LS se posiciona en un mínimo local y, en ese momento, también se puede apreciar un estancamiento parcial en las versiones híbridas. El estancamiento en las versiones híbridas es más o menos pronunciado dependiendo del número de muestras. Al aumentar el número de muestras, los híbridos también dedican un mayor porcentaje de evaluaciones a LS, esto explica por qué el estancamiento es más pronunciado. Este efecto se aprecia de forma mucho más reducida cuando la dimensionalidad del problema se eleva a 100. Al aumentar las dimensiones, el número de evaluaciones por iteración dedicadas a ED por los algoritmos híbridos, siguen siendo las mismas (la población establecida), en cambio el número de evaluaciones dedicadas a LS aumenta proporcionalmente al número de dimensiones. Esto explica por qué los híbridos tienen mayores dificultades para salir de los mínimos locales cuando la dimensionalidad del problema aumenta.

Los híbridos obtienen, en prácticamente todos los casos, mejores resultados que LS. A excepción de los escenarios con 100 dimensiones y 6 o 12 muestras, donde probablemente el porcentaje de evaluaciones dedicadas a ED no esté siendo suficiente para que los híbridos consigan salir del mínimo local donde se encuentran. Al mismo tiempo, LS consigue explotar el mínimo local donde se encuentra de mejor manera, obteniendo mejores resultados que los híbridos, mientras estos dedican parte de sus esfuerzos (evaluaciones de *fitness*), a intentar salir sin éxito, del mínimo local en el que se encuentran.

ED obtiene los mejores resultados en todos los casos. Consigue converger hacia la solución óptima de forma constante durante todo el proceso.



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.10: Gráfica de resultados: Función F08 - Generalized Schwefel's Problem 2.26. Todas las gráficas son el promedio de 1000 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	-12215.820476918696	619.8249120533882
LS	300060	-8242.469577179816	606.4469270864284
Híbrido 1	300043	-10364.50982912835	309.80073545236786
Híbrido 2	300114	-10820.029559178363	391.4815249909566

Tabla 3.50: Resultados F08 - 30 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	-12557.155803842054	110.94763332206338
LS	300071	-8167.158277874398	657.0147916725542
Híbrido 1	300205	-9226.596051926363	298.3738369592413
Híbrido 2	300169	-9618.240743452858	288.8816410775468

Tabla 3.51: Resultados F08 - 30 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	-12545.505889882143	154.53495434570303
LS	300140	-8228.213258783435	571.4090252655238
Híbrido 1	300127	-8525.48597443119	447.73586230146094
Híbrido 2	300301	-8732.968927093825	321.97150494746916

Tabla 3.52: Resultados F08 - 30 dimensiones, $muestras = 12$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	-40603.87698865687	1647.9544665602727
LS	1000177	-27576.898669367754	1050.0108938590133
Híbrido 1	1000022	-28185.33539564632	1067.9700561978866
Híbrido 2	1000084	-28202.753959097234	903.2058286511658

Tabla 3.53: Resultados F08 - 100 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	-40902.49801844451	1505.4718134234654
LS	1000066	-27154.646663204694	1967.2488548243812
Híbrido 1	1000495	-26847.81740666894	1230.3764450349054
Híbrido 2	1000210	-27129.29389275961	1151.9413359544358

Tabla 3.54: Resultados F08 - 100 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	-40521.304273026326	2037.3170782334141
LS	1000287	-27234.95102010765	1527.5239950174741
Híbrido 1	1000393	-26638.103472425217	1283.662601993032
Híbrido 2	1000990	-26849.090537214084	1162.2932510043622

Tabla 3.55: Resultados F08 - 100 dimensiones, $muestras = 12$ en LS.

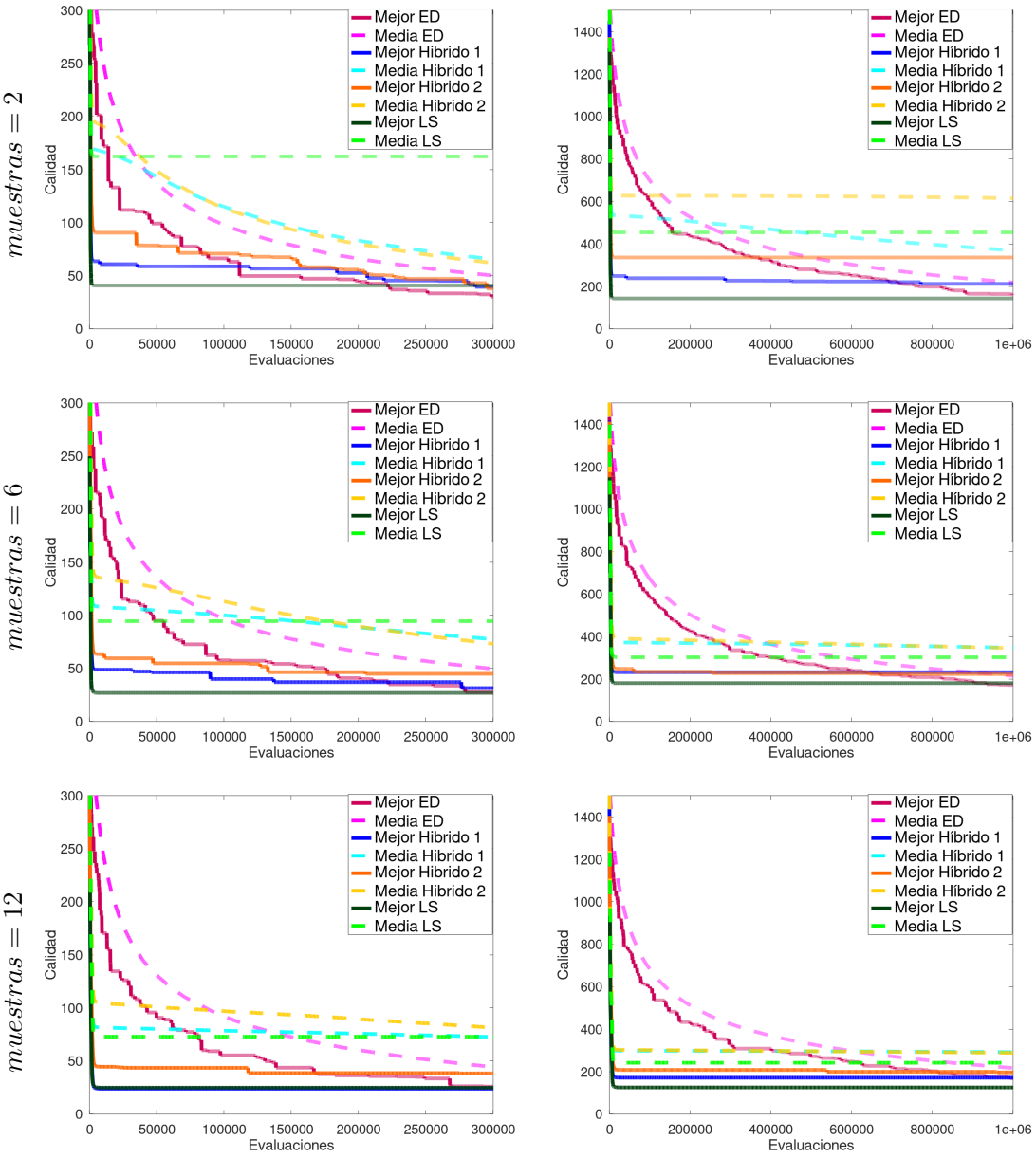
3.10 F09 - Generalized Rastrigin's Function

F09 es una función con características similares a F08. Dispone de un espacio de búsqueda grande, con un número muy elevado de mínimos locales.

ED es el algoritmo que más consigue acercarse a la solución óptima en todos los escenarios, como cabría esperar en un problema como F09, dada la naturaleza exploradora de ED.

Observando las gráficas se puede apreciar, al igual que en F08, cómo LS evoluciona muy rápidamente y acaba estancándose en un mínimo local. Aumentando el número de muestras, LS parece conseguir llegar a mínimos locales mejor posicionados. Esto puede deberse a que, probablemente, con un número de muestras superior, LS tiene mayor facilidad para detectar otros mínimos locales próximos más ventajosos, al menos mientras el intervalo entre muestras no se haya reducido en gran medida.

Las estrategias híbridas parecen estar teniendo problemas en el momento de aumentar el número de muestras y la dimensionalidad del problema. Es decir, empeoran al depender más de LS, incluso llegando al punto de obtener peores resultados que LS puro, como se puede apreciar en las tablas 3.60 y 3.61. Con un número de evaluaciones más elevado, cabría esperar que los híbridos acabasen superando a LS en todos los casos, ya que LS estaría irremediablemente atrapado en un mínimo local, mientras que los híbridos, en mayor o menor medida, podrían acabar saliendo de esta situación, más o menos rápido dependiendo del porcentaje de evaluaciones dedicadas a LS. Si observamos las gráficas de 30 dimensiones con 2 o 6 muestras, en en las cuales los híbridos dedican menos porcentaje de evaluaciones a LS que en el resto de escenarios, podemos observar cómo, a pesar de cierta ralentización cercana al punto donde LS se estanca, los híbridos consiguen seguir avanzando.



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.11: Gráfica de resultados: Función F09 - Generalized Rastrigin's Function. Todas las gráficas son el promedio de 1000 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	50.14915116940461	7.021249836156661
LS	300060	162.35169088471784	59.264758450046386
Híbrido 1	300043	65.53703702355622	6.795237544003757
Híbrido 2	300114	61.95563473471041	7.0086620935389075

Tabla 3.56: Resultados F09 - 30 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	49.45559120028217	8.237579417169584
LS	300071	94.45846594320824	30.05099282409132
Híbrido 1	300205	77.41065134828266	11.88842364158522
Híbrido 2	300169	73.12685017323894	8.285446528955081

Tabla 3.57: Resultados F09 - 30 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	44.363230981294834	6.090813289089116
LS	300140	72.94661085713633	27.775344683011557
Híbrido 1	300127	72.68514274937962	15.99152150777772
Híbrido 2	300301	81.31518847358646	13.726674046518594

Tabla 3.58: Resultados F09 - 30 dimensiones, $muestras = 12$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	218.09269747556704	14.320878433969915
LS	1000177	454.1641745219161	169.88690489065135
Híbrido 1	1000022	369.09312569090565	57.3839602573071
Híbrido 2	1000084	615.2281003000805	90.02123735008597

Tabla 3.59: Resultados F09 - 100 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	214.99157305860226	12.131614776607758
LS	1000066	302.0920806997861	93.36964647765878
Híbrido 1	1000495	347.681262195811	57.622834704002635
Híbrido 2	1000210	345.53334091101516	57.35117601569342

Tabla 3.60: Resultados F09 - 100 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	217.04755710311483	12.486641310274662
LS	1000287	241.57995969187948	82.69093856984686
Híbrido 1	1000393	292.00847259020213	61.51580234983712
Híbrido 2	1000990	288.56018264689203	61.0783445550089

Tabla 3.61: Resultados F09 - 100 dimensiones, $muestras = 12$ en LS.

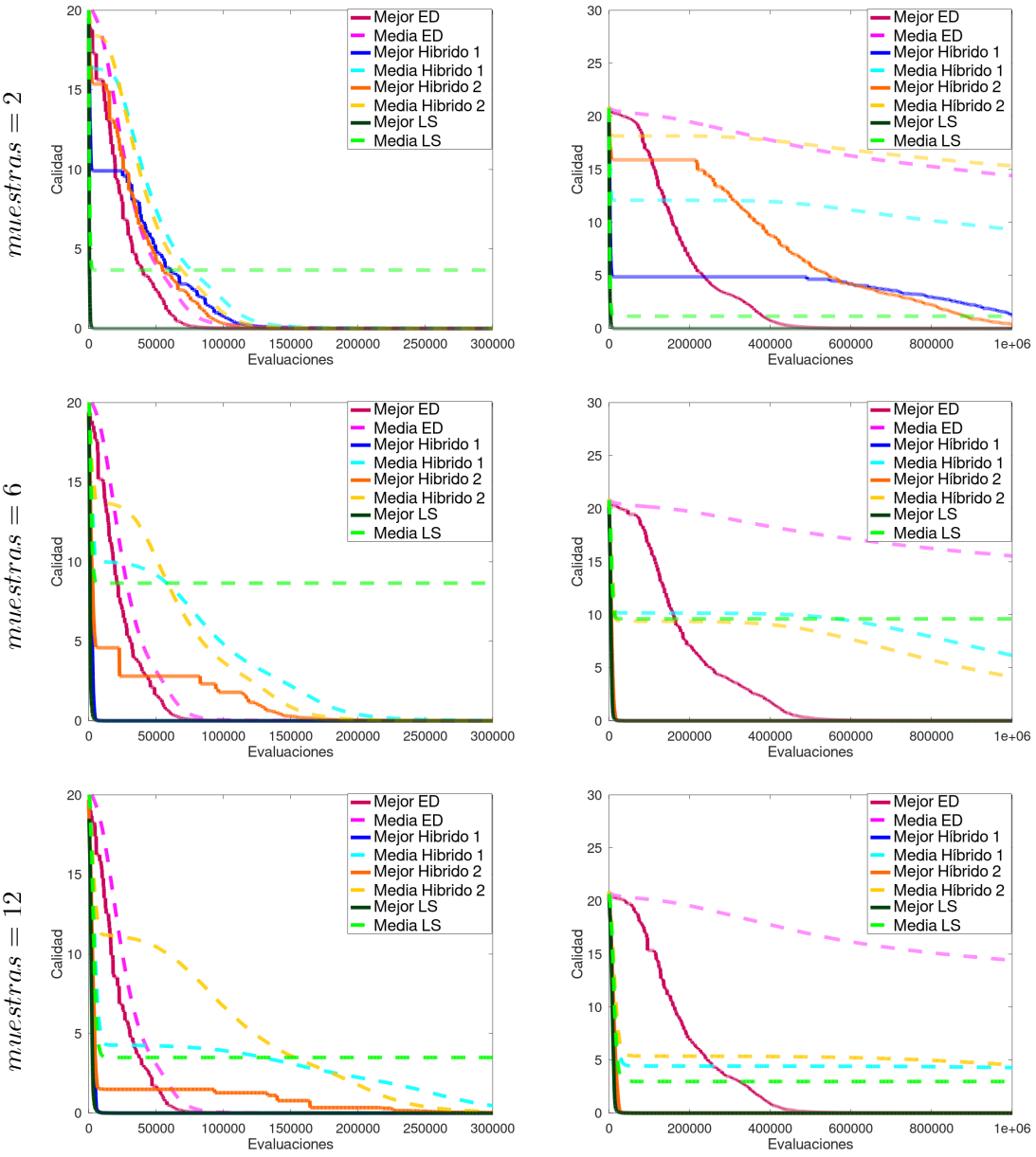
3.11 F10 - Ackley's Function

Primero, cabe remarcar la elevada desviación típica obtenida en los distintos escenarios (teniendo en cuenta su valor con respecto a la media obtenida). En las gráficas también se puede apreciar que, por lo general, la mejor ejecución de los distintos algoritmos dista mucho de la media obtenida. Por tanto, sería conveniente tener en cuenta que los datos de las ejecuciones independientes, podrían llevar a unas conclusiones diferentes a las que se pueden sacar observando las medias obtenidas. Todo lo anteriormente expuesto es un claro indicativo de la dificultad que presenta F10.

ED evoluciona rápidamente y obtiene los mejores resultados cuando la dimensionalidad del problema es baja. Todo lo contrario ocurre cuando la dimensionalidad es alta, obtiene los peores resultados y su evolución es muy lenta. La cosa cambia completamente si nos fijamos en la mejor ejecución de ED. Probablemente los resultados de ED en F10 estén dependiendo mucho de la suerte que este tenga, seleccionando los distintos individuos de la población para formar el vector donador. Todo parece indicar que el esquema utilizado para ED, en F10, no es el adecuado. Los problemas de ED en F10 podrían solucionarse utilizando otro esquema con mayor presión selectiva como podría ser DE/best/1/bin.

LS avanza rápidamente en las primeras evaluaciones y se estanca en un mínimo local. No se aprecia una tendencia en ningún caso, que indique que a mayor o menor número de muestras se mejorarán los resultados. Se puede observar cómo los resultados, al pasar de 2 a 6 muestras, empeoran, pero al incrementarse a 12 muestras, estos vuelven a mejorar. Esto puede indicar que, al igual que pasaba en F02, el número de muestras que se considera está influyendo en la probabilidad de que el punto seleccionado se sitúe en un lugar en el espacio de F10 más cercano a la solución óptima, en donde el progreso es mucho más sencillo.

En los escenarios de 30 dimensiones, los algoritmos híbridos parecen estar evitando de forma sencilla estancarse en un mínimo local, como en el caso de LS. Con 2 o 6 muestras, los híbridos tienen unos resultados cercanos a los alcanzados por ED, y la velocidad a la que evolucionan no dista mucho de este. En los escenarios con 100 dimensiones, el bajo rendimiento de ED, junto al fácil estancamiento de LS, parecen estar provocando que las estrategias híbridas no consigan funcionar correctamente. La excepción es si el número de muestras es igual a 6 ya que, en este caso, los algoritmos híbridos son los que consiguen los mejores resultados.



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.12: Gráfica de resultados: Función F10 - Ackley's Function. Todas las gráficas son el promedio de 1000 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	5.191913955257712E-4	0.01634281596496856
LS	300060	3.671805128711494	7.2506471914192705
Híbrido 1	300043	0.019781398325388017	0.5628104289661816
Híbrido 2	300114	1.5200407272449598E-4	0.004575517619051447

Tabla 3.62: Resultados F10 - 30 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	2.5493279323271413E-5	8.060142111066503E-4
LS	300071	8.653045457533002	8.142012940632787
Híbrido 1	300205	0.015177119531809918	0.09065947228942708
Híbrido 2	300169	0.0018390058349772324	8.026566726662545E-4

Tabla 3.63: Resultados F10 - 30 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	3.337322617369415E-9	2.2224747958739306E-8
LS	300140	3.496860626078518	6.536281794161647
Híbrido 1	300127	0.46670151238405644	0.37240622392647404
Híbrido 2	300301	0.08862377686365795	0.035546642370013384

Tabla 3.64: Resultados F10 - 30 dimensiones, $muestras = 12$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	14.408565166167284	8.084920290474939
LS	1000177	1.1577277099353465	4.258334898592053
Híbrido 1	1000022	9.343366448061575	6.488627621477715
Híbrido 2	1000084	15.330313293873582	5.923813015190183

Tabla 3.65: Resultados F10 - 100 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	15.536323170533128	7.395464226280971
LS	1000066	9.612356067945543	7.635279263909744
Híbrido 1	1000495	6.166427106764983	3.1952194686626094
Híbrido 2	1000210	4.144806566137629	2.60213347831336

Tabla 3.66: Resultados F10 - 100 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	14.402293414517338	8.159870067639984
LS	1000287	2.980269495384875	5.970251756728812
Híbrido 1	1000393	4.286646463360503	4.246499996842176
Híbrido 2	1000990	4.534296103893006	3.438406878215007

Tabla 3.67: Resultados F10 - 100 dimensiones, $muestras = 12$ en LS.

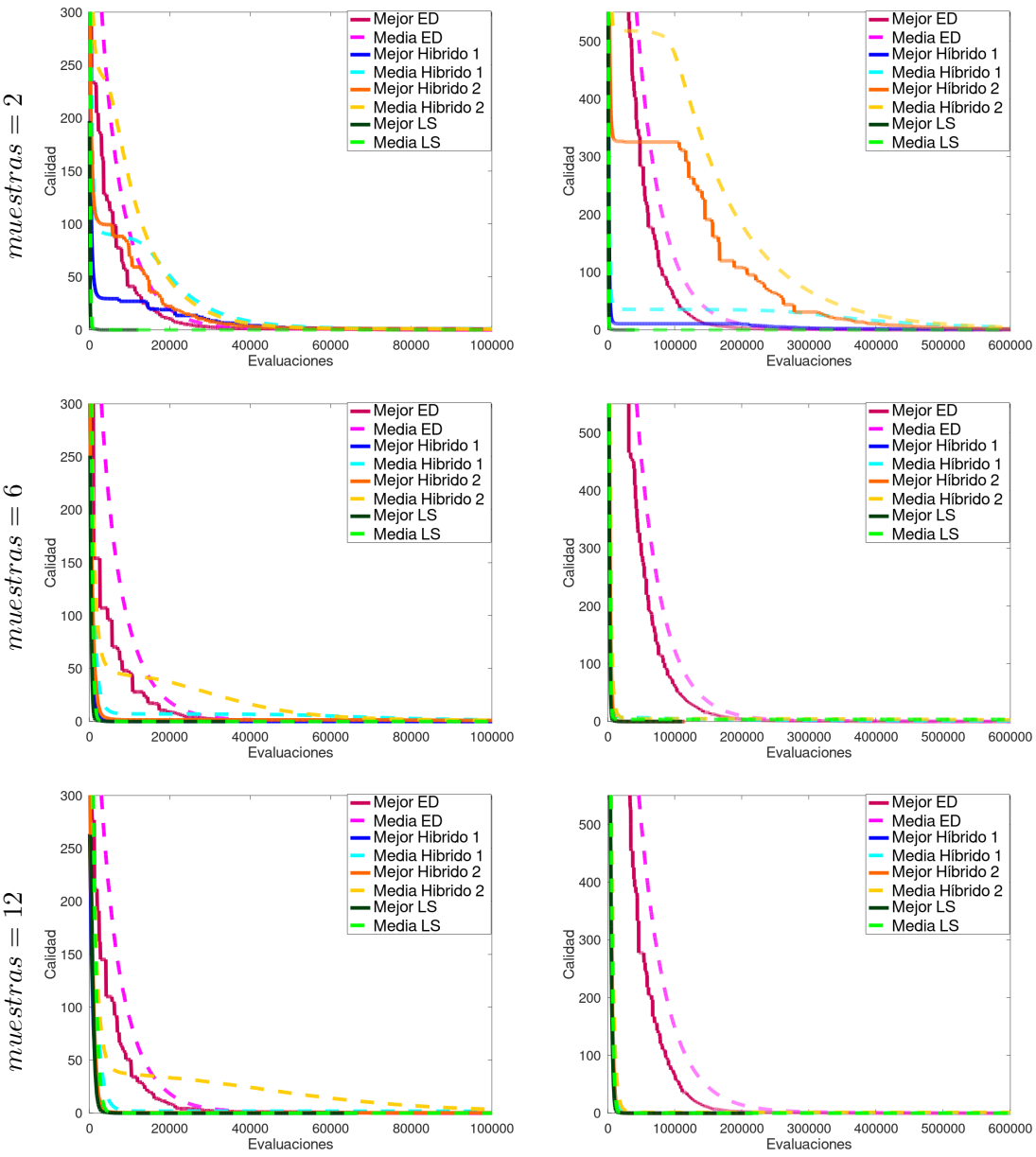
3.12 F11 - Generalized Griewank's Function

Cabe destacar principalmente en F11 que, a pesar de ser una función multimodal, el espacio entre los distintos mínimos locales es tan reducido que, mientras el intervalo entre muestras no se reduzca mucho, LS se comportará prácticamente como si estos mínimos locales no existiesen. Es por esto que hasta alcanzar un número de evaluaciones elevada, F11 resulta un problema muy sencillo para LS, como si se tratase, por ejemplo, de de una función como F01. Un buen ajuste del factor de reducción parece clave tanto para LS como para las versiones híbridas en un problema como F11.

Si bien LS no obtiene los mejores resultados finales en ningún escenario (en cuanto a la calidad promediada de las mejores soluciones finales), claramente se puede observar en las gráficas cómo LS consigue acercarse mucho a la solución óptima en un número de evaluaciones muy reducido. Alcanzado este punto, el intervalo entre muestras ya será demasiado pequeño, por tanto lo más seguro es que LS se encuentre situado en un mínimo local del cual no pueda salir.

ED no evoluciona tan rápido como LS, pero ED sí consigue los mejores resultados finales en todos los escenarios.

Los algoritmos híbridos no se muestran muy consistentes ante este problema. En ningún escenario alcanzan los resultados finales alcanzados por ED. En algunos escenarios consiguen mejorar los resultados de LS, pero del mismo modo también los empeoran en otros. Es decir, las versiones híbridas de nuevo se ven perjudicadas por el mal comportamiento del algoritmo de búsqueda local LS.



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.13: Gráfica de resultados: Función F11 - Generalized Griewank's Function. Todas las gráficas son el promedio de 1000 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	4.193534408614131E-14	5.494736556319464E-13
LS	300060	0.0064373441603762255	0.02832531495533133
Híbrido 1	300043	6.18614856038649E-7	6.49861028361788E-6
Híbrido 2	300114	1.608111956929999E-8	1.2660264265465896E-7

Tabla 3.68: Resultados F11 - 30 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	4.6393999753035776E-14	9.66330640111575E-13
LS	300071	0.01799829178191696	0.03874379415267914
Híbrido 1	300205	0.014333747686202634	0.022668892467919952
Híbrido 2	300169	6.040837936335775E-4	0.0022487286324400463

Tabla 3.69: Resultados F11 - 30 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	2.948497002108755E-14	3.357662890363125E-13
LS	300140	0.03560168114543189	0.038279557375551006
Híbrido 1	300127	0.5629374228016302	0.25166108708249735
Híbrido 2	300301	0.17393164304826234	0.09149280098930768

Tabla 3.70: Resultados F11 - 30 dimensiones, $muestras = 12$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	2.9803186874540713E-6	1.664577590943222E-5
LS	1000177	0.0024731915521274154	0.03134791068540237
Híbrido 1	1000022	1.229449480515925	0.1573079001963367
Híbrido 2	1000084	1.000258839295296	0.20059616718867215

Tabla 3.71: Resultados F11 - 100 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	2.3735537795007694E-6	1.1353383457614024E-5
LS	1000066	3.634547596257826	84.2570609452202
Híbrido 1	1000495	0.872732483806636	0.5965250599738793
Híbrido 2	1000210	1.0855875874663616	0.26650601799866136

Tabla 3.72: Resultados F11 - 100 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	2.5107796811752933E-5	1.0207833334063066E-4
LS	1000287	0.018722650243414928	0.037880537083692475
Híbrido 1	1000393	0.5639170029050307	0.5060848513498186
Híbrido 2	1000990	1.2675698225850536	1.1563898830397845

Tabla 3.73: Resultados F11 - 100 dimensiones, $muestras = 12$ en LS.

3.13 F12 - Generalized Penalized Function No.01

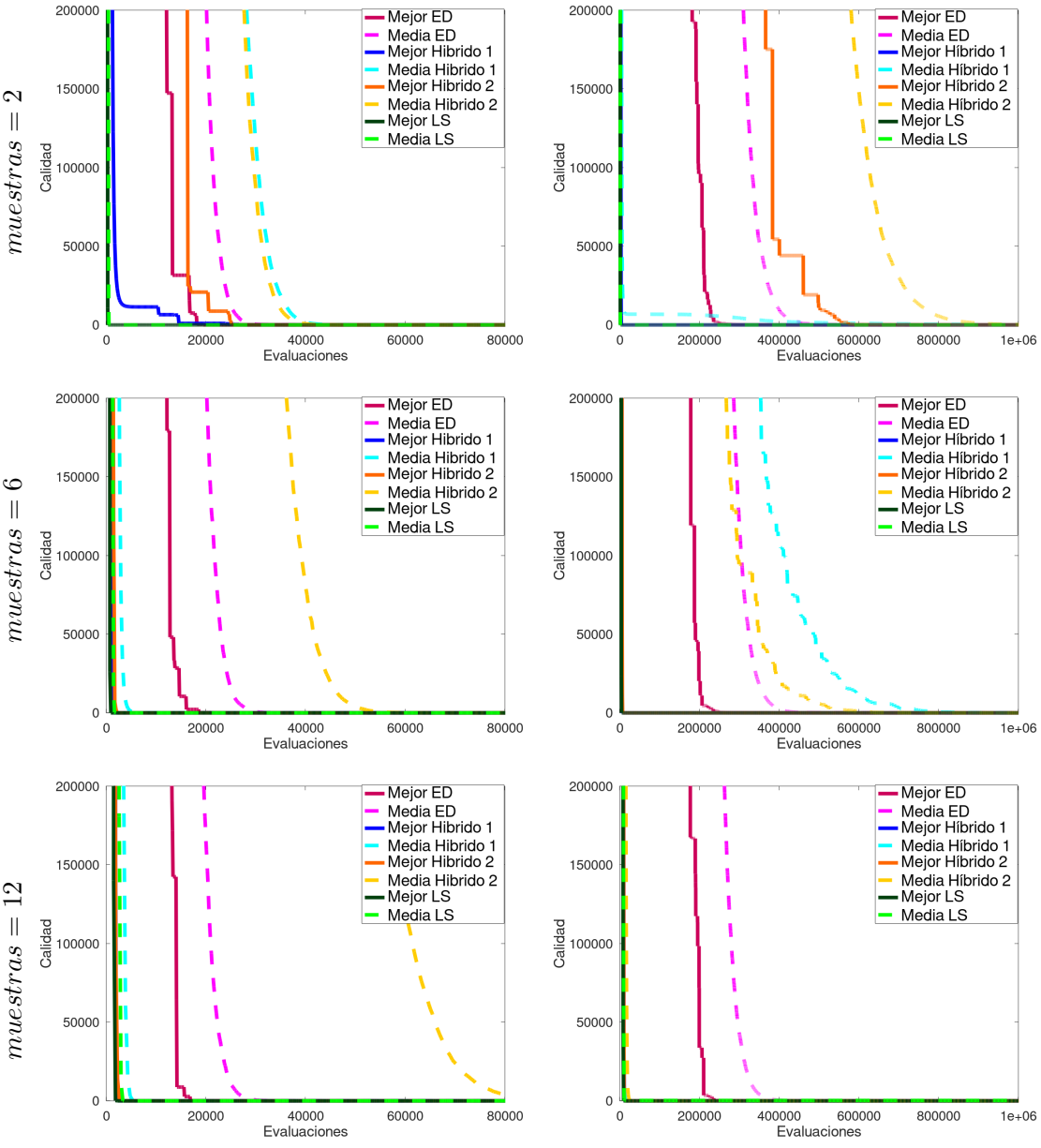
En general, todos los algoritmos presentan una desviación típica muy elevada en todos los escenarios, sobretodo en los escenarios con 100 dimensiones.

ED obtiene los mejores resultados en los escenarios con 30 dimensiones y en el caso de 100 dimensiones con 6 muestras. En el resto de casos, ED no se queda muy lejos del ganador. Por lo general, ED parece ser una opción bastante robusta teniendo en cuenta la dificultad de F12.

Aumentar el número de muestras en 30 dimensiones, parece mejorar el rendimiento de LS, pero ocurre lo contrario con las versiones híbridas.

Cuando la dimensionalidad del problema es 100, los resultados, tanto de LS como de los algoritmos híbridos, se muestran muy sensibles al específico número de muestras establecido. Con 2 muestras los algoritmos híbridos parecen tener muchos problemas y presentan una desviación típica muy elevada y, en cambio, LS obtiene los mejores resultados. Por contra, con 6 o 12 muestras establecidas, es LS el que presenta los peores resultados, además de una desviación típica muy elevada.

En el caso de 100 dimensiones con 12 muestras, las versiones híbridas obtienen los mejores resultados de entre todos los algoritmos.



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.14: Gráfica de resultados: Función F12 - Generalized Penalized Function No.01. Todas las gráficas son el promedio de 1000 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	2.7664294683168197E-15	2.570563397757653E-14
LS	300060	0.08052829981852765	0.19565821945979556
Híbrido 1	300043	9.521430997405923E-9	2.175522181676582E-8
Híbrido 2	300114	4.7489380691867E-10	5.990639388757136E-10

Tabla 3.74: Resultados F12 - 30 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	2.0541940671970618E-14	1.1567244130947095E-13
LS	300071	0.03407139727153453	0.21449693279148885
Híbrido 1	300205	4.1408313808921865E-4	4.962248257505577E-4
Híbrido 2	300169	1.2065758595568663E-5	1.5082162095006345E-5

Tabla 3.75: Resultados F12 - 30 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	2.3141813459271283E-15	1.8479966658971733E-14
LS	300140	0.012373883971675045	0.08891689927820627
Híbrido 1	300127	0.06106678440013278	0.1066998509202014
Híbrido 2	300301	0.03857554850999988	0.04078115720466737

Tabla 3.76: Resultados F12 - 30 dimensiones, $muestras = 12$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	2.2261407918622194	2.5528599734721036
LS	1000177	0.03880094232342467	0.12562053605975043
Híbrido 1	1000022	158.05651691549107	1178.2304189411584
Híbrido 2	1000084	214.15381449157985	738.2814121644468

Tabla 3.77: Resultados F12 - 100 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	1.2943682179361389	2.0157946565596765
LS	1000066	1.4788496430542106E7	1.437114266389779E8
Híbrido 1	1000495	8.901024428396905	161.4681929924179
Híbrido 2	1000210	1.9876823168579507	3.292653458927814

Tabla 3.78: Resultados F12 - 100 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.4104349563978902	1.1562772587665555
LS	1000287	235.26758187721964	7439.033776156684
Híbrido 1	1000393	0.14107017092076082	0.24740236626973955
Híbrido 2	1000990	0.32931090294117227	0.41994029564620083

Tabla 3.79: Resultados F12 - 100 dimensiones, $muestras = 12$ en LS.

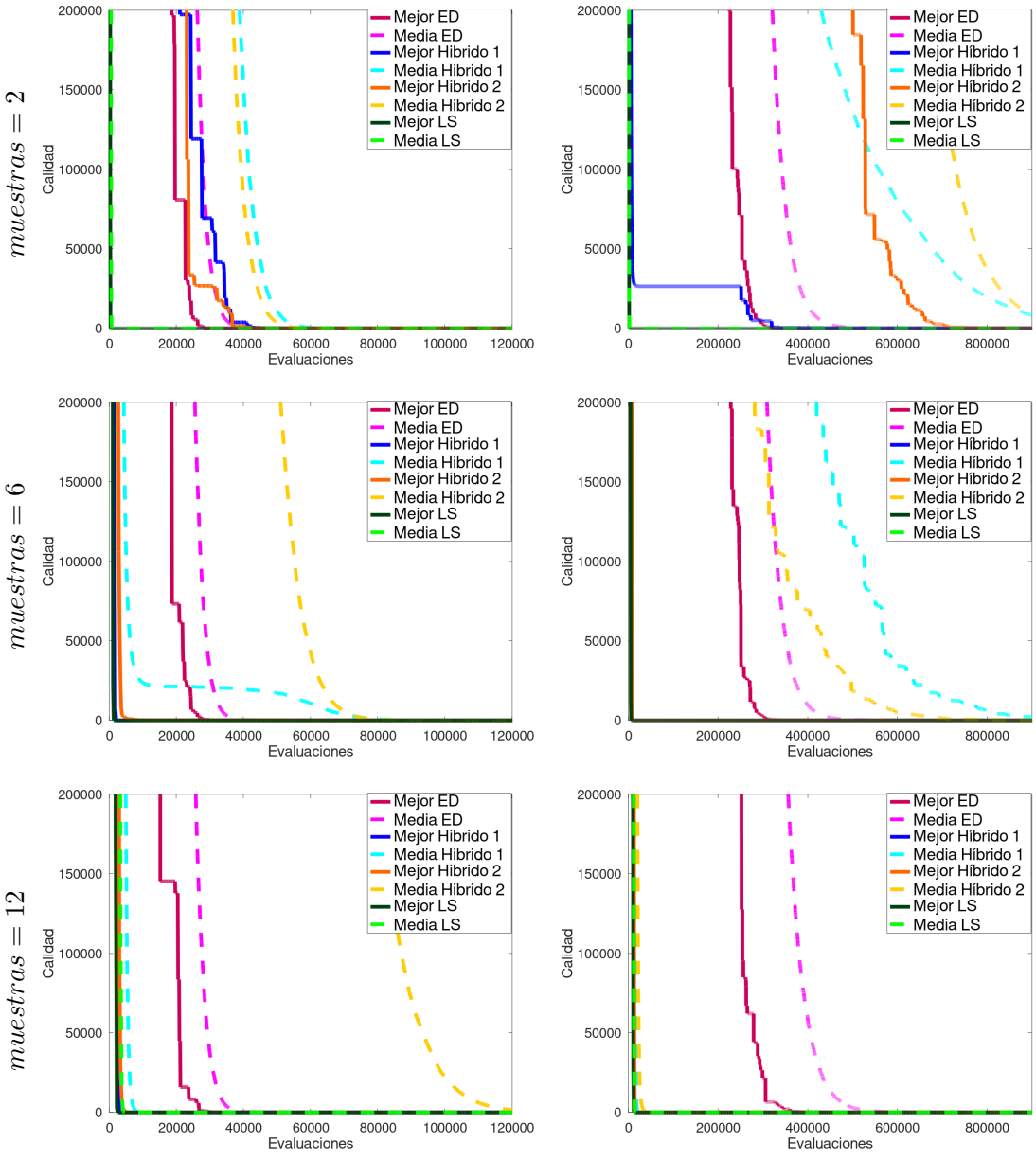
3.14 F13 - Generalized Penalized Function No.02

Al igual que en F12, ED parece ser el algoritmo más confiable en el caso de carecer de la información adecuada a la hora de ajustar la configuración de los otros algoritmos (LS e híbridos).

ED obtiene los mejores resultados en los escenarios con 30 dimensiones y en el caso de 100 dimensiones con 6 muestras. En el caso de 100 dimensiones y 12 muestras, la configuración de ED es la misma que en los otros escenarios de 100 dimensiones, pero se puede observar que la media dista de la obtenida en los otros 2 escenarios. Esto se explica observando la alta desviación típica en este caso. Lo más seguro es que alguna de las ejecuciones de ED haya tenido muy "mala suerte".

LS parece muy sensible al número especifico de muestras establecidas. En concreto, 6 muestras, tanto en 30 como 100 dimensiones, provoca un resultado pésimo con una muy elevada desviación típica. LS consigue ganar en el escenario de 100 dimensiones con 2 muestras.

Las versiones híbridas, con 30 dimensiones empeoran sus resultados al aumentar las muestras, en cambio con 100 dimensiones mejoran.



(a) $dimensiones = 30$ (b) $dimensiones = 100$

Figura 3.15: Gráfica de resultados: Función F13 - Generalized Penalized Function No.02. Todas las gráficas son el promedio de 1000 ejecuciones independientes del correspondiente algoritmo.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	1.9715455692865174E-15	1.0274541595050617E-14
LS	300060	9.339260960501198E-4	0.003065708449346487
Híbrido 1	300043	1.767521501743541E-8	2.4110929280767712E-8
Híbrido 2	300114	6.568383088281714E-10	8.668433372203179E-10

Tabla 3.80: Resultados F13 - 30 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	7.226553282438513E-16	6.4908052301597076E-15
LS	300071	1215468.83707467	3.843649897707829E7
Híbrido 1	300205	9.927185911016805E-4	8.021250253518745E-4
Híbrido 2	300169	4.656278033338043E-5	3.593692893994524E-5

Tabla 3.81: Resultados F13 - 30 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	300000	2.4527202205317606E-15	5.5592187222251296E-14
LS	300140	2.4865768091432585E-4	0.006841310589424338
Híbrido 1	300127	0.03922746052269658	0.11372445714018355
Híbrido 2	300301	0.13327151854012162	0.06735839470914479

Tabla 3.82: Resultados F13 - 30 dimensiones, $muestras = 12$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.07440320216927991	0.19258485594319347
LS	1000177	0.001582180680367266	0.0038594815514757202
Híbrido 1	1000022	3300.136140176032	19157.26998684911
Híbrido 2	1000084	2112.270237766797	7007.853273700955

Tabla 3.83: Resultados F13 - 100 dimensiones, $muestras = 2$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.07274262302382956	0.2212526862428017
LS	1000066	2.547782085625981E7	2.6492645739051077E8
Híbrido 1	1000495	771.5401395787195	17766.453366463047
Híbrido 2	1000210	18.350644638545354	180.80040627875323

Tabla 3.84: Resultados F13 - 100 dimensiones, $muestras = 6$ en LS.

Método	№ Evaluaciones	Media	Desviación típica
ED	1000000	0.5961500505203237	3.0210279794989323
LS	1000287	2.016661011283358	63.77057634379401
Híbrido 1	1000393	0.03630605944704035	0.5342113574921965
Híbrido 2	1000990	18.395474902317737	336.2909521516402

Tabla 3.85: Resultados F13 - 100 dimensiones, $muestras = 12$ en LS.

3.15 Resumen de resultados

En esta sección se mostrará un resumen comparativo de la posición obtenida en los distintos escenarios de configuración de cada uno de los cuatro algoritmos participantes en los experimentos. Esta posición se determinará sobre la base del mejor resultado medio obtenido en la última evaluación realizada.

Para realizar un resumen comparativo que tenga en cuenta los distintos escenarios de configuración es necesario de alguna manera otorgar una puntuación a cada uno de los algoritmos, para ello se ha decidido otorgar una puntuación dependiendo de la posición obtenida por cada algoritmo en los distintos escenarios de configuración (13 funciones x 6 escenarios de configuración). La posición se corresponde a la obtenida en los experimentos descritos anteriormente en este capitulo (media de 1000 ejecuciones independientes del correspondiente algoritmo en los escenarios de 30/100 dimensiones y 2/6/12 muestras.)

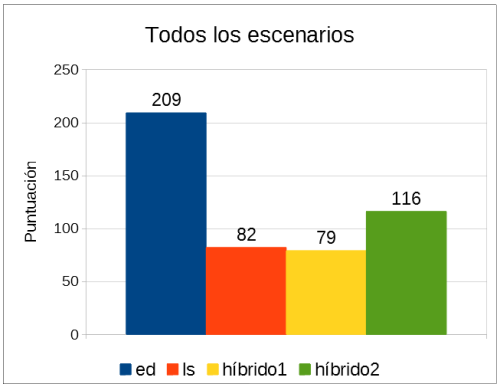
Los puntos se han otorgado de la siguiente manera:

- Alcanzar la solución -> 3 puntos
- 1ª posición -> 3 puntos
- 2ª posición -> 2 puntos
- 3ª posición -> 1 puntos
- 4ª posición -> 0 puntos

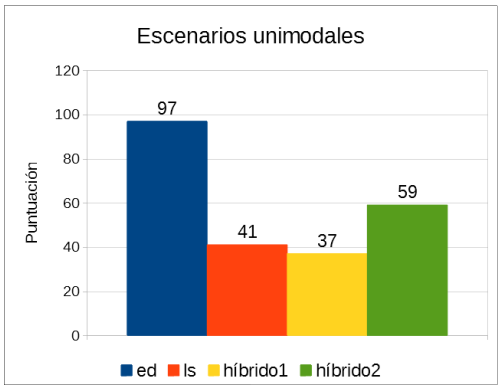
De esta forma el algoritmo que obtenga el mejor resultado final (media de 1000 ejecuciones independientes) obtendrá 3 puntos mientras que el último obtendrá 0 puntos. Una vez obtenida la puntuación de un algoritmo en uno de los 6 escenarios de configuración, esta se suma a la puntuación obtenida en los cinco escenarios restantes, obteniendo de esta manera la puntuación total de un algoritmo para una de las funciones de *benchmark*.

Es importante remarcar que dada la complejidad de los escenarios resulta difícil determinar un sistema de puntuación que muestre un resumen genérico comparativo de resultados justo. Existen muchos otros factores que podrían ser interesante tener en cuenta además de la posición como podría ser el tiempo de ejecución, descartar escenarios de configuración claramente perjudiciales etc... Es por estos motivos que este resumen de resultados no ha de entenderse como una verdad absoluta.

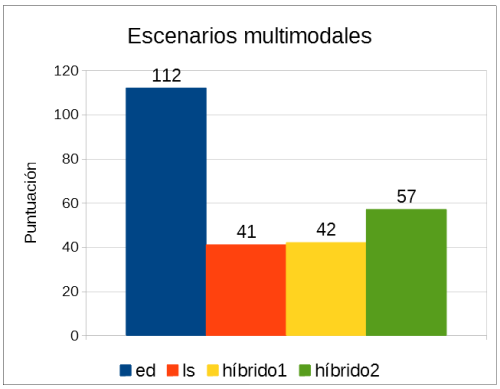
La puntuación obtenida para los distintos tipos de escenarios se puede encontrar en la figura 3.16. En la tabla 2.1 se puede observar a que tipo de escenario pertenece cada una de las funciones de *benchmark* utilizadas en los experimentos.



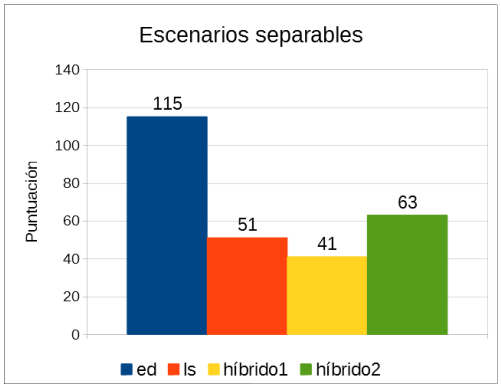
(a) Puntuación obtenida en las funciones de F01 a F13



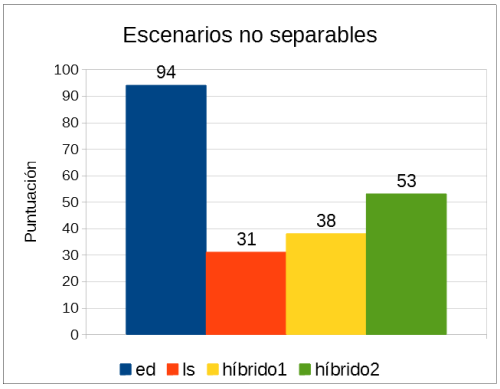
(b) Puntuación obtenida en las funciones de F01 a F04 y de F06 a F07



(c) Puntuación obtenida en las funciones F05 y de F08 a F13



(d) Puntuación obtenida en las funciones de F01 a F02, F04 y de F06 a F09



(e) Puntuación obtenida en las funciones F03, F05 y de F10 a F13

Figura 3.16: Puntuación obtenida por los distintos algoritmos en los distintos tipos de escenarios.

Capítulo 4

Proceso de ingeniería

En este capítulo se detallará la metodología aplicada para la realización de este proyecto. Así mismo también se mostrarán de forma breve unos resúmenes de la gestión del proyecto realizada, fases del proyecto (*sprints*), estimación de tiempo y coste y herramientas utilizadas.

4.1 Metodología

Este proyecto comenzó con un estudio teórico exhaustivo del dominio del problema. La finalidad de este estudio fue la de identificar necesidades así como valorar los riesgos del proyecto con el fin de poder cumplir los objetivos establecidos. Un conocimiento extenso del dominio también evita el desarrollo de soluciones destinadas al fracaso.

Después de identificar el problema y de recopilar la información necesaria, fue necesaria la elección de una metodología ágil que se adaptase a los requerimientos propios de un proyecto de investigación como es el caso. Los requisitos metodológicos para un proyecto como este son:

- Desarrollo iterativo: En muchas ocasiones los pasos a seguir en una investigación se deciden en función a los resultados obtenidos, por ello es necesario poder dividir el trabajo en pequeñas iteraciones. Esto permite obtener resultados cuanto antes para poder estudiarlos y adaptar los siguientes pasos del proyecto en consecuencia, incorporando el conocimiento adquirido

- Cambios en los requisitos: En una investigación los requisitos suelen variar con frecuencia conforme a los resultados obtenidos en los experimentos. Por tanto, una metodología adecuada será aquella que proporcione mecanismos para tratar con este problema.

- Gestión temprana de riesgos: Detectar y minimizar su impacto lo antes posible resulta clave en este tipo de proyectos. Imaginemos, por ejemplo, el impacto que puede suponer el trabajar con datos mal clasificados o detectar tarde un error en el código fuente

en mitad de la ejecución de un experimento que puede llegar a tardar hasta 1 mes en ejecutarse.

Estas restricciones y propiedades pueden ser satisfechas por una metodología de desarrollo ágil. En este caso se ha elegido *Scrum* por la flexibilidad que ofrece a la hora de adaptarlo a este proyecto. Entre las peculiaridades de este proyecto están las siguientes: no existe un cliente como tal y el equipo está formado por una única persona.

4.1.1 Adaptaciones de *Scrum* necesarias

Dadas las circunstancias de este proyecto se han definido únicamente 2 roles:

- Dueño del producto: Representa la voz del cliente, haciendo de puente entre este y el equipo. Su tarea consiste en el mantenimiento del *Product Backlog* (descripción ordenada de las tareas a realizar). En este proyecto, este rol se ha asignado a los directores del trabajo José Santos Reyes y Javier Parapar López.
- Desarrollador: Persona encargada de realizar entregas incrementales potencialmente utilizables en cada *sprint*. Esta persona es responsable de llevar a cabo todas las tareas necesarias para producir los incrementos, como análisis, diseño, desarrollo o pruebas. En resumen su función es transformar los elementos del *Product Backlog* en incrementos. Este rol ha sido asumido por el alumno.

La *Daily Scrum* es una reunión corta que tiene lugar cada día del *sprint* (bloque de tiempo para la realización de las tareas asignadas al mismo) con el fin de evaluar el progreso del proyecto. En esta reunión cada miembro del equipo trata de responder las siguientes preguntas:

- ¿Qué hiciste ayer?
- ¿Qué harás hoy?
- ¿Hay impedimentos en tú camino?

Debido a que en este proyecto el equipo de desarrollo está formado únicamente por el alumno, estas reuniones se han sustituido por reflexiones diarias sobre estas cuestiones al inicio de la jornada.

4.2 Gestión del proyecto

El proyecto partió con una implementación en Java del algoritmo ED proporcionada por los directores del mismo, así como una implementación de alguna de las funciones *benchmark* finalmente utilizadas.

En esta sección se describen los bloques de tareas (*Sprints*) en los que se han agrupado las tareas realizadas para este trabajo. Posteriormente se realiza una estimación aproximada de tiempo y coste del proyecto

4.2.1 Sprints

El proyecto se ha dividido en los siguientes *sprints*:

- ***Sprint* 0 - Estudios iniciales**: En este *sprint* inicial se realizó un estudio del dominio: búsqueda global y local, LS, ED, y configuraciones comúnmente utilizadas.
- ***Sprint* 1 - Organización, lectura y almacenamiento de datos**: Se desarrolló el código necesario para poder extraer y guardar de forma eficiente los datos de los futuros experimentos a realizar. Se le dio especial importancia a optimizar estas tareas, por ejemplo minimizando el número de lecturas/escrituras a ficheros con el fin de reducir lo máximo posible el tiempo de ejecución de los experimentos.
- ***Sprint* 2 - Visualización de datos**: Para facilitar el análisis de los datos se desarrolló un script en Octave cuya tarea consistía en la generación de gráficas de los distintos experimentos a partir de los ficheros de datos generados. Este fichero se desarrolló teniendo en mente futuras necesidades: añadir más algoritmos, nuevas funciones de *benchmark*, generar únicamente gráficos de un tipo de problema específico , etc. Por lo tanto, este fichero debía ser altamente configurable de forma rápida y sencilla.
- ***Sprint* 3 - Normalización de problemas y corrección de errores**: Se requería normalizar el rango del problema entre -1 y 1 exceptuando a la hora de evaluar la función de *fitness*. También era necesario corregir un error detectado en el *sprint* anterior a la hora de contabilizar el número de evaluaciones realizadas. Este problema se debía a una mala sincronización entre los distintos hilos de ejecución.
- ***Sprint* 4 - Implementación de LS**
- ***Sprint* 5 - Implementación de funciones de *benchmark* restantes**
- ***Sprint* 6 - Detección y corrección de errores**: Se realizaron pruebas unitarias para detectar posibles errores de implementación de las funciones de *benchmark*. También

se estudiaron los resultados de los experimentos realizados sobre la base teórica para comprobar la correcta implementación de LS y ED.

- ***Sprint* 7 - Implementación de los algoritmos híbridos**
- ***Sprint* 8 - Realización de experimentos con distintas configuraciones**: Una vez construido el entorno apropiado en los anteriores *sprints* se realizaron experimentos con distintos parámetros de configuración buscando una configuración favorable ante los 13 problemas propuestos. También se realizaron otros experimentos (algunos de ellos descartados y otros finalmente incluidos en este documento) como, por ejemplo, la prueba con otro esquema de ED.
- ***Sprint* 9 - Desarrollo de la memoria (Cap 2)**
- ***Sprint* 10 - Desarrollo de la memoria (Cap 3)**: Para la realización de este capítulo fue necesario volver a repasar el estudio inicial para poder realizar de la mejor manera un análisis exhaustivo de los resultados de los experimentos.
- ***Sprint* 11 - Desarrollo de la memoria (Cap 1,4,5)**

4.2.2 Estimación de tiempo y coste

Sin tener en cuenta los largos tiempos de ejecución de los experimentos, el trabajo ha supuesto unas 770h de trabajo efectivo aproximadamente. De media cada *sprint* ha supuesto unas 70h.

En las figuras 4.1 y 4.2 se pueden encontrar los diagramas de Gantt que representan los períodos de trabajo correspondientes al año 2019 y 2020 respectivamente.

Si se estima que el sueldo medio de un estudiante de Ingeniería Informática es de 15 euros la hora, obtenemos un coste de 11550 euros.

Figura 4.1: Diagrama de Gantt del desarrollo del proyecto, mostrando la distribución temporal de los *sprints* correspondientes al año 2019

Figura 4.2: Diagrama de Gantt del desarrollo del proyecto, mostrando la distribución temporal de los *sprints* correspondientes al año 2020

4.3 Equipo y herramientas utilizadas

- Bibliografía
- Ordenador (Windows 10 pro 64 bits) (AMD Ryzen 5 3600X)
- Java
- Octave
- Eclipse
- Git
- Taiga

Capítulo 5

Conclusiones y trabajo futuro

En este trabajo se ha implementado una solución híbrida entre ED y LS, y una solución híbrida alternativa en la cual la búsqueda local actúa de modo diferente sobre cada posición del individuo (parámetro o gen). Posteriormente se ha realizado una comparativa de calidades obtenidas, utilizando un mismo número de evaluaciones necesarias de calidad, entre las diferentes soluciones. Para ello se han utilizado 13 funciones *benchmark* ampliamente utilizadas en computación evolutiva para la comparativa de algoritmos (unimodales, multimodales, separables, no separables, ...). Por último, se ha realizado un análisis de los resultados de las soluciones híbridas frente a las soluciones puras de ED y LS, teniendo en cuenta el nivel de epístasis (interdependencia entre genes/parámetros) de los distintos problemas.

Los híbridos desarrollados se han mostrado muy dependientes de la configuración aplicada, así como de las características especificas del escenario al que se enfrentaban. Por lo general evolucionan mucho más rápido en las primeras evaluaciones que ED y consiguen ser más robustos ante los distintos escenarios a la hora de evitar estancarse en mínimos locales que LS, lo cual podría hacerlos preferibles a la hora de enfrentarse a problemas desconocidos en un tiempo corto. El mayor problema de las hibridaciones desarrolladas es que no han conseguido destacar por encima de sus dos progenitores (ED y LS), dejando a estes los mejores resultados obtenidos dependiendo de si el problema favorecía la exploración (ED) o la explotación (LS). Probablemente, la implementación/configuración escogida para los híbridos no haya sido la más adecuada y estén pesando más las desventajas de uno de los progenitores ante un problema concreto que las ventajas que pueda aportar el otro.

Como futuras líneas de investigación, sería interesante analizar el efecto sobre los híbridos de utilizar otros esquemas de ED con mayor presión selectiva, con el objetivo de que los individuos refinados con LS puedan cobrar mayor relevancia a la hora de formar las subsiguientes generaciones. También podría resultar de interés realizar un estudio más cuidadoso sobre la fuerte influencia del parámetro "número de muestras" en algunos problemas donde resulta difícil explicar el comportamiento de los híbridos al aumentarlo o disminuirlo.

Una importante dirección de trabajo a seguir podría ser la de desarrollar y refinar un mecanismo que permita al híbrido detectar en qué momento puede ser más provechoso dedicar un mayor número de evaluaciones a ED o LS. Por ejemplo, este mecanismo podría tratar de detectar cuándo un individuo se encuentra cercano a un mínimo local con el fin de evitar "malgastar" evaluaciones que se centren en realizar explotación en lugar de exploración.

Apéndices

Apéndice A

Funciones de benchmark

A.1 Apéndice. Definición de las funciones benchmark utilizadas

En este apéndice se puede encontrar la definición de las funciones de *benchmark* utilizadas en el proyecto.

Se pueden encontrar más detalles sobre las funciones utilizadas en [15][16]

f01 - Sphere Model

$$f_1(x) = \sum_{i=1}^{30} x_i^2$$

$$-100 \leq x_i \leq 100$$

$$\text{minimum at } f_1(0, \cdots, 0) = 0$$

f02 - Schwefel's Problem 2.22

$$f_2(x) = \sum_{i=1}^{30} |x_i| + \prod_{i=1}^{30} |x_i|$$

$$-10 \leq x_i \leq 10$$

$$\text{minimum at } f_2(0, \cdots, 0) = 0$$

f03 - Schwefel's Problem 1.2

$$f_3(x) = \sum_{i=1}^{30}(\sum_{j=1}^{i} x_j)^2$$

$$-100 \leq x_i \leq 100$$

$$\text{minimum at } f_3((0, \cdots, 0) = 0$$

f04 - Schwefel's Problem 2.21

$$f_4(x) = max_i\{|x_i|, 1 \leq i \leq 30\}$$

$$-100 \leq x_i \leq 100$$

$$\text{minimum at } f_4(0, \cdots, 0) = 0$$

f05 - Generalized Rosenbrock's Function

$$f_5(x) = \sum_{i=1}^{29} |100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2|$$

$$-30 \leq x_i \leq 30$$

$$\text{minimum at } f_5(1, 1, \cdots, 1) = 0$$

f06 - Step Function

$$f_6(x) = \sum_{i=1}^{30}(\lfloor x_i + 0.5 \rfloor)^2$$

$$-100 \leq x_i \leq 100$$

$$\text{minimum at } f_6(0, \cdots, 0) = 0$$

f07 - Quartic Function with Noise

$$f_7(x) = \sum_{i=1}^{30} ix_i^4 + \text{random}[0, 1)$$

$$-1.28 \leq x_i \leq 1.28$$

$$\text{minimum at } f_7(0, \cdots, 0) = 0$$

f08 - Generalized Schwefel's Problem 2.26

$$f_8(x) = \sum_{i=1}^{30}(x_i sin(\sqrt{|x_i|}))$$

$$-500 \leq x_i \leq 500$$

$$\text{minimum at } f_8(420.9687, 420.9687, \cdots, 420.9687) = -12569.5$$

f09 - Generalized Rastrigin's Function

$$f_9(x) = \sum_{i=1}^{30}(x_i^2 - 10cos(2\pi x_i) + 10)$$

$$-5.12 \leq x_i \leq 5.12$$

$$\text{minimum at } f_9(0, \cdots, 0) = 0$$

f10 - Ackley's Function

$$f_{10}(x) = -20exp(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30}x_i^2}) - exp(\frac{1}{30}\sum_{i=1}^{30}cos(2\pi x_i)) + 20 + e$$

$$-32 \leq x_i \leq 32$$

$$\text{minimum at } f_{10}(0, \cdots, 0) = 0$$

f11 - Generalized Griewank's Function

$$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{30}x_i^2 - \prod_{i=1}^{30}cos(\frac{x_i}{\sqrt{i}}) + 1$$

$$-600 \leq x_i \leq 600$$

$$\text{minimum at } f_{11}(0, \cdots, 0) = 0$$

f12, f13 - Generalized Penalized Functions

$$f_{12}(x) = \frac{\pi}{30}\left\{10\sin^2(\pi y_i) + \sum_{i=1}^{29}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} +$$

$$\sum_{i=1}^{30} u(x_i, 10, 100, 4)$$

$$-50 \leq x_i \leq 50$$

$$\text{minimum at } f_{12}(1, \cdots, 1) = 0$$

$$f_{13}(x) = 0.1\left\{\sin^2(\pi 3x_i) + \sum_{i=1}^{29}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_{30})]\right\} +$$

$$\sum_{i=1}^{30} u(x_i, 5, 100, 4)$$

$$-50 \leq x_i \leq 50$$

$$\text{minimum at } f_{13}(1, \cdots, 1) = 0$$

where:

$$u(x_i, a, k, m)\begin{cases} k(x_i - a)^m, & \text{if } x_i > a \\ 0, & \text{if } -a \leq x_i \leq a \\ k(-x_i - a)^m, & \text{if } x_i < -a \end{cases}$$

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

A.2 Apéndice. Gráficas 2D de las funciones benchmark utilizadas

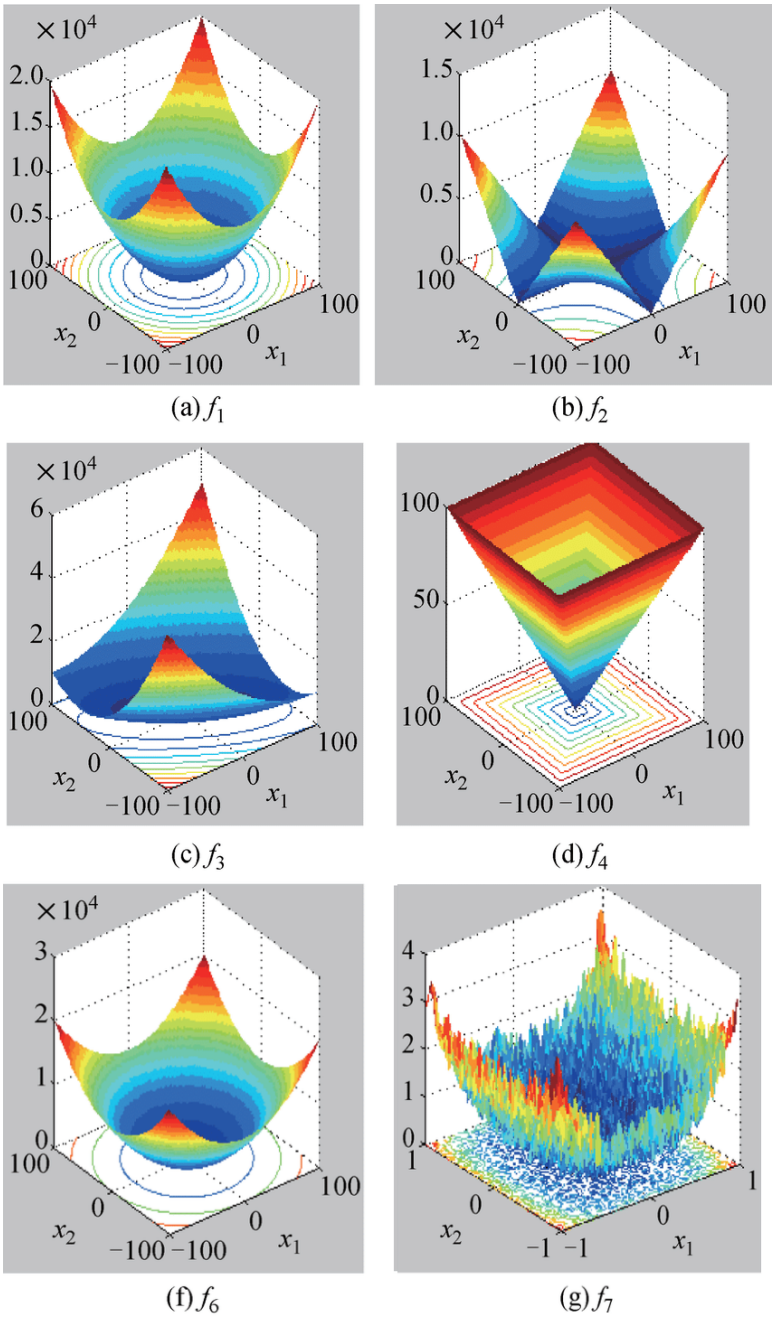


Figura A.1: Gráficas 2D de las funciones *benchmark* unimodales.

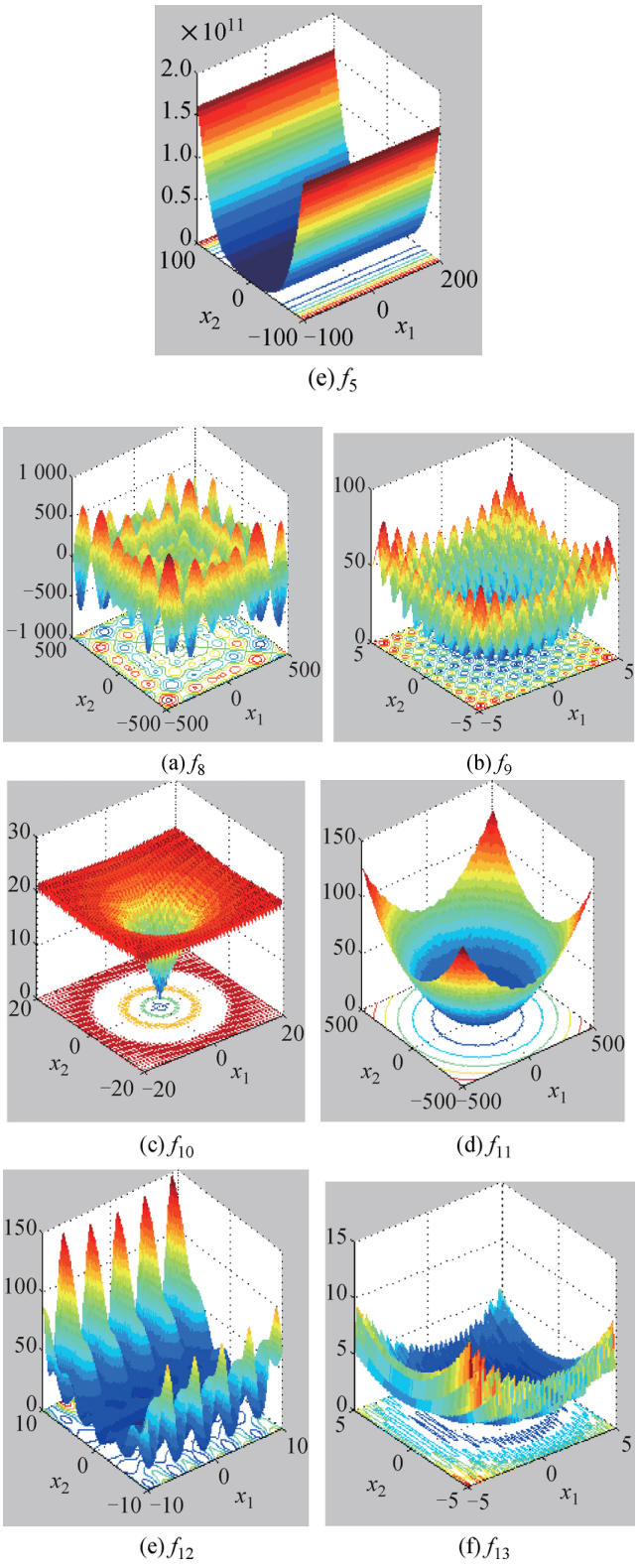


Figura A.2: Gráficas 2D de las funciones *benchmark* multimodales.

Bibliografía

[1] D. Whitley, V. Gordon, and K. Mathias, "Lamarckian evolution, the Baldwin Effect and function optimization," *Lecture Notes in Computer Science*, vol. 866, pp. 6–15, 1994.

[2] P. Moscato and C. Cotta, "A modern introduction to memetic algorithms," *Handbook of Metaheuristics. International Series in Operations Research & Management Science, Gendreau M., Potvin JY. (Eds.)*, vol. 146, pp. 141–183, 2010.

[3] J. Santos and M. Diéguez, "Differential evolution for protein structure prediction using the HP model," *Lecture Notes in Computer Science*, vol. 6686, pp. 323 –323, 2011.

[4] D. Varela and J. Santos, "A hybrid evolutionary algorithm for protein structure prediction using the Face-Centered Cubic lattice model," in *Proceedings International Conference on Neural Information Processing - ICONIP 2017, Lecture Notes in Computer Science 10634*, 2017, pp. 628–638.

[5] J. Chenlo, J. Parapar, D. Losada, and J. Santos, "Finding a needle in the blogosphere: An information fusion approach for blog distillation search," *Information Fusion*, vol. 23, p. 58–68, 2015.

[6] J. Parapar, M. Vidal, and J. Santos, "Finding the best parameter setting. Particle swarm optimization," in *Proceedings CERI 2012 - II Congreso Español de Recuperación de Información (CERI 2012*, 2012, pp. 49–60.

[7] R. Storn and K. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 23, 01 1995.

[8] S. Das and P. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[9] V. Feoktistov, *Differential evolution: in search of solutions*. NY: Springer, 2006.

[10] K. Price, R. Storn, and J. Lampinen, *Differential evolution. A practical approach to global optimization.* Springer - Natural Computing Series, 2005.

[11] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[12] W. Sun and Y.-x. Yuan, "Optimization theory and methods. Nonlinear programming," vol. 1, 01 2006.

[13] M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, and C. Burges, "Optimisation methods for ranking functions with multiple parameters," in *Proceedings of the 15th ACM international conference on Information and knowledge management*, ser. CIKM '06. New York, NY, USA: ACM, 2006, pp. 585–593. [En línea]. Disponible en: http://doi.acm.org/10.1145/1183614.1183698

[14] D. G. Luengerber, *Linear and nonlinear programming.* Addison Wesley, 1984.

[15] X. Yao, Y. Liu, K.-H. Liang, and G. Lin, "Fast evolutionary algorithms," 2003.

[16] E. Mezura-Montes, J. Velázquez-Reyes, and C. Coello, "A comparative study of differential evolution variants for global optimization," vol. 1, 01 2006, pp. 485–492.