UNIVERSIDAD NACIONAL DE LA PATAGONIA AUSTRAL

DIRECCIÓN DE POSTGRADO

Mejoras al algoritmo de Evolución Diferencial para resolver problemas de optimización con restricciones

HERNÁNDEZ, Sebastián Alejandro

Tesis para optar por el grado de Magíster en Informática y Sistemas

Director: Dr. LEGUIZAMÓN, Guillermo

Codirector: Dr. MEZURA-MONTES, Efrén

Río Gallegos, 01 de Septiembre de 2015.

Dedicado a mi familia,
por su apoyo incondicional

Agradecimientos

A mis afectos, por la constante presencia y aliento. A mis asesores, Guillermo y Efrén, por aceptar mi ritmo desordenado de trabajo a pesar de la distancia que nos separa. A la UNPA, por los fondos del proyecto 29/A317 con los que pude adquirir la licencia de MatLab.

Índice general

Capítulo 1

Introducción

La matemática es el pilar fundamental de la ciencia desde que se comenzaron a crear modelos, tratando de simular la realidad en un intento por comprender el mundo que nos rodea. Si se quiere hacer un resumen muy acotado de la historia de la matemática, entonces es posible decir que se evolucionó desde, entre otros aspectos, el concepto de número, luego la creación de funciones (en su forma más sencilla, funciones lineales) y se llegó a la obtención de métodos algebraicos de resolución. Este esquema, clásico en la educación básica y en los primeros años de estudio de cualquier carrera relacionada con las ciencias exactas, sufre un colapso al enfrentarse a modelos más complejos y más aproximados al caso real, que dependen de funciones no lineales o funciones lineales de alto grado.

Es en esta etapa de la evolución científica donde surgen los métodos numéricos, que proveen herramientas simples pero potentes como una alternativa de resolución para todo tipo de problemas. Una de las ideas más utilizadas dentro de los métodos numéricos es el de *esquema iterativo*. Es decir que se utiliza una función o mapeo que se retroalimenta con su propia salida, obteniendo en cada paso una mejora en la búsqueda de solución con respecto al paso anterior. Por mencionar un ejemplo, es muy conocido y utilizado el método de Newton-Raphson [8, 23], para la obtención de raíces de funciones explícitas. Éste opera a través de la función y su derivada, de acuerdo al siguiente esquema iterativo:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},$$

donde es necesario elegir convenientemente el valor de x_0, llamado *semilla*, para luego iterar hasta lograr la condición de convergencia elegida (obtención de solución) o finalizar la iteración por haber alcanzado la máxima cantidad de pasos a realizar. Este método es de rápida convergencia en la mayoría de los casos y simple de utilizar. A pesar de lo mencionado, si el problema a resolver involucra dos o más variables, se transforma la división de la función por su derivada en un producto de vector por inversa de matriz, eliminando la simpleza del método y su facilidad de operatoria.

Lo que sucede con el ejemplo anterior es lo que ocurre a menudo, es decir que a medida de que aumenta la cantidad de variables involucradas en un problema, éste se torna cada vez más difícil de resolver por técnicas tradicionales. Es en este punto donde aparecen los métodos heurísticos. Éstos son procedimientos simples, que proveen buenos resultados (aunque tal vez no óptimos) a problemas complicados [57], donde las principales complicaciones son la dimensionalidad, el tiempo de cómputo de resolución por métodos tradicionales y la no linealidad del problema a resolver (para problemas de variable continua). A grandes rasgos, se pueden identificar varios tipos diferentes de métodos heurísticos:

- **Constructivos**, donde se construye la solución en pasos bien definidos y cada paso es básico para generar (construir) el siguiente.
- **De descomposición**, donde se descompone el problema original en pequeños problemas más simples. La resolución de cada uno de ellos aporta un componente a la solución del problema original.
- **De reducción**, donde se centra el foco en el tipo de problema presentado con el fin de identificar características que permitan hace simplificaciones, tales como simetrías o condiciones de borde.
- **De manipulación del problema**, donde se genera un nuevo problema, similar al original, pero con menos complejidad, manteniendo las características básicas. Entre los procedimientos más comunes es posible mencionar la linealización de una función compleja, la eliminación de condiciones consideradas no vitales, entre otros.
- **De búsqueda por entorno**, donde a partir de una solución no óptima se construye una solución *más cercana* a la óptima a través de modificaciones sucesivas (iteraciones). En este grupo se encuentran las metaheurísticas.

Las técnicas metaheurísticas fueron concebidas para resolver diversos problemas matemáticos de optimización [10, 34]. Si bien existen muchas diferencias con los métodos numéricos, también hay muchos puntos en común: se basan en esquemas iterativos con reglas sencillas, utilizan semillas (generalmente aleatorias) e iteran hasta obtener la condición de convergencia o lograr el máximo de iteraciones a realizar. A diferencia de los métodos numéricos, no hay condiciones establecidas para lograr satisfactoriamente la optimización deseada. Es importante destacar que ninguna metaheurística garantiza la identificación de una solución óptima, ni permiten conocer la cercanía de la solución obtenida con respecto a la óptima. A pesar de ello, ofrecen un abanico de técnicas que pueden lidiar con problemas de diversa índole, obteniendo resultados de calidad como lo demuestra la gran cantidad de publicaciones científicas referidas a este tema.

En cualquier problema de optimización, pueden identificarse dos grandes procedimientos sobre el espacio de búsqueda: *exploración* y *explotación* [33]. El primer término se refiere al reconocimiento del espacio de soluciones potenciales, realizando una búsqueda amplia. El segundo se refiere al aprovechamiento de una región promisoria para realizar una búsqueda intensiva, es decir, se realiza una inspección minuciosa dentro de un espacio acotado. Si bien lo ideal sería utilizar un método que conjugue exploración y explotación en forma equitativa, existen otros que se concentran principalmente en la explotación. Son los denominados *buscadores locales* [35]. Generalmente presentan una baja capacidad de exploración del espacio de búsqueda, pero una gran capacidad de explotación de las soluciones potenciales dentro de un esquema iterativo retroalimentado. Esto, que puede suponer una desventaja al aplicarse como método único, es aprovechado al momento de hibridar dos métodos: uno con gran capacidad de exploración con otro que posea una gran capacidad de explotación. Al operar en forma secuencial entre exploración y explotación es posible suponer que cada región potencial de soluciones (obtenida por exploración) será analizada con detalle por el buscador local.

En esta tesis se presentan y analizan tres propuestas, todas utilizando una técnica metaheurística (*Evolución Diferencial*) y su hibridación con un buscador local (*Hill Climbing*). Las modificaciones realizadas a los algoritmos de base, permiten obtener soluciones de mejor calidad a problemas de optimización que los obtenidos en forma directa aplicando sólo Evolución Diferencial. Los algoritmos presentados son:

- Hibridación de Evolución Diferencial y *Hill Climbing* modificado, llamado EH+HC, se aplica búsqueda local en todas las iteraciones de ED.

- Modificación de ED+HC, ahora presentando un nuevo operador de *mutación* para Evolución Diferencial mientras que se mantiene la mejora por búsqueda local, denominado ED+HC2.
- Modificación de ED+HC2, al cual se le agrega un operador de *reparación* de soluciones no factibles y una nueva técnica de selección para Evolución Diferencial. También se mantiene la búsqueda local y se agrega un operador ya implementado, la selección por torneo. Esto define al algoritmo ED+HC3.

De estas tres propuestas, dos de ellas han servido como base para publicaciones: ED+HC en el *XVIII Congreso Argentino de Ciencias de la Computación* y ED+HC2 en el *2013 IEEE Congress on Evolutionary Computation*. La propuesta restante, ED+HC3, se utilizará como base para redactar un artículo que será propuesto para algún congreso de relevancia sobre Algoritmos Evolutivos o bien para alguna revista de divulgación científica.

Capítulo 2

Evolución Diferencial

2.1. Descripción

Dada una función objetivo, es decir una función que modela un problema del cual se requiere conocer un cero, un máximo o un mínimo, existen muchas técnicas de optimización que tratan de identificar uno de sus extremos. Si bien no es la única clasificación válida, es posible separar las técnicas de acuerdo a características propias de la función objetivo y del problema a resolver. Es muy común clasificar estos algoritmos iterativos como:

- Esquemas basados en derivadas.
- Esquemas de búsqueda directa (sin derivadas).

Una de las ventajas de usar esquemas de búsqueda directa, reside en la no dependencia del cálculo de derivadas en cada punto de la iteración. Los esquemas que se basan en derivadas, como *Steepest descent* y *Newton-Raphson* por citar algunos de los más conocidos, incorporan a la resolución del problema una carga importante de proceso computacional al tener que calcular derivadas y evaluarlas en los parámetros actuales, en cada iteración. Al margen de esto, a veces los óptimos se encuentran cerca de los puntos en los cuales esas derivadas (o gradientes, o jacobianos de acuerdo a la función objetivo) se anulan, con lo que el método utilizado es incapaz de resolver el problema al encontrarse con un caso de *overflow* ó tal vez *underflow*.

Dentro de los esquemas de búsqueda directa, se encuentran las metaheurísticas. En ellas pueden identificarse métodos donde se opera con un único individuo, ya sea un escalar ó un vector, que cambia con el paso de las iteraciones. Estos son esquemas denominados *single-point*. También puede operarse con un algoritmo que simule la evolución de una población, donde el proceso principal es la interacción de muchos individuos que generan nuevos individuos a través de las iteraciones. A este segundo tipo de esquemas se los conoce como *multi-point* ó como algoritmos poblacionales. Dentro de este tipo, se destacan los *Algoritmos Evolutivos* y dentro de ellos, la *Evolución Diferencial*.

Algunos de los algoritmos poblacionales más conocidos son:

- **Algoritmos genéticos**. La idea original fue propuesta por Holland en 1975 [16] y se basó en *la supervivencia del más apto*, concepto acuñado por Charles Darwin. La idea fundamental detrás de este algoritmo es que, dentro de una población, los individuos mejor preparados producirán un mayor número de descendientes. Entonces la población debe, con el paso del tiempo, converger a la uniformidad, donde todos los individuos se transformen en *superindividuos*, es decir individuos con la mejor preparación posible para sobrevivir. La implementación de este algoritmo para problemas de variable real puede hacerse mediante una codificación en base binaria puesto que

las *cadenas de cromosomas* utilizadas son secuencias de ceros y unos. La población evolucionará a través del tiempo con los procesos de cruzamiento, mutación y selección. Se considera que el algoritmo genético llegó a la convergencia cuando todos los individuos son idénticos. Al final de la evolución, debe retransformarse la cadena binaria en un vector de números decimales, en el caso de resolver un problema de variable real.

- **Nelder-Mead**. También conocido como *método simplex*, su primera publicación fue realizada en 1965 por J. Nelder y R. Mead en *Computer Journal* [38]. Es fundamental para entender este método el concepto de simplex, que es una *figura geométrica en dimensión n, de volumen no nulo, que es la envolvente conexa de* $n + 1$ *puntos.* Es decir, es un politopo de $n + 1$ vértices en un espacio de n dimensiones. Cada iteración comienza con los $n + 1$ vértices distintos del simplex, donde cada uno de ellos tiene un valor asociado (ajuste). En las sucesivas iteraciones se trata de eliminar el *peor* vértice, remplazado por un nuevo vértice que se consigue por exploración del espacio de búsqueda a través de cuatro procesos: reflexión, expansión, contracción y reducción. Este algoritmo no requiere codificación especial y se considera que se alcanzó la solución cuando el simplex encierra un volumen casi nulo, es decir que todos los vértices del simplex están muy cercanos.

- **Optimización del enjambre de partículas**. Presentado en 1995 por J. Kennedy y R. Eberhart en *Neural Networks* [24], es otro algoritmo bioinspirado que surgió para elaborar modelos de conductas sociales tales como un banco de peces o una nube de insectos, pero que después se descubrió se aplicación como optimizador. Las bases son sencillas, cada solución candidata es una *partícula* que tiene libertad de movimiento por el espacio de búsqueda. La posición y velocidad de las partículas se rigen de acuerdo a reglas matemáticas sencillas, donde se tiene en cuenta (para cada partícula) la mejor posición local encontrada hasta el momento e influye también la mejor posición global de todo el enjambre. Este movimiento es permitido hasta lograr la convergencia, donde se considera que todo el enjambre confluye en un punto del espacio de búsqueda, o sus movimientos son siempre centrados en él.

Si bien los antes mencionados tienen un buen desempeño al ejecutarse para resolver problemas, en cierta medida ED ha mostrado un desempeño superior al resto para problemas de variable real, ya que:

- No requiere codificación especial para crear y utilizar los individuos de la población. Éstos simplemente son vectores de números reales ó enteros de acuerdo a la función objetivo.

- Utiliza operadores independientes que se conjugan para hacer evolucionar la población a través de iteraciones denominadas *epochs*.

- Los operadores que utiliza son básicamente transformaciones lineales entre vectores reales.

2.2. Operadores

El algoritmo de evolución diferencial [41] sigue el esquema clásico de algoritmo poblacional, es decir que inicia su población generando n individuos (vectores) cuyas k componentes surgen a través de números aleatorios. Luego, los individuos operan entre sí, simulando un proceso de evolución sobre una población, hasta alcanzar las T iteraciones.

2.2.1. Inicialización

Para comenzar el proceso de inicialización son necesarios los parámetros:

- n, tamaño de la población inicial. Este valor es constante a lo largo de todo el proceso de evolución. Sin embargo, la salida final de ED es el mejor individuo[1] de la población final.
- k, cantidad de componentes de un individuo (vector). Una de las características de ED es poder resolver problemas sin límite de variables, por lo que este es un parámetro propio de cada función objetivo y del problema a optimizar.
- b_U y b_L, rangos dentro de los cuales se crea cada componente de los individuos de la población. Estos pueden ser valores reales o enteros. Sin embargo, ED trata a cada valor como un número de punto flotante.

Con los parámetros mencionados, se genera cada uno de los n individuos de la población. Por ejemplo, el valor inicial ($epoch = 0$) de la componente j del vector i de la población es:

$$x_{j,i,0} = \mathrm{rand}(0,1) \cdot (\mathrm{b_{j,U}} - \mathrm{b_{j,L}}) + \mathrm{b_{j,L}} \tag{2.1}$$

Luego de generada la población inicial, se indexa cada individuo con un valor entero desde 0 hasta $n-1$. Después, a través de iteraciones (*epochs*), se crean n nuevos individuos, a través de los operadores, que no son otra cosa que perturbaciones de los individuos de la población actual. Luego se seleccionan los *mejores* de acuerdo al problema a resolver, es decir que la selección depende de la función objetivo, llamada también *fitness* ó función de ajuste. Estos dos procesos de perturbación para la creación de nuevos individuos y permanencia de estos nuevos individuos en la población dependen de tres operadores clásicos de ED: Mutacion diferencial, Cruzamiento y Selección.

2.2.2. Mutación diferencial

El primer operador que se aplica sobre la población, un individuo a la vez, es la Mutación Diferencial. El proceso es sencillo, a partir de tres individuos seleccionados al azar se crea un nuevo individuo denominado *vector mutante* $\mathbf{v}$. En la posición i de la nueva población, para la *epoch* g, se genera el vector mutante de la siguiente forma:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}), \tag{2.2}$$

donde:

- $\mathbf{x}_{r0,g}$ es el *vector base*.
- $\mathbf{x}_{r1,g}$ y $\mathbf{x}_{r2,g}$ son los *vectores para diferencia*.
- F es el *factor de escala* y es un número real que controla la proporción de evolución de la población.
- Los índices $r0$, $r1$ y $r2$ se escogen de manera aleatoria, son distintos entre sí y distintos a i.

Por cada uno de los n individuos de la población, se genera un vector mutante. Luego se les aplica el siguiente proceso evolutivo, el operador de cruzamiento.

[1] desde el punto de vista del ajuste, ya sea máximo o mínimo

2.2.3. Cruzamiento

El segundo operador que interviene en ED es el Cruzamiento. En su versión original, fue definido como *Cruzamiento Uniforme*, aunque también se lo conoce como *Recombinación Discreta* ya que recombina dos vectores diferentes seleccionando, para crear el nuevo vector, una componente a la vez. Este operador genera el *vector de prueba* cruzando cada vector de la población original con el que posee el mismo índice en la población mutante, seleccionando una componente por vez de acuerdo a un parámetro aleatorio:

$$\mathbf{u}_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} \text{ si } \text{rand}_j(0,1) \leqslant Cr \text{ o } j = j_{\text{rand}} \\ x_{j,i,g} \text{ en otro caso} \end{cases}, \tag{2.3}$$

donde:

- $v_{j,i,g}$ es la componente j, del vector mutante indizado i, en la *epoch* g.
- $x_{j,i,g}$ es la componente j, del vector original indizado i, en la *epoch* g.
- Cr es la *probabilidad de cruzamiento* y es un número real que controla, en forma aleatoria, qué componentes del vector mutante se copiarán para generar el vector de prueba.
- La condición $j = j_{\text{rand}}$ se agrega para evitar que el vector de prueba sea una copia exacta del vector padre.

Luego de generar n vectores de prueba, es el momento de decidir cuáles sobrevivirán para formar la población en la siguiente *epoch*. El responsable de este proceso es el operador de selección.

2.2.4. Selección

El tercero de los operadores básicos de ED es la selección. Es el proceso que decide si, para la siguiente generación (*epoch*), permanece en la población el vector $\mathbf{x_g}$ (vector original o *padre*) ó el vector de prueba $\mathbf{v}$ (también denominado *hijo*). La decisión depende de la función objetivo y, para el caso de minimización, la regla de selección es:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} \text{ si } f(\mathbf{u}_{i,g}) < f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} \text{ en otro caso} \end{cases} \tag{2.4}$$

Luego de aplicar el operador de selección para los n individuos de la población, es el momento de empezar una nueva *epoch*. Este proceso iterativo y evolutivo continúa hasta alcanzar las t_{max} iteraciones (evolución).

2.2.5. Pseudocódigo de ED

Una de las ventajas de ED es su sencillez de implementación. En el sitio web de Storn y Price (http://www.ICSI.Berkeley.edu/∼storn/code.html) es posible encontrar implementaciones de ED para diversos lenguajes, entre los que se destacan Java, C, C++, Fortran90, entre otros.

Este pseudocódigo de la versión clásica de ED fue extraído del sitio de Storn y Price:

```
Procedure DE{
t = 0;
   Initialize Pop(t); /* of |Pop(t)| Individuals */
   Evaluate Pop(t);
   While (Not Done)
```

```
{for i = 1 to |Pop(t)| do
   {parent1, parent2, parent3} = Select_3_Parents(Pop(t));
   thisGene = random_int(|Pop(t)|);
   for k = 1 to n do /* n genes per Individual */
      if (random < p) /* p is crossover constant in [0,1] */
         Offspring(i) = parent1(i) + mu (parent2(i) - parent3(i));
      else
         Offspring(i) = Individual(i) in Pop(t);
   end /* for k */
   Evaluate(Offspring(i));
end /* for i */
Pop(t+1) = {j | Offspring(j) is_better_than Individual(j)}
{k | Individualk is_better_than Offspringk};
t = t + 1;}
```

2.3. Variantes de ED

El esquema presentado para ED es el denominado *ED Clásico*. Las diferentes modificaciones realizadas sobre el esquema original se presentan de la forma *DE/xx/yy/zz* (de acuerdo al idioma inglés) donde *xx* representa qué vector se utilizará como vector base en la mutación diferencial, *yy* es la cantidad de diferencias que se tendrán en cuenta para la mutación y *zz* es el esquema que seguirá el cruzamiento, generalmente binomial o exponencial. Por ejemplo, la versión clásica puede escribirse como *DE/rand/1/bin* [53], porque el vector base es elegido en forma aleatoria, se adiciona 1 diferencia de vectores en el operador de mutación y porque la cantidad de componentes donadas por el vector mutante se aproxima a una distribución binomial. Las siguientes son algunas de las estrategias propuestas por Storn y Price en su sitio web para la mutación diferencial, que es el operador encargado de generar diversidad poblacional:

- ***DE/best/1/exp***, $\mathbf{v}_{i,g} = \mathbf{x}_{best,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g})$
- ***DE/rand/1/exp***, $\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g})$
- ***DE/rand-to-best/1/exp***, $\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + \lambda \cdot (\mathbf{x}_{best,g} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{r0,g} - \mathbf{x}_{r1,g})$
- ***DE/best/2/exp***, $\mathbf{v}_{i,g} = \mathbf{x}_{best,g} + F \cdot (\mathbf{x}_{r0,g} + \mathbf{x}_{r1,g} - \mathbf{x}_{r2,g} - \mathbf{x}_{r3,g})$
- ***DE/rand/2/exp***, $\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F \cdot (\mathbf{x}_{r1,g} + \mathbf{x}_{r2,g} - \mathbf{x}_{r3,g} - \mathbf{x}_{r4,g})$
- ***DE/best/1/bin***, $\mathbf{v}_{i,g} = \mathbf{x}_{best,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g})$
- ***DE/rand/1/bin***, $\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g})$
- ***DE/rand-to-best/1/bin***, $\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_{best,g} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{r0,g} - \mathbf{x}_{r1,g})$
- ***DE/best/2/bin***, $\mathbf{v}_{i,g} = \mathbf{x}_{best,g} + F \cdot (\mathbf{x}_{r0,g} + \mathbf{x}_{r1,g} - \mathbf{x}_{r2,g} - \mathbf{x}_{r3,g})$
- ***DE/rand/2/bin***, $\mathbf{v}_{i,g} = \mathbf{x}_{r5,g} + F \cdot (\mathbf{x}_{r0,g} + \mathbf{x}_{r1,g} - \mathbf{x}_{r2,g} - \mathbf{x}_{r3,g})$

En la lista anterior, el vector de base puede ser *best* (aquel individuo con el mejor ajuste en cada *epoch*), *rand* (un individuo seleccionado en forma aleatoria, independiente de su valor de ajuste) o *rand-to-best* (se utiliza una combinación lineal entre la diferencia del vector en la posición i y el mejor de la población, al que se le suma el vector en la posición i), no siendo éstas las únicas opciones.

Es importante diferenciar los conceptos de *cruzamiento binomial* y *cruzamiento exponencial*. En la primera opción, las componentes del vector mutante durante la aplicación de este operador se seleccionan para formar el vector de prueba si se cumple la condición establecida en (2.3), realizando la comparación para cada una de las componentes. Por lo tanto, luego del cruzamiento, el vector de prueba se forma con componentes del vector mutante y del vector original, en forma aleatoria. En cambio, en el cruzamiento exponencial, una vez que se verificó la condición establecida para efectuar el cruzamiento en una de las variables, se realiza el intercambio de valores hasta la componente k. Es decir que, si la condición de cruzamiento se verifica en la posición j, entonces las primeras $j-1$ componentes del vector de prueba corresponden al vector original y las componentes desde la j hasta la k se extraen del vector mutante. Diversos experimentos muestran que es más efectiva la utilización del cruzamiento binomial.

No todas las versiones propuestas para la mutación diferencial son buenas para los diversos tipos de problemas a resolver con ED, sino que el rendimiento puede variar en función de la cantidad de variables del problema, los factores de escala y/o la *epoch* en la que se encuentra la evolución [32].

2.4. Conclusiones

En este capítulo se presentó un breve comentario de los algoritmos evolutivos, centrándose en el algoritmo ED y haciendo hincapié en detalles de estructura de sus operadores. También se mostraron las modificaciones realizadas por los creadores de ED, Storn y Price, sobre la estructura de la mutación diferencial y el tipo de cruzamiento a utilizar.

En esta tesis se utiliza como base al algoritmo de ED en su versión clásica, es decir *DE/rand/1/bin*, puesto que es el más utilizado y por ende el más analizado en la literatura.

Capítulo 3

Problemas Restringidos

Los algoritmos evolutivos fueron creados basándose en las premisas de la evolución darwiniana: una población sobrevive si evoluciona a través del tiempo, siendo los procesos que logran la evolución la interacción entre ellos (cruzamiento), la aparición de diversidad genética (mutación) y la supervivencia del más apto (selección). Esto se lleva a cabo codificando el problema y/o modelando la situación a través de una función f llamada *función objetivo* ó *fitness*.

Sin embargo, ninguna metaheurística en su versión original, considera la aplicación de restricciones sobre las variables de definición del problema. Esto es importante de considerar ya que para cualquier problema de aplicación es imposible que las variables puedan tomar valores sin ningún tipo de restricción. Por ejemplo, cualquier problema que involucre la creación de un compuesto donde las variables representen masas de elementos químicos, admite como solución solamente valores positivos, ya que representan cantidades a agregar.

La cuestión principal de un problema de optimización restringida es obtener $\mathbf{x}$ tal que el valor de $f(\mathbf{x})$ sea extremo (generalmente mínimo), donde f es la función objetivo y la utilización de $\mathbf{x}$ como solución del problema está sujeta a:

- $g_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, m$
- $h_j(\mathbf{x}) = 0, j = 1, 2, \ldots, p$

donde las g_i son restricciones de desigualdad y las h_j son restricciones de igualdad. La búsqueda del valor $\mathbf{x}$ óptimo se realiza dentro del *espacio de búsqueda* $\mathcal{S}$, definido por el rango de cada una de las k componentes de $\mathbf{x}$: $x_{kL} \leq x_k \leq x_{kU}$. En el caso de un problema restringido, no es posible utilizar la totalidad de elementos de $\mathcal{S}$, sino que la búsqueda debe acotarse dentro del *espacio factible* $\mathcal{F}$, de forma tal que $\mathcal{F} \subseteq \mathcal{S}$ y donde los valores de $\mathbf{x}$ cumplen con las restricciones g_i y h_j [51].

Existen diversas técnicas de manejo de restricciones para dotar a las metaheurísticas de la capacidad para resolver problemas restringidos, desde *castigar* a los individuos no factibles hasta intentar *reparar* a aquel individuo que no pertenezca a la región factible. Algunas de las dificultades que presentan la mayoría de estas técnicas son, entre otras, la correcta elección de parámetros, la estimación del tamaño del espacio factible dentro del espacio de búsqueda y la disminución de la variación poblacional al no considerar en la población a ciertos individuos no factibles pero con una composición promisoria [3, 31].

3.1. Funciones de penalidad

Las funciones de penalidad constituyen una de las técnicas más utilizadas que se aplican sobre un algoritmo evolutivo para resolver problemas con restricciones [34]. Su objetivo es

simple, fueron creadas para *castigar* a los individuos no factibles de la población. Para su aplicación es necesaria la creación de una nueva función objetivo que considere la violación a las restricciones:

$$\phi(\mathbf{x}) = f(\mathbf{x}) \pm \left[\sum_{i=1}^{m} r_i \cdot G_i + \sum_{j=1}^{p} c_j \cdot L_j \right], \tag{3.1}$$

donde $\phi(\mathbf{x})$ es la nueva función objetivo; $G_i(\mathbf{x})$ y $L_j(\mathbf{x})$ son funciones de las restricciones; r_i y c_j son los factores de penalidad. Una de las formas más utilizadas para definir las funciones de las restricciones es:

$$G_i(\mathbf{x}) = \text{máx}\left[g_i(\mathbf{x}), 0\right]^{\beta}, \qquad L_j(\mathbf{x}) = |h_j(\mathbf{x})|^{\gamma} \tag{3.2}$$

y los valores más comunes para β y γ son 1 ó 2. La ecuación (3.1) corresponde a la clase de funciones de *penalidad exterior*. En estos casos, es posible tomar de un individuo no factible y luego, por medio de los sucesivos *castigos* otorgados por la función $\phi(\mathbf{x})$ a lo largo de la evolución y luego de la ejecución del operador de selección, obtener un individuo factible. En las funciones de *penalidad interior*, es necesario contar con individuos factibles y el papel de la penalidad a lo largo de la evolución es establecer una barrera que no se puede cruzar durante el proceso de optimización. Es mucho más compleja la creación y utilización de funciones de penalidad interior que las de penalidad exterior.

3.1.1. Penalidad estática

Para aplicar penalidad estática como técnica de manejo de restricciones es necesario crear una nueva función de ajuste que responde, generalmente, al modelo de la ecuación (3.1). En este caso, los coeficientes de penalidad r_i y c_j permanecen constantes a lo largo de la evolución [17]. Otro punto a destacar es que en la mayoría de los casos se utiliza un *factor de tolerancia* ε, muy pequeño, para transformar las restricciones de igualdad en restricciones de desigualdad, de esta forma:

$$L_j(\mathbf{x}) = \begin{cases} 0, \text{ si } |h_j(\mathbf{x})| - \varepsilon \leqslant 0 \\ |h_j(\mathbf{x})|, \text{ en otro caso} \end{cases} \tag{3.3}$$

Es complicada la elección de un factor de penalidad estática, puesto que si se utilizan valores muy pequeños es posible que no se castigue adecuadamente a los individuos no factibles y éstos terminen siendo parte de la población final; sin embargo si se usa un valor muy grande como factor de penalidad, los individuos no factibles pueden ser castigados en exceso y por ende descartados de la población, logrando poca diversidad poblacional dentro de la evolución y, tal vez, logrando una *convergencia prematura*. Es decir que se logra rápida convergencia de la población a un óptimo local, lo cual no permite una exploración amplia del espacio de búsqueda y esto trae como consecuencia la obtención de una solución sub-óptima.

3.1.2. Penalidad dinámica

La técnica de manejo de restricciones conocida como penalidad dinámica es similar a la descrita anteriormente, solo que en este caso los factores de penalidad no son valores fijos sino que se modifican a medida que transcurre la evolución. Una de las formas más simples y utilizadas para crear los factores de penalidad dinámica es crear una función que dependa de la cantidad de *epochs* transcurridas, t [21]. Los siguientes son esquemas utilizados en publicaciones del tema:

- $\{r_i, c_j\} = (C \cdot t)^{\alpha}$, donde C es un valor fijo y α es un entero, generalmente 1 ó 2.

- $\{r_i, c_j\} = \left(\frac{t}{T}\right)^{\alpha}$, donde T es la cantidad total de *epochs*, es decir el tiempo de evolución, y α cumple con los mismos requisitos que en el ítem anterior.
- $\{r_i, c_j\} = k(t)$, donde $k(t)$ es un polinomio de grado 1, 2, 3 ó 4 en la mayoría de los casos, cuya variable es la cantidad de *epochs* transcurridas.

Al igual que en el esquema de penalidad estática, en este caso es compleja la elección del tipo de función que definirá a los factores de penalidad dinámica. Es posible pensar que la penalidad dinámica arroja mejores resultados que la penalidad estática, aunque en la práctica esto no siempre se confirma y se observa el mismo comportamiento descrito antes: un factor de penalidad dinámica muy bajo permite que individuos no factibles sean parte de la población final, mientras que un factor muy alto logra convergencia con individuos sub-óptimos y tal vez una convergencia prematura debido a la falta de diversidad poblacional.

3.1.3. Penalidad adaptativa

Esta es una clase de funciones donde varían los coeficientes de penalidad a medida que evoluciona la población, pero la variación de los coeficientes no depende de una elección arbitraria, sino que se considera la situación de la población actual [13]. Es frecuente confundirla con la penalidad dinámica ya que comparten la idea de un coeficiente variable, pero en este caso el coeficiente puede aumentar o disminuir en base a la factibilidad de los individuos. Para aplicarla se crea una nueva función objetivo:

$$\phi(\mathbf{x}) = f(\mathbf{x}) + \lambda(t)\left[\sum_{i=1}^{m} g_i^2(\mathbf{x}) + \sum_{j=1}^{p} |h_j(\mathbf{x})|\right], \tag{3.4}$$

donde $\lambda(t)$ es una función que depende del tiempo de evolución de la siguiente forma:

$$\lambda(t+1) = \begin{cases} (1/\beta_1)\cdot\lambda(t), & \text{si se cumple el caso A} \\ \beta_2\cdot\lambda(t), & \text{si se cumple el caso B} \\ \lambda(t), & \text{en otro caso} \end{cases}, \tag{3.5}$$

donde:

- Caso A: el mejor individuo en las últimas k iteraciones es siempre factible.
- Caso B: el mejor individuo en las últimas k iteraciones es siempre no factible.

Además, β_1 y β_2 son mayores que 1, $\beta_1 > \beta_2$. Esta configuración permite que el factor de penalidad aumente si en las últimas k iteraciones el mejor individuo es no factible, pero si es factible en las mismas últimas iteraciones el factor disminuirá. Como toda técnica que depende de parámetros, la complicación radica en definir *correctamente* los valores de k, β_1 y β_2. El escoger un valor pequeño para k puede llevar a cambios frecuentes en el factor de penalidad, con lo que se dificulta la convergencia de la población debido a la variación de los valores de la función de ajuste. Si k es un valor muy grande, puede que la población converja a un valor sub-óptimo del cual puede ser complicado salir, dependiendo del operador que genere la variación poblacional. El análisis del comportamiento de los valores de β_1 y β_2 es similar al de los coeficientes de la penalidad estática, donde valores muy pequeños permiten una amplia exploración y habilitan la convivencia de individuos factibles con no factible, pero valores muy grandes limitan la capacidad de exploración.

3.1.4. Penalidad mortal

Este esquema es el más sencillo de implementar y no depende de ningún parámetro externo: los individuos no factibles no pueden ser parte de la población [48]. El gran problema es que sólo los individuos factibles evolucionan, con una limitada diversidad genética. En el caso de contar con un espacio factible muy pequeño, la evolución se parecerá a una caminata aleatoria dentro del espacio de búsqueda, puesto que no se admiten individuos no factibles en la población.

3.2. Algoritmos de reparación

En la optimización de problemas del tipo combinatorio es sencillo *reparar* un individuo no factible por medio de modificaciones de los estados de sus componentes [27]. Si bien no hay una técnica establecida, sino que debe adaptarse de acuerdo a cada problema, esta técnica se basa en un procedimiento simple: realizar pequeñas modificaciones sobre el individuo no factible con el objetivo de satisfacer la restricción violada. Es posible que se confunda con una búsqueda local, pero la principal diferencia radica en que el individuo reparado no vuelve a la población, o puede ser parte de la población pero con una probabilidad muy baja de que este evento ocurra. En algoritmos con números reales es bastante difícil lograr un algoritmo de reparación que no otorgue cierto sesgo a la población, disminuyendo la variación poblacional, debido a restricciones muy difíciles de satisfacer. En aquellos casos en que no es muy compleja la creación de estas reglas de reparación porque es simple satisfacer las restricciones, puede ser considerada como una técnica interesante.

3.3. Separación de restricciones y objetivos

Al contrario que en las funciones de penalidad, en esta clase de manejo de restricciones se evita la combinación de ajuste y restricciones en una única función $\phi(\mathbf{x})$.

3.3.1. Co-evolución

Dado un problema de optimización, para aplicar esta técnica, se tienen en cuenta dos poblaciones que evolucionan en paralelo [2]. Una de ellas contiene sólo las restricciones, la segunda población contiene las soluciones potenciales (posiblemente inválidas) del problema. Del mismo modo que en un modelo predador-presa, la presión de selección de los miembros de una población depende del ajuste de los miembros de la otra. Es decir que existe un *fitness* inverso interactuando entre las dos poblaciones y por lo tanto, un individuo con un valor alto de ajuste en la segunda población representa una solución que satisface una gran cantidad de restricciones, del mismo modo, un individuo con un alto valor de ajuste en la primera población representa una restricción que no es satisfecha por una gran cantidad de soluciones. Una dificultad que se presenta al utilizar esta técnica es que si se tiene en cuenta la información histórica de un individuo puede llegar a estancarse su evolución, es decir que puede no progresar más, en el caso de que la mayoría de (o todas) las restricciones son difíciles de satisfacer.

3.3.2. Memoria conductista

Esta técnica, presentada por Schoenauer y Xanthakis, surge como un algoritmo para resolver problemas de optimización sin restricciones [47]. El planteo es el siguiente: se intenta satisfacer todas las restricciones, una por vez. Esta satisfacción de restricciones en

secuencia se ejecuta de la siguiente forma: cuando cierto porcentaje de la población satisface la primera restricción, se intenta satisfacer la segunda sin dejar la condición anterior. Una de las dificultades que se observan en la ejecución es que el orden de procesamiento de las restricciones influye en el resultado final, además de que es necesario que exista un orden lineal en las restricciones.

3.3.3. Superioridad de los puntos factibles

Otra de las técnicas más utilizadas de esta categoría es la *superioridad de los puntos factibles* [40], donde se crea una nueva función $\phi(\mathbf{x})$, pero ésta consta de dos definiciones: la primera para individuos factibles y la segunda para individuos no factibles. Dentro de este esquema, una de las definiciones más comunes de $\phi(\mathbf{x})$ es:

$$\phi(\mathbf{x}) = \begin{cases} f(\mathbf{x}), \text{ si } \mathbf{x} \text{ es factible} \\ f_{peor} + \sum_{i=1}^{m} G_i(\mathbf{x}) + \sum_{j=1}^{p} L_i(\mathbf{x}), \text{ en otro caso} \end{cases}, \tag{3.6}$$

donde $G_i(\mathbf{x})$ y $L_j(\mathbf{x})$ son las mismas funciones que se definieron en (3.1). Esta técnica logra buenos resultados en forma general, salvo que la región factible del espacio de búsqueda sea muy pequeña. Es posible que falle también en su objetivo si en la población inicial no existe al menos un individuo factible.

3.4. Conclusiones

En este capítulo se presentaron las técnicas de manejo de restricciones más conocidas para problemas de optimización, de manera simple. En un artículo de Coello Coello [3] y en el libro de Mezura-Montes y Coello Coello [31] es posible encontrar mayor profundidad de análisis y variedad de técnicas de manejo.

Capítulo 4

Estado del arte

Desde su presentación en el año 1996, Evolución Diferencial se mantiene como una de las principales corrientes de investigación en computación evolutiva. Esto es así puesto que es un algoritmo sencillo de implementar, con buen desempeño frente a los problemas de pequeñas y grandes dimensiones, y con una robustez remarcable frente a problemas complejos. Es por estos motivos que muchos investigadores realizan constantes modificaciones a la estructura de ED ó agregan nuevos operadores con el objetivo de mejorar el desempeño de ED, además de realizar implementaciones con base en un problema concreto de optimización. Es posible encontrar publicaciones en los últimos tiempos referentes a:

- **Hibridaciones**. Resolver un problema a través de ED operando en conjunto con un buscador local, otro algoritmo especial para generar la diversidad poblacional o sistemas especiales de selección.
- **Estimación de parámetros**. ED depende de pocos parámetros, tales como el factor de escala para la mutación y la probabilidad de cruzamiento. Sin embargo, es crucial identificar los valores óptimos para que la población logre una convergencia uniforme y no prematura a la solución.
- **Análisis de los operadores tradicionales**. Es necesario comprender cuál es el comportamiento de los operadores evolutivos, desde la inicialización de la población hasta el cruzamiento de individuos, con el fin de poder desarrollar nuevos operadores o mejorar los ya utilizados.
- **Aplicaciones**. Muchos de los especialistas que desarrollan investigación aplicada resuelven problemas complejos utilizando a ED como su herramienta principal.

4.1. Hibridaciones

- **Memetic Differential Evolution for Constrained Numerical Optimization Problems**, es un artículo del año 2013, cuyos autores son Domínguez-Isidro, Mezura-Montes y Leguizamón [9]. En él se presenta un algoritmo memético para resolver problemas de optimización restringida. La propuesta consiste un utilizar a ED como algoritmo de búsqueda global, hibridándolo con un buscador local: el método de las direcciones conjugadas de Powell. Una de las ventajas que se aprovechan de este buscador local es que la función a optimizar no necesita ser derivable y no se utilizan derivadas de ningún tipo. Es común realizar hibridaciones con buscadores locales para mejorar el desempeño de ED, sin embargo la dificultad que aún permanece es cómo realizar el manejo de restricciones. Para esta propuesta se utiliza el método ε-*constrained*. Se prueba la eficacia de esta propuesta con los 36 problemas de la sesión

especial *Single Objective Constrained Real-Parameter Optimization* del CEC2010. El algoritmo propuesto logra resultados competitivos con respecto al algoritmo ganador de la sesión.

- **An Improved $(\mu + \lambda)$-Constrained Differential Evolution for Constrained Optimization**. Este artículo del año 2013, es la presentación de una nueva hibridación de ED con un buscador local por parte de Jia, Wang, Cai y Jin [20]. La versión original de $(\mu + \lambda)$-*Constrained Differential Evolution* fue presentada por Wang y Cai en 2011, pero evidenciaba algunos inconvenientes. Con el fin de subsanarlos se dotó al nuevo algoritmo, ICDE, de varias estrategias de mutación. Es un hecho que la población puede estar en cualquiera de estos tres estados: factible, semi-factible y no factible, por lo que se aplican diferentes estrategias de manejo de restricciones para cada situación. También es importante mantener la diversidad poblacional mientras se logra la convergencia al óptimo, para ello aplican el esquema de selección de individuo jerárquico no-dominado. Los resultados obtenidos por ICDE con las funciones de la sesión especial del CEC2006 sobre optimización, muestran que no sólo resuelven los inconvenientes de la versión original del algoritmo, sino que el desempeño es bueno comparado con otros algoritmos actuales.

- **Parameter estimation for chaotic systems by hybrid differential evolution algorithm and artificial bee colony algorithm**. Este es un artículo del año 2014 cuyos autores son Li y Yin [26]. En él se plantea y resuelve el problema de la estimación de parámetros de un sistema caótico y no lineal, reformulándolo como un problema de optimización multidimensional. La resolución del mismo es a través de un algoritmo híbrido, que combina evolución diferencial (buena exploración) con el algoritmo de la colonia artificial de abejas (buena explotación). Una vez desarrollada la propuesta, realizan experimentos con los sistemas de Lorenz y Chen, utilizando el algoritmo para estimar los parámetros de los sistemas caóticos. El resultado de las simulaciones muestra que el algoritmo híbrido obtiene valores mejores o al menos comparables a los obtenidos a través de evolución diferencial, el algoritmo de la colonia artificial de abejas, la optimización del enjambre de partículas y algoritmo genético.

- **RDEL: Restart Differential Evolution algorithm with Local Search Mutation for global numerical optimization**. Es un artículo del año 2014, cuyo único autor es Mohamed [36]. La propuesta presentada es hibridar a evolución diferencial con un par de versiones de mutación con búsqueda local y un mecanismo de reinicio. Esta propuesta es capaz de resolver problemas de optimización sobre espacios continuos. El operador de mutación con búsqueda local se inspira en el algoritmo de la optimización del enjambre de partículas, basados en la posición de los individuos mejor y peor de la población. Además, el operador de mutación propuesto se aplica en conjunto con la mutación original, de acuerdo a los valores de una función lineal decreciente. Con el fin de evitar el estancamiento de la población y/o la convergencia prematura, se realiza la hibridación con el esquema de mutación del *Breeder Genetic Algorithm*. El desempeño de esta propuesta, comparado con evolución diferencial original, y versiones de ED con parámetros adaptativos, es muy bueno. Con estas modificaciones se logra mejorar el rendimiento de ED en términos de calidad de solución, eficiencia y robustez.

4.2. Estimación de parámetros

- **An Adaptive Invasion-based Model for Distributed Differential Evolution**. En este artículo de De Falco, Cioppa, Maisto, Scafuri y Tarantino [5] del año 2014

se presenta una versión nueva de evolución diferencial distribuida, caracterizado por un modelo de migración inspirado en el fenómeno conocido como invasión biológica. Este modelo adaptativo posee tres esquemas que se actualizan para establecer en forma aleatoria los parámetros de mutación y cruzamiento. Estos esquemas se relacionan con la migración y están ligados al rendimiento medido entre dos migraciones consecutivas. Para medir la eficiencia de esta implementación adaptativa de ED se comparan sus resultados contra las versiones original (parámetros fijos) y también distribuida (versión clásica de buen rendimiento) de ED. Los resultados son promisorios, puesto que esta propuesta muestra gran efectividad en términos de calidad y velocidad de convergencia en la mayoría de los problemas analizados.

- **Repairing the Crossover Rate in Adaptive Differential Evolution**. Artículo del año 2013 de Gong, Cai y Wang, donde los investigadores parten de la premisa que utilizando parámetros adaptativos se logra un buen rendimiento en ED [12]. Es por esto que analizan el comportamiento del operador de cruzamiento y llegan a la conclusión que el vector *trial* está directamente relacionado a su cadena binaria, pero no relacionado en forma directa al factor de escala. Por lo tanto proponen una técnica de reparación del factor de escala para generar una versión adaptativa de ED: el factor de escala es *reparado* de acuerdo a su correspondiente cadena binaria, es decir utilizando el promedio de componentes tomadas del mutante. Este promedio de la cadena binaria es utilizado para reemplazar el factor de escala original. Los experimentos los realizan con las 25 funciones presentadas en la competencia del CEC2005. Los resultados obtenidos son mejores, o en algunos casos comparables, a los del estado del arte.

- **An Adaptive Unified Differential Evolution Algorithm for Global Optimization**, es un artículo desarrollado por Qiang y Mitchell del año 2015. En él proponen una versión unificada y adaptativa de evolución diferencial para optimización global de objetivo único [42]. La propuesta se basa en la mutación diferencial, pero en vez de proponer variantes a este operador, presentan una única ecuación donde unifican múltiples estrategias en una sola expresión matemática. Resaltan la elegancia de utilizar una única ecuación, que además proporciona una amplia exploración del espacio generado por los operadores de mutación. Como todos los parámetros de control son auto-adaptativos se elimina la carga de elegir apropiadamente esos parámetros, mientras que aumenta la performance del algoritmo. Analizan el rendimiento de esta propuesta con trece funciones básicas, unimodales y multimodales, mostrando resultados promisorios al compararse con otras versiones de evolución diferencial.

- **Differential Evolution With Dynamic Parameters Selection for Optimization Problems**. Este es un artículo del año 2014, cuyos autores son Sarker, Elsayed y Ray, donde plantean que la capacidad de ED para resolver exitosamente problemas de optimización se debe, en gran parte, a los operadores de búsqueda y al control de parámetros [46]. Esto es en sí un problema de combinatoria que también puede ser optimizado. Por lo tanto, su propuesta consiste en una versión de ED que utiliza un mecanismo para seleccionar dinámicamente la mejor combinación de parámetros (factor de amplificación, probabilidad de cruzamiento y tamaño de la población) para un problema específico durante una evolución completa. Para medir la calidad de esta propuesta se ejecutan tres *benchmarks* de optimzación, dos de ellos con restricciones y uno sin restricciones. Los resultados muestran que esta versión de ED logra mejoras con respecto al tiempo de cómputo y también con la calidad de los resultados. Además, este mecanismo propuesto puede ser aplicado a otros algoritmos poblacionales.

- **A new differential evolution algorithm with a hybrid mutation operator and self-adapting control parameters for global optimization problems**. Artículo cuyos autores son Yi, Gao, Li y Zhou y fue publicado 2014. En él analizan los intentos de diversos investigadores para mejorar el rendimiento de ED, modificando el operador de mutación y los parámetros de control [55]. Un problema frecuente es que no es simple establecer un control de parámetros efectivo para ED. Por ello, su propuesta es dotar a la versión original de ED de una mutación híbrida y auto-adaptación de los parámetros de control. Esto se logra categorizando a la población en dos partes donde se aplican diferentes tipos de mutación, además de los parámetros auto-adaptativos. Los resultados experimentales y el análisis estadístico de esta propuesta indican que se logra una buena performance con 46 funciones de prueba.

4.3. Análisis de operadores

- **Differential Evolution with ranking-based mutation operators**, es un artículo del año 2013 donde Gong y Cai analizan el comportamiento del operador de mutación y realizan una propuesta [11]. Primero analizan el estado del arte sobre artículos donde la mutación diferencial es el núcleo del trabajo, destacando trabajos de Price, Das *et al* y Zhang y Sanderson, entre otros. Luego realizan su propuesta: mejorar la habilidad de exploración de ED modificando el operador de mutación. Esta modificación está inspirada en el proceso evolutivo natural, donde los buenos especímenes contienen buena información genética y es más probable que sean elegidos para propagar descendencia. Esto se realiza clasificando a los individuos de acuerdo a un ranking de calidad y luego asignando una probabilidad de selección. A partir de ese proceso, el vector base y el terminal de la diferencia se escogen de acuerdo al ranking, el inicial de la diferencia no puesto que sino se vería muy limitada la variación poblacional.

- **A study of two penalty-parameterless constraint handling techniques in the framework of MOEAD**, fue publicado en 2103 por Jan y Khanum. En ese artículo analizan el efecto de modificar los coeficientes de penalidad cuando el manejo de restricciones se realiza a través de funciones de penalidad en los algoritmos evolutivos multiobjetivo basados en descomposición (MOEAD, por su sigla en inglés) [19]. A partir de ese análisis implementan, como métodos de manejo de restricciones, las versiones extendidas de *Stochastic ranking* y *constraint-domination principle* con el fin de evitar la elección de coeficientes de penalización.

- **Effects of Population Initialization on Differential Evolution for large scale optimization**. Este artículo del año 2014, cuyos autores son Kazimipour, Li y Qin, es un estudio sobre las diferentes configuraciones que se pueden aplicar sobre el operador de inicialización [22]. El primer análisis que realizan es sobre los estudios anteriores, por parte de otros autores, sobre la calibración de parámetros de ED. Detectan varios problemas, como por ejemplo que la calibración se realizó con problemas de pocas dimensiones. Además, los experimentos realizados fueron hechos utilizando parámetros arbitrarios, sin tener en cuenta cuál es la mejor configuración de parámetros para cada problema. Es por ello que su investigación se centró en un análisis sistemático para obtener los mejores parámetros al momento de optimizar con ED un problema de gran dimensión. Además, comparan los parámetros óptimos para un problema de baja y alta dimensión. Como segundo gran tópico, comparan los distintos tipos de inicialización de población para los parámetros más comunes y la mejor configuración de parámetros.

- **Adaptive Parameter Controlling Non-dominated Ranking Differential Evolution for Multi-objective Optimization of Electromagnetic Problems**. En este artículo del año 2014, los investigadores Baater, Jeong y Koh proponen un control de parámetros adaptativo cuando se aplica el ranking no-dominado en ED, para obtener el diseño óptimo en problemas electromagnéticos [1]. Los parámetros variables, como el factor de escala en la mutación diferencial y la probabilidad de cruzamiento, son generados basándose en la información de los parámetros anteriores que lograron un buen desempeño y el número de individuos en el frente de Pareto en la epoch actual. El algoritmo presentado combina las ventajas de exploración y explotación de ED con los mecanismos del ranking no-dominado y la clasificación de distancia por hacinamiento.

- **Investigation of Self-adaptive Differential Evolution on the CEC-2013 Real-Parameter Single-Objective Optimization Testbed**, es un artículo presentado por Qin, Li, Pan y Xia en el año 2013. El algoritmo analizado es SaDE, sigla en inglés de Evolución Diferencial Auto-Adaptativa [43]. Esta es una versión de ED que adapta gradualmente la estrategia utilizada para generar el vector trial y además aprende de la performance de las múltiples estrategias no utilizadas y su elección de parámetros asociadas. El análisis se basa en la resolución del benchmark del CEC2013 Real-parameter Single Objective. Para medir la sensibilidad de los parámetros utilizan tests con hipótesis estadísticas avanzadas, tratando de detectar la mejor configuración paramétrica.

4.4. Aplicaciones

- **Performance Comparison of Differential Evolution, Particle Swarm Optimization and Genetic Algorithm in the Design of Circularly Polarized Microstrip Antennas**. En 2014, Deb, Roy y Gupta presentan esta publicación. En ella comparan el rendimiento de varias técnicas de optimización evolutiva cuando son aplicadas al problema del diseño óptimo de antenas *microstrip* polarizadas circularmente [6]. Las técnicas analizadas son evolución diferencial, optimización del enjambre de partículas y algoritmos genéticos. La optimización se realiza para obtener la adaptación de impedancia en un rango de frecuencia predefinida, la polarización circular y de alta ganancia a una frecuencia central. Además utilizan una función de costo difuso para asegurar un resultado adecuado en cada algoritmo de optimización. Los parámetros de antena obtenidos como resultado final de los algoritmos se fabricaron y se midieron sus rendimientos.

- **Controller parameters tuning of differential evolution algorithm and its application to load frequency control of multi-source power system**. Este artículo de Mohanty, Panda y Hota del año 2014 es en primera medida un análisis sobre el rendimiento de diferentes estrategias de mutación diferencial y luego ejecutan varias alternativas de parámetros [37]. El problema que intentan resolver es cómo optimizar el control de frecuencia de carga de sistemas de potencia de múltiples unidades, con diferentes fuentes de generación de energía como termal, hidroeléctrica y centrales de gas. El control de parámetros lo realizan utilizando ED y comienzan el estudio considerando una única fuente de energía. Luego plantean y resuelven el problema de la multi carga.

- **Adaptive MIMO Neural Network Model Optimized by Differential Evolution Algorithm for Manipulator Kinematic System Identification**, es un artículo de Son y Anh del año 2104 en el que los autores utilizan una red neuronal

adaptativa de múltiples entradas y múltiples salidas (*MIMO*, por su sigla en inglés) para modelar e identificar la cinemática directa de un manipulador robótico que admite movimientos en 3 dimensiones [49]. Las características no lineales del sistema de cinemática del manipulador robótico se modelan a través de un modelo de red neuronal *MIMO* adaptativa basado en el algoritmo de evolución diferencial. Luego se utiliza ED para generar de manera óptima los pesos neuronales apropiados, con el fin de caracterizar perfectamente las características no lineales de la cinemática directa del manipulador robótico. Los resultados muestran que el modelo de red neuronal *MIMO* adaptativa propuesto, entrenado por el algoritmo de evolución diferencial, para la identificación de la cinemática directa de un robot manipulador en 3 dimensiones se comporta mejor que si fuera entrenado por *back-propagation*.

- **Asteroids Exploration Trajectory Optimal Design with Differential Evolution Based on Mixed Coding**. En este artículo del año 2015, cuyos autores son Wang, Song, Dai, Peng y Zheng, se considera el problema de la exploración de asteroides dentro del espacio profundo [54]. Las trayectorias de vuelo espacial tienen que cumplir muchos requisitos: requieren mucho tiempo, tienen muchas limitaciones de ingeniería, tienen un gran número de objetivos, y además poseen una serie de soluciones factibles. Entonces, cómo obtener el vuelo óptimo es el núcleo de la exploración espacial. Los autores proponen un nuevo método para diseñar la trayectoria óptima para la exploración de asteroides utilizando ED con codificación mixta. Una de las consideraciones planteadas, es que la secuencia celestial y la secuencia de tiempo se codifican juntos dentro de los cromosomas de ED y por lo tanto se optimizan en forma simultánea. Una vez ejecutado este algoritmo, los resultados muestran que tanto la eficiencia computacional y el rendimiento del algoritmo son superiores a lo que se aplica en la actualidad.

- **An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning**, es un artículo del año 2015, donde Zhang y Duan formulan el problema de la planificación de una ruta global para los vehículos aéreos no tripulados como un problema de optimización restringida [58]. La resolución del mismo es a través de una implementación mejorada de ED, con lo que logran generar una ruta factible óptima. Unas de las condiciones de la ruta de vuelo es que debe tener una corta duración y mantenerse a baja altitud. Además, deben considerarse múltiples restricciones físicas, tales como los ángulos máximos de giro, las inclinaciones máximas de ascenso y descenso, las áreas prohibidas de vuelo y las condiciones del mapa de vuelo. Luego de ejecutar el algoritmo de ED, comparan los resultados con otros seis algoritmos para optimización restringida. Los resultados numéricos obtenidos muestran una buena performance en términos de calidad de solución, robustez y el manejo de restricciones.

4.5. Conclusiones

En este capítulo se presentó una revisión del estado del arte sobre las distintas líneas de investigación que se han publicado recientemente sobre Evolución Diferencial. La lista es mucho más amplia, puesto que una gran cantidad de problemas de ingeniería se resuelven por medio de esta metaheurística. Es por este motivo que se siguen analizando diferentes propuestas de mejora a ED.

Capítulo 5

Propuesta I - Algoritmo ED+HC

El algoritmo ED+HC fue presentado en el *XVIII Congreso Argentino de Ciencias de la Computación*, realizado en la ciudad de Bahía Blanca en Octubre del año 2012. Surge como el resultado de la hibridación de ED con un buscador local [14]. Para medir su eficiencia se contrastaron los resultados obtenidos al optimizar el suite de funciones de la sesión especial del CEC2010 con los valores obtenidos por los ganadores de esa sesión, Takahama y Sakai, resultando superior el desempeño de la nueva propuesta.

5.1. Introducción

El algoritmo ED posee la capacidad de resolver problemas complejos de optimización en forma simple [41]. Esto es así porque es capaz de hacer exploración y explotación del espacio de búsqueda a lo largo de todo el proceso evolutivo. En cualquier algoritmo evolutivo, darle prioridad a la exploración permite escapar de los óptimos locales y/o identificar zonas promisorias en la búsqueda del deseado óptimo global, aunque no se asegura encontrarlo. Sin embargo, priorizar la explotación permite identificar valores cada vez mejores dentro de una región promisoria. Lo ideal es conjugar una buena exploración con una adecuada explotación. Esto se intenta a través de la hibridación de ED con una versión modificada de HC, generando el algoritmo ED+HC.

5.2. Buscador local

El concepto de *algoritmo de búsqueda local* para resolver problemas de optimización surge alrededor de 1960, con la publicación del artículo *A method for solving traveling-salesman problems* en *Operations Research* [4]. La propuesta es simple: dado un problema de optimización y una solución potencial del mismo, llamada $\tilde{\mathbf{x}}$, al introducir pequeños cambios en el estado de $\tilde{\mathbf{x}}$ se genera $\overline{\mathbf{x}}$, llamado vecino de $\tilde{\mathbf{x}}$. Si se cumple que $\overline{\mathbf{x}}$ proporciona un mejor ajuste al problema, medido por la función de ajuste, entonces se descarta $\tilde{\mathbf{x}}$ y se mantiene $\overline{\mathbf{x}}$ como mejor solución potencial. Este esquema es fácil de implementar en problemas con variables discretas, tales como el problema del viajante, el problema de las 8 reinas ó el problema del coloreado de grafos, y ofrece una forma sencilla de mejorar cualquier estado actual a partir del análisis de los estados de los vecinos, en forma iterativa.

Extender el concepto de búsqueda local y poder utilizar el mismo esquema que soluciona problemas discretos con problemas continuos es atractivo, porque de esta forma no es necesario depender de derivadas para analizar crecimientos/decrecimientos de la función a optimizar ni conocerla geométricamente. Es así que surgen métodos de búsqueda local co-

mo *Hill Climbing*[1] [45], Nelder-Mead [38], *Pattern Search*[2] [18], *Iterated Local Search*[3] [28], entre otros. Sin embargo, la sencillez de implementación y ejecución contrasta con otros aspectos, como la imposibilidad de escapar de mínimos locales (amplia explotación pero exploración muy restringida), la tendencia a estancarse en valles de la función a optimizar (dependencia de los parámetros intrínsecos de explotación) y la lentitud de convergencia al intentar resolver un problema con una gran cantidad de variables (baja adaptación a la escalabilidad). Es decir que son algoritmos potentes para identificar mínimos locales, pero no se garantiza la obtención de un mínimo global dentro del espacio de búsqueda, ni siquiera se asegura una amplia exploración del espacio de búsqueda.

Algunas metaheurísticas logran resolver el inconveniente de la baja capacidad de exploración, dotando al algoritmo general de un módulo de exploración aleatoria. Este es el caso de *Simulated Annealing*[4] [25] y *Random Walk*[5] [39] que, al estancarse en un mínimo local o en un valle, buscan en forma aleatoria una nueva posición para explotar.

5.3. *Hill Climbing* - HC

La estructura general iterativa de este buscador local [44], adaptado para optimizar un problema de variable real de n dimensiones, es:

1. Seleccionar el elemento actual, es decir un vector de n componentes denominado $\mathbf{x}$, y calcular su ajuste por medio de la función f, es decir calcular el valor de $f(\mathbf{x})$.

2. Duplicar $2n$ veces al vector $\mathbf{x}$, con el fin de formar una familia de vectores. A cada integrante de la familia se le perturbará sólo una componente, n veces en forma positiva y n veces en forma negativa, con una perturbación adaptativa de la siguiente manera:

   - $\mathbf{x}_{i+}(i) = \mathbf{x}(i) + \dfrac{\mathbf{x}(i)}{100 + 20j}$,
   - $\mathbf{x}_{i-}(i) = \mathbf{x}(i) - \dfrac{\mathbf{x}(i)}{100 + 20j}$,

   con $j = 0, 1, 2, \ldots$.

3. Calcular el ajuste de la familia de vectores recientemente creada, a fin de determinar en qué dirección y con qué tipo de incremento (positivo o negativo) se obtiene un mejor ajuste.

4. Al ajuste del vector $\mathbf{x}_j$, mejor vector de la familia, se lo compara con el ajuste del elemento actual. Si $f(\mathbf{x}_j) < f(\mathbf{x})$, entonces el nuevo elemento actual es $\mathbf{x}_j$; sino puede optarse entre modificar el grano de búsqueda, aumentando a j en 1, o bien terminar la exploración por *Hill Climbing*.

En cada iteración se conserva la información sobre qué variable fue perturbada para generar el nuevo elemento actual. De esta forma, si en la creación de una nueva familia perturbada se escoge nuevamente en forma seguida esa variable, la perturbación será reducida con el fin de ajustar el grano de búsqueda.

Este algoritmo finaliza cuando no es posible perturbar el estado actual con el fin de conseguir un mejor ajuste ó se llega a la cantidad máxima de exploraciones.

[1] Trepado de colinas
[2] Búsqueda por patrones
[3] Búsqueda local iterada
[4] Recocido Simulado
[5] Caminata Aleatoria

5.4. Versión modificada - HCMod

HC es un algoritmo de búsqueda local ampliamente analizado en la literatura y utilizado en diversos experimentos [30,50,56]. Sin embargo, si se lo quiere utilizar como buscador local hibridándolo dentro de un método de optimización global, se observa que consume muchos recursos. HC necesita, al menos, $2n$ evaluaciones de función para conseguir un vector que mejore el ajuste del elemento actual, de dimensión n. En una ejecución con recursos acotados, por ejemplo al tratar de optimizar una función de 30 dimensiones con una capacidad máxima de 6×10^5 evaluaciones de función, cada mejora al estado actual supone el uso mínimo de 60 evaluaciones de función. Si la cantidad máxima de exploraciones es m, entonces el costo de las evaluaciones de función por búsqueda local con HC será $2nm$. Por este motivo se diseñó una modificación a la versión original de HC, denominada HCMod. El esquema iterativo es:

1. Seleccionar el elemento actual, es decir un vector de n componentes denominado $\mathbf{x}$, y calcular su ajuste por medio de la función f, $f(\mathbf{x})$.
2. Establecer los valores de los parámetros de exploración: $k = 0$ y $pot = 0$.
3. Seleccionar en forma aleatoria v variables de las n posibles. A partir de la selección se crea el vector $\tilde{\mathbf{x}}$, perturbando v componentes de $\mathbf{x}$, de la siguiente manera:
   - $\tilde{\mathbf{x}}(v_i) = \mathbf{x}(v_i) + (-1)^{pot} \dfrac{\mathbf{x}(v_i)}{100 + 20k}$,

   para ciertos valores seleccionados de i. Luego de cada perturbación, *pot* se incrementa en 1, para perturbar en forma positiva y negativa alternando variables.
4. Calcular el ajuste de $\tilde{\mathbf{x}}$ por medio de f. Si $f(\tilde{\mathbf{x}}) < f(\mathbf{x})$, entonces el elemento actual será $\tilde{\mathbf{x}}$; sino se modificará el grano de búsqueda haciendo que k se incremente en 1. Esta exploración se repite h veces[6], siempre con las mismas v variables.
5. Repetir desde el paso 2 hasta el 4, j veces.

De esta manera, con HCMod la cantidad de evaluaciones de función será hj, pudiendo ajustar la cantidad v de variables que serán perturbadas en forma simultánea sin afectar la cantidad de evaluaciones de función necesaria para una mejora del elemento actual.

Ejemplo 1. *Se desea optimizar la función* Sphere *de 7 dimensiones:*

$$F(x) = \sum_{i=1}^{7} x_i^2, \tag{5.1}$$

donde el estado actual es:

$$\mathbf{x} = [-0{,}932498; 1{,}79403; -2{,}36173; 1{,}30868; -0{,}150002; 1{,}11524; 0{,}332306], \tag{5.2}$$

y su valor de ajuste es $f(\mathbf{x}) = 12{,}7551$. *Al efectuar una búsqueda local con HC, se establece el parámetro que controlará el incremento inicia en* $j = 0$. *Utilizando la variable 1 se generan los vectores:*

- $\mathbf{x}_{1+} = [\mathbf{-0{,}941823}; 1{,}79403; -2{,}36173; 1{,}30868; -0{,}150002; 1{,}11524; 0{,}332306]$
- $\mathbf{x}_{1-} = [\mathbf{-0{,}923173}; 1{,}79403; -2{,}36173; 1{,}30868; -0{,}150002; 1{,}11524; 0{,}332306]$

Con la variable 2, se generan:

[6] ajustable por un parámetro del algoritmo, de acuerdo a lo necesario

- $\mathbf{x}_{2+} = [-0{,}932498; \mathbf{1{,}81197}; -2{,}36173; 1{,}30868; -0{,}150002; 1{,}11524; 0{,}332306]$
- $\mathbf{x}_{2-} = [-0{,}932498; \mathbf{1{,}77609}; -2{,}36173; 1{,}30868; -0{,}150002; 1{,}11524; 0{,}332306]$

Repitiendo este proceso para las 5 variables restantes, se obtiene la familia de 14 vectores perturbados. Evaluando la familia con la función de ajuste, se detecta que el mejor ajuste se obtiene al perturbar en forma negativa a la tercera variable:

- $\mathbf{x}_{3-} = [-0{,}932498; 1{,}79403; \mathbf{-2{,}33811}; 1{,}30868; -0{,}150002; 1{,}11524; 0{,}332306]$,

donde $f(\mathbf{x}_{3-}) = 12{,}6441$. *El costo para obtener esta mejora con HC fue de 14 evaluaciones funcionales. Sin embargo, para optimizar con una exploración de HCMod el estado actual, (5.2), se establecen los parámetros* $pot = 0$, $k = 0$, $v = 4$, *se genera* $\tilde{\mathbf{x}} = \mathbf{x}$, *se reordenan las posiciones de los índices del vector:*

$$IndexPos = [5; 3; 1; 2; 0; 4; 6]$$

y se aplican las perturbaciones:

- $\tilde{\mathbf{x}}(5) = \mathbf{x}(5) + (-1)^0 \dfrac{\mathbf{x}(5)}{100}$,
- $\tilde{\mathbf{x}}(3) = \mathbf{x}(3) + (-1)^1 \dfrac{\mathbf{x}(3)}{100}$,
- $\tilde{\mathbf{x}}(1) = \mathbf{x}(1) + (-1)^2 \dfrac{\mathbf{x}(1)}{100}$,
- $\tilde{\mathbf{x}}(2) = \mathbf{x}(2) + (-1)^3 \dfrac{\mathbf{x}(2)}{100}$,

generando el vector:

- $\tilde{\mathbf{x}} = [\mathbf{-0{,}941823}; \mathbf{1{,}77609}; \mathbf{-2{,}33811}; 1{,}30868; \mathbf{-0{,}151502}; 1{,}11524; 0{,}332306]$

cuya evaluación de ajuste es $f(\tilde{\mathbf{X}}) = 12{,}5980$, *lograda con una única evaluación de función.*

Nota. *No siempre una evaluación de función de HCMod supera a las* $2n$ *evaluaciones de la familia de vectores perturbados creados por HC, esto depende del estado actual y de la función a optimizar. En general, HCMod obtiene mejores resultados que HC con pocas evaluaciones de función, aunque a medida que la cantidad de evaluaciones aumenta, este resultado se revierte. Esto es así puesto que HC, siempre que sea posible, conseguirá una mejora del estado actual al evaluar todas las posibilidades de perturbación. HCMod, en cambio, depende de una selección aleatoria de variables a perturbar.*

5.5. Algoritmo ED+HC

Al momento de hibridar ED con HCMod en forma controlada[7], es posible implementar varios esquemas sobre la aplicación de la búsqueda local:

- Con respecto al individuo:
  - Aplicar HCMod sólo a uno de los individuos de la población, seleccionado en forma aleatoria.
  - Aplicar HCMod al mejor individuo de la población.
- Con respecto al tiempo de aplicación:

[7] con respecto a la cantidad de evaluaciones de función

- Aplicar HCMod en cada *epoch* de la evolución.
- Aplicar HCMod cada cierta cantidad de *epochs*.

La combinación de cada uno de estos esquemas tiene influencias muy diversas sobre la implementación del algoritmo y sobre el resultado final. Aplicar el buscador local a un individuo escogido al azar no sigue la idea detrás de la hibridación, que es explotar en forma intensiva una región promisoria (en este caso, la región es un entorno del individuo seleccionado, aunque éste sea el peor de la población); pero aplicar búsqueda local sobre el mejor individuo es una opción más atractiva ya que la explotación se realiza sobre el individuo que se destaca en la población, tratando de mejorar aún más su valor de ajuste. Si se quiere aplicar búsqueda local en cada *epoch*, debe considerarse el costo computacional, ya que es más *económico* aplicarlo cada cierta cantidad de *epochs*. Pero, retomando nuevamente la idea principal de la hibridación, aplicar en todas las *epochs* logra una buena explotación del entorno del mejor individuo. Es bastante probable que el mejor individuo de la población en la *epoch* k lo sea nuevamente en la *epoch* $k+1$, más aún cuando la población se torna homogénea y se está cerca de la finalización del tiempo de evolución. Esto permite al buscador local explotar en forma intensa la región alrededor del mejor individuo, inclusive en más de una *epoch*. Con base en lo expuesto, se aplicó búsqueda local al mejor individuo de cada *epoch* a lo largo de todas las *epochs* de la evolución. Esta hibridación se denominó ED+HC.

5.6. Experimentos y resultados

El algoritmo ED+HC, hibridación de ED con HCMod, presenta las siguientes características al momento de ser ejecutado:

- **Inicialización de la población**: aleatoria. Se utilizó un generador de números pseudoaleatorios que siguen una distribución uniforme y con una transformación lineal se obtuvieron valores dentro del rango permitido en cada variable.
- **Mutación**: clásica. Para generar el vector mutante se utilizó el esquema

$$\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g})$$

- **Cruzamiento**: uniforme. También conocido como recombinación discreta ya que recombina dos vectores diferentes seleccionando, para crear el nuevo vector, una componente a la vez.
- **Selección**: clásica. Dados dos individuos a comparar, permanece en la población aquel que tenga mejor valor de ajuste.
- **Hibridación**: con HCMod. Se aplicó el buscador local al mejor individuo obtenido en cada *epoch*.
- **Manejo de restricciones**: penalidad estática. Se utiliza la nueva función de ajuste

$$\phi(\mathbf{x}) = f(\mathbf{x}) + c_p \left(\sum_{i=1}^{m} G_i + \sum_{j=1}^{p} L_j \right),$$

  con $c_p = 150$. Las restricciones de desigualdad, $h_j(\mathbf{x})$, fueron convertidas en restricciones de igualdad utilizando una tolerancia $\varepsilon = 1 \times 10^{-4}$, como se plantea en la ecuación 3.3.

- **Parámetros del algoritmo para 10D**:

- Tamaño de la población: 41 individuos.
- Tiempo de evolución: 4000 *epochs*.
- Probabilidad de cruzamiento: 0.6, valor sugerido por los autores de ED.
- Factor de escala en mutación: 0.6, valor sugerido por los autores de ED.
- Coeficiente de penalidad: 50, para las restricciones de desigualdad.
- Exploraciones por HCMod: 3
- Variables exploradas por HCMod en cada *epoch*: 2

▪ **Parámetros del algoritmo para 30D**:

- Tamaño de la población: 55 individuos.
- Tiempo de evolución: 7000 *epochs*.
- Probabilidad de cruzamiento: 0.6, valor sugerido por los autores de ED.
- Factor de escala en mutación: 0.6, valor sugerido por los autores de ED.
- Coeficiente de penalidad: 150, para las restricciones de desigualdad.
- Exploraciones por HCMod: 3
- Variables exploradas por HCMod en cada *epoch*: 6

Para comprobar la eficiencia del algoritmo ED+HC se utilizó el suite de funciones del *Single Objective Constrained Real-Parameter Optimization* del CEC2010 [29], con 10D y 30D. Todas las ejecuciones de esta propuesta fueron ejecutadas bajo MatLab 7.9 x64, con una HP EliteBook (Intel Core2Duo P8600, 2.40GHz, 4Gb RAM). Los resultados obtenidos se comparan con el algoritmo ganador de la sesión especial, εDEag, presentado por Takahama y Sakai [52]. A fin de equiparar condiciones de ejecución con el algoritmo de comparación, se consideró un máximo de 6×10^5 evaluaciones de función para 30D y 2×10^5 para 10D. En las tablas 5.1 y 5.2 se presentan los resultados para 10D, mientras que en las tablas 5.3 y 5.4 se muestran los resultados para 30D. Para ambas dimensiones el experimento consistió en 25 ejecuciones independientes del algoritmo ED+HC.

A modo de resumen, el mejor desempeño en los diversos tópicos fue obtenido por:

▪ Para 10D:

- Mejor: ED+HC (2), εDEag (5), Empate (11)
- Mediana: ED+HC (3), εDEag (9), Empate (6)
- Peor: ED+HC (3), εDEag (11), Empate (4)
- Promedio: ED+HC (1), εDEag (14), Empate (3)
- Desviación: ED+HC (5), εDEag (2), Empate (11)

▪ Para 30D:

- Mejor: ED+HC (13), εDEag (4), Empate (1)
- Mediana: ED+HC (11), εDEag (6), Empate (1)
- Peor: ED+HC (9), εDEag (9), Empate (0)
- Promedio: ED+HC (9), εDEag (9), Empate (0)
- Desviación: ED+HC (7), εDEag (11), Empate (0)

	F01		**F02**	
	ED+HC	**εDEag**	**ED+HC**	**εDEag**
Mejor	-7,473104E-01	-7,473104E-01	-1,524016E+00	-2,277702E+00
Mediana	-7,473104E-01	-7,473104E-01	-5,998230E-01	-2,269502E+00
Peor	-7,284826E-01	-7,405572E-01	1,061240E+00	-2,174499E+00
Promedio	-7,448930E-01	-7,470102E-01	-4,705034E-01	-2,258870E+00
Desviación	5,552960E-03	1,323339E-03	6,091665E-01	2,389779E-02
	F03		**F04**	
Mejor	0,000000E+00	0,000000E+00	-9,999999E-06	-9,992345E-06
Mediana	8,879924E+00	0,000000E+00	-9,999999E-06	-9,977276E-06
Peor	8,879924E+00	0,000000E+00	9,892771E-01	-9,282295E-06
Promedio	8,524378E+00	0,000000E+00	6,669847E-02	-9,918452E-06
Desviación	1,775912E+00	0,000000E+00	2,352037E-01	1,546730E-07

Tabla 5.1: Resultados de los experimentos para 10D con las funciones F01 a F04.

5.7. Conclusiones

Se presentó el algoritmo ED+HC, que surge al hibridar la versión clásica de ED con la modificación del algoritmo Hill Climbing, HCMod. La modificación dada al buscador local consiste en aplicar selección aleatoria de variables a modificar, perturbar todas al mismo tiempo y con una perturbación aplicada en tres grados diferentes, tratando de ajustarse al grano de búsqueda. La aplicación de la búsqueda local se realiza sobre el mejor individuo en cada *epoch* de la evolución. Al momento de comparar resultados se utilizan los obtenidos por Takahama y Sakai, creadores del algoritmo εDEag. Para 10D se observa que ED+HC es superior 14 veces y se logran 35 empates, sobre un total de 90 comparaciones; mientras que para 30D la diferencia es notoria a favor del nuevo algoritmo, 49 veces es superior y 2 se obtienen empates. Como los resultados obtenidos con 30D son prometedores, se intentará modificar otros parámetros de ED+HC o incluso la creación de nuevos operadores con el fin de mejorar los resultados obtenidos en estos experimentos.

	F05		**F06**	
	ED+HC	**εDEag**	**ED+HC**	**εDEag**
Mejor	-2,438797E+02	-4,8361060E+02	-5,785851E+02	-5,786581E+02
Mediana	-5,799827E+01	-4,8361060E+02	-4,262194E+02	-5,7865332E+02
Peor	5,158370E+01	-4,8361060E+02	3,890567E+02	-5,7864480E+02
Promedio	-8,472041E+01	-4,8361060E+02	-3,231163E+02	-5,7865280E+02
Desviación	8,388243E+01	3,8903500E-13	2,838851E+02	3,6271690E-03
	F07		**F08**	
Mejor	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00
Mediana	0,000000E+00	0,000000E+00	1,094154E+01	1,094154E+01
Peor	0,000000E+00	0,000000E+00	1,094154E+01	1,537535E+01
Promedio	0,000000E+00	0,000000E+00	9,161397E+00	6,727528E+00
Desviación	0,000000E+00	0,000000E+00	4,082066E+00	5,560648E+00
	F09		**F10**	
Mejor	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00
Mediana	0,000000E+00	0,000000E+00	4,172589E+01	0,000000E+00
Peor	1,152187E+02	0,000000E+00	2,352812E+02	0,000000E+00
Promedio	4,608748E+00	0,000000E+00	5,155337E+01	0,000000E+00
Desviación	2,304374E+01	0,000000E+00	4,869726E+01	0,000000E+00
	F11		**F12**	
Mejor	-1,522713E-03	-1,522713E-03	-5,700899E+02	-5,700899E+02
Mediana	-1,522713E-03	-1,522713E-03	-1,992458E-01	-4,231332E+02
Peor	-1,522713E-03	-1,522713E-03	-1,992458E-01	-1,989129E-01
Promedio	-1,522713E-03	-1,522713E-03	-8,623932E+01	-3,367349E+02
Desviación	3,725363E-12	6,3410350E-11	1,517830E+02	1,782166E+02
	F13		**F14**	
Mejor	-6,842937E+01	-6,842937E+01	0,000000E+00	0,000000E+00—
Mediana	-6,842937E+01	-6,842936E+01	0,000000E+00	0,000000E+00
Peor	-6,227640E+01	-6,842936E+01	0,000000E+00	0,000000E+00
Promedio	-6,749903E+01	-6,842936E+01	0,000000E+00	0,000000E+00
Desviación	1,645533E+00	1,025960E-06	0,000000E+00	0,000000E+00
	F15		**F16**	
Mejor	0,000000E+00	0,000000E+00	6,219470E-01	0,000000E+00
Mediana	3,673239E+00	0,000000E+00	1,047007E+00	2,819841E-01
Peor	4,497445E+00	4,497445E+00	1,109355E+00	1,018265E+00
Promedio	2,677700E+00	1,798978E-01	1,027937E+00	3,702054E-01
Desviación	1,712098E+00	8,813156E-01	9,315798E-02	3,710479E-01
	F17		**F18**	
Mejor	8,859727E-03	1,463180E-17	0,000000E+00	3,7314390E-20
Mediana	2,110528E-01	5,653326E-03	0,000000E+00	4,0979090E-19
Peor	1,669083E+01	7,301765E-01	4,606481E-19	9,2270270E-18
Promedio	1,031391E+00	1,249561E-01	3,134641E-20	9,6787650E-19
Desviación	3,286647E+00	1,937197E-01	1,080473E-19	1,8112340E-18

Tabla 5.2: Resultados de los experimentos para 10D con las funciones F05 a F18.

	F01		**F02**	
	ED+HC	**εDEag**	**ED+HC**	**εDEag**
Mejor	-8,199345E-01	-8,218255E-01	-1,961395E+00	-2,169248E+00
Mediana	-8,199343E-01	-8,206172E-01	-1,752225E+00	-2,152145E+00
Peor	-8,127290E-01	-8,195466E-01	-1,080479E+00	-2,117096E+00
Promedio	-8,194010E-01	-8,208687E-01	-1,683329E+00	-2,151424E+00
Desviación	1,663286E-03	7,103893E-04	2,786315E-01	1,197582E-02
	F03		**F04**	
Mejor	2,867347E+01	2,867347E+01	3,528574E-04	4,698111E-03
Mediana	2,867347E+01	2,867347E+01	1,641278E-03	6,947614E-03
Peor	2,867347E+01	3,278014E+01	3,836657E-03	1,777889E-02
Promedio	2,867347E+01	2,883785E+01	1,529283E-03	8,162973E-03
Desviación	1,347989E-06	8,047159E-01	8,342196E-04	3,067785E-03
	F05		**F06**	
Mejor	-4,771455E+02	-4,531307E+02	-5,304233E+02	-5,285750E+02
Mediana	-4,746565E+02	-4,500404E+02	-5,296760E+02	-5,280407E+02
Peor	-4,665849E+02	-4,421590E+02	-5,275613E+02	-5,264539E+02
Promedio	-4,744888E+02	-4,495460E+02	-5,294831E+02	-5,279068E+02
Desviación	3,132289E+00	2,899105E+00	6,397257E-01	4,748378E-01
	F07		**F08**	
Mejor	2,470121E-29	1,147112E-15	2,470121E-29	2,518693E-14
Mediana	8,212328E-26	2,114429E-15	6,092315E-26	6,511508E-14
Peor	1,261938E-24	5,481915E-15	9,180261E+01	2,578112E-13
Promedio	1,853336E-25	2,603632E-15	3,672104E+00	7,831464E-14
Desviación	2,904451E-25	1,233430E-15	1,836052E+01	4,855177E-14
	F09		**F10**	
Mejor	2,370624E-20	2,770665E-16	3,130908E+01	3,252002E+01
Mediana	6,920619E+01	1,124608E-08	3,130909E+01	3,328903E+01
Peor	1,617975E+02	1,052759E+02	2,650643E+02	3,463243E+01
Promedio	6,193226E+01	1,072140E+01	4,802656E+01	3,326175E+01
Desviación	4,094189E+01	2,821923E+01	5,830148E+01	4,545577E-01
	F11		**F12**	
Mejor	-3,923384E-04	-3,268462E-04	-1,992635E-01	-1,991453E-01
Mediana	-3,922639E-04	-2,843296E-04	-1,992634E-01	5,337125E+02
Peor	-3,917380E-04	-2,236338E-04	1,181352E+02	5,461723E+02
Promedio	-3,922238E-04	-2,863882E-04	-1,460144E+01	3,562330E+02
Desviación	1,408836E-07	2,707605e-05	1,010505E+02	2,889253E+02
	F13		**F14**	
Mejor	-6,706095E+01	-6,642473E+01	0,000000E+00	5,0158630E-14
Mediana	-6,116401E+01	-6,531507E+01	2,131502E-27	1,3593060E-13
Peor	-5,288011E+01	-6,429690E+01	2,749435E+02	2,9235130E-12
Promedio	-6,019908E+01	-6,535310E+01	1,099774E+01	3,0894070E-13
Desviación	3,879552E+00	5,733005E-01	5,498871E+01	5,6084090E-13

Tabla 5.3: Resultados de los experimentos para 30D con las función F01 a F14.

	F15		**F16**	
	ED+HC	**εDEag**	**ED+HC**	**εDEag**
Mejor	2,160323E+01	2,160345E+01	1,131750E+00	0,000000E+00
Mediana	2,160326E+01	2,160375E+01	1,204322E+00	0,000000E+00
Peor	2,160361E+01	2,160403E+01	1,284807E+00	5,421011E-20
Promedio	2,160329E+01	2,160376E+01	1,206492E+00	2,168404E-21
Desviación	8,447100E-05	1,104834E-04	4,461097E-02	1,062297E-20
	F17		**F18**	
Mejor	8,237259E+00	2,165719E-01	2,595158E-02	1,226054E+00
Mediana	2,855598E+01	5,315949E+00	5,837677E+00	2,679497E+01
Peor	7,755800E+01	1,889064E+01	5,419900E+01	7,375363E+02
Promedio	3,266024E+01	6,326487E+00	7,769467E+00	8,754569E+01
Desviación	1,591151E+01	4,986691E+00	1,028637E+01	1,664753E+02

Tabla 5.4: Resultados de los experimentos para 30D con las función F15 a F18.

Capítulo 6

Propuesta II - Algoritmo ED+HC2

El algoritmo ED+HC2 fue presentado en el *2013 IEEE Congress on Evolutionary Computation (CEC)*, realizado en Cancún (México) en Junio de 2013 [15]. Surge como la mejora de ED+HC [14] al agregar una nueva mutación diferencial aplicada en etapas. Es decir que ED+HC2 es un algoritmo híbrido (se aplica búsqueda local sobre el mejor individuo de cada *epoch*) con dos mutaciones diferenciales (la nueva propuesta que se centra en la factibilidad de los individuos para generar el vector mutante y la versión clásica propuesta por Storn y Price [41]) que se aplican según la *epoch* en la que se encuentre la evolución.

6.1. Introducción

La gran cantidad de publicaciones referidas al tema permiten suponer que es efectivo el generar algoritmos híbridos basados en ED. Muchos de ellos utilizan la versión original de ED como algoritmo de base, es decir que los operadores de mutación, cruzamiento y selección son los descritos según la clasificación *DE/1/rand/bin*. Si bien la ejecución de la propuesta ED+HC mostró resultados de calidad, fue posible modificar los operadores con el fin de crear un nuevo algoritmo híbrido capaz de resolver los problemas de optimización planteados y obtener resultados de mayor calidad.

6.2. Mutación tradicional

La mutación diferencial es uno de los operadores de ED cuyo aporte es la variación genética de la población, junto con el cruzamiento. Mientras que éste hace foco en la capacidad de sobrevivir de un individuo, la mutación centra su atención en generar diversidad genética al crear nuevos individuos basados en tres de ellos, ya pertenecientes a la población:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}),$$

escogidos al azar. Es importante prestar atención al término que aporta la diversidad en la fórmula anterior, compuesto por dos factores, uno de ellos se denomina *factor de escala* F y el otro es una diferencia de vectores $\Delta\mathbf{x}_{r1,r2} = (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g})$.

El factor de escala se utiliza, en forma general, como un valor constante dentro del intervalo $(0, 1)$. Sin embargo el límite superior se establece en forma empírica, sólo por una cuestión de velocidad de convergencia. Hay trabajos que muestran que se logra convergencia para $F > 1$, aunque es un proceso lento que involucra mucho tiempo de evolución [41]. Si $F = 1$ entonces se reduce a la mitad la capacidad de crear nuevos individuos mutantes. El operador de mutación incrementa la diversidad poblacional al generar nuevos individuos

que competirán entre sí para permanecer en la población, pero el operador de selección tiende a reducir esta diversidad, acentuando esto en las etapas de convergencia de la población. Para evitar la convergencia prematura es necesario que F tenga un valor tal que sea posible la generación de nuevos individuos lo suficientemente diversos. De acuerdo a los autores de ED, si se utiliza un valor fijo para el factor de escala, $F = 0{,}6$ permite la interacción entre mutación y selección de forma exitosa.

La diferencia de vectores que utiliza la versión *DE/1/rand/bin*, $\Delta\mathbf{x}_{r1,r2}$, genera diversidad poblacional de acuerdo a la cantidad de individuos presentes en la población [41,53]. Si la población es de n individuos, y se escoge uno de ellos como *base*, entonces es posible generar $(n-1)(n-2)$ diferencias no nulas de vectores. En el lado izquierdo de la figura 6.1 se muestra una población de 5 individuos y en el lado derecho se muestran las 12 posibilidades para generar un vector mutante una vez que fue escogido como vector base a $\mathbf{x}_3$, el factor de escala utilizado es 0,6.

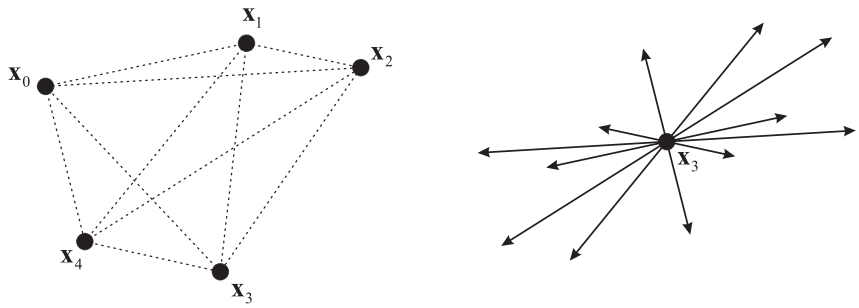

Figura 6.1: Población de 5 individuos y los posibles mutantes.

6.3. Mutación selectiva

La nueva propuesta de mutación diferencial mantiene el esquema de selección aleatoria de los tres individuos que participarán en la generación del mutante, sin embargo es necesario conocer cuáles son el ajuste y la factibilidad de cada uno de ellos. Al momento de seleccionar cuál será utilizado como base, se tendrán en cuenta las siguientes reglas:

- Si los tres individuos son factibles, será elegido como base aquel que tenga menor valor de *fitness*.
- En el caso de haber uno solo factible, automáticamente será elegido como base, independientemente de que otro presente menor valor de *fitness*.
- Si sólo hay uno no factible, será elegido como base aquel de los dos factibles con el menor *fitness*.
- Si los tres individuos son no factibles, será elegido como base aquel que presente menor valor en la suma total de violaciones.

Luego de seleccionar al individuo base, se sigue la misma forma ya utilizada:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r_A,g} + F \cdot (\mathbf{x}_{r_B,g} - \mathbf{x}_{r_C,g}),$$

donde la selección de los subíndices B y C depende de valores aleatorios entre los dos individuos no elegidos como base. En la figura 6.2 se muestra un esquema de la propuesta de mutación.

La idea detrás de este esquema de mutación es priorizar la pertenencia del mutante al espacio factible o sino utilizar como base a aquel que presente menor grado de violación

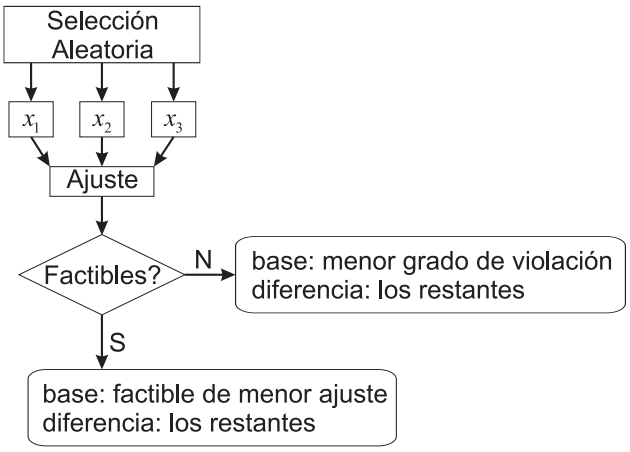

Figura 6.2: Propuesta de nueva mutación para ED+HC2.

i	$\mathbf{x}_i$	$\phi(\mathbf{x}_i)$	$g_1(\mathbf{x}_i)$	$g_2(\mathbf{x}_i)$
0	[1,1837991; 1,6724613]	−0,2710810	0,198507	0
1	[1,2036380; 0,6897186]	0,7590258	0	0
2	[2,0328157; 2,6507228]	1,4816170	7,15867	0
3	[0,2883790; 1,9758602]	−1,8926977	0	0,711620
4	[0,7953025; 0,5991685]	0,0333376	0	0,204697
5	[3,1452641; 1,7479086]	8,1447777	8,94787	0
6	[0,2378696; 0,6231376]	−0,5665556	0	0,762130
7	[2,5504460; 3,6446693]	2,8601056	15,7883	0
8	[0,8678400; 2,0158088]	−1,2626624	0,816631	0,132159
9	[1,1674396; 0,9833731]	0,3795421	0	0

Tabla 6.1: Población luego de k *epochs*.

de restricciones, a fin de lograr una población totalmente factible o que tienda a ello con el paso de las *epochs*. Existe el riesgo de una convergencia prematura, ya que la población puede estancarse en algún óptimo local debido a la eliminación de variación poblacional. Por este motivo es que es bueno combinar esta nueva mutación, que utiliza una explotación intensiva, junto con la original de ED, cuyo esquema permite la interacción entre exploración y explotación.

Ejemplo 2. *Se desea optimizar la función de dos dimensiones*

$$f(\mathbf{x}) = x_1^2 - x_2,$$

en el espacio $-5 \leq x_i \leq 5$ *para* $i = 1, 2$*, sujeta a las restricciones*

$$g_1(\mathbf{x}) = x_1^2 + x_2^2 - 4 \leq 0, \qquad g_2(\mathbf{x}) = -x_1 + 1leq0.$$

Luego de k epochs*, se tiene la población de 10 vectores que se muestra en la tabla 6.1, donde además figuran los valores de* fitness *y de las restricciones* g_1 *y* g_2*. En la tabla 6.2 se explicitan los tres vectores seleccionados en forma aleatoria para generar los vectores mutantes, los vectores base y la diferencia de vectores para la mutación tradicional y la mutación selectiva. Es de notar que los vectores mutantes en los índices* 0*,* 3*,* 5*,* 6*,* 7 *y* 9 *son los mismos independientes del tipo de mutación aplicada, ya que el primer vector seleccionado será usado como base en ambos casos.*

Luego de utilizar, en ambas mutaciones, el factor de escala $F = 0{,}6$ *se obtienen dos nuevas poblaciones mutantes que serán comparadas con la población original, lo que se*

		Mutación Tradicional		Mutación Selectiva	
Orden	Aleatorios	Base	Diferencia	Base	Diferencia
0	$10, 4, 5$	$\mathbf{x}_{10}$	$\mathbf{x}_4 - \mathbf{x}_5$	$\mathbf{x}_{10}$	$\mathbf{x}_4 - \mathbf{x}_5$
1	$8, 4, 5$	$\mathbf{x}_8$	$\mathbf{x}_4 - \mathbf{x}_5$	$\mathbf{x}_5$	$\mathbf{x}_8 - \mathbf{x}_4$
2	$5, 2, 1$	$\mathbf{x}_5$	$\mathbf{x}_2 - \mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_5 - \mathbf{x}_1$
3	$7, 8, 3$	$\mathbf{x}_7$	$\mathbf{x}_8 - \mathbf{x}_3$	$\mathbf{x}_7$	$\mathbf{x}_8 - \mathbf{x}_3$
4	$4, 1, 8$	$\mathbf{x}_4$	$\mathbf{x}_1 - \mathbf{x}_8$	$\mathbf{x}_1$	$\mathbf{x}_4 - \mathbf{x}_8$
5	$10, 4, 8$	$\mathbf{x}_{10}$	$\mathbf{x}_4 - \mathbf{x}_8$	$\mathbf{x}_{10}$	$\mathbf{x}_4 - \mathbf{x}_8$
6	$9, 3, 8$	$\mathbf{x}_9$	$\mathbf{x}_3 - \mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_3 - \mathbf{x}_8$
7	$1, 7, 3$	$\mathbf{x}_1$	$\mathbf{x}_7 - \mathbf{x}_3$	$\mathbf{x}_1$	$\mathbf{x}_7 - \mathbf{x}_3$
8	$3, 5, 4$	$\mathbf{x}_3$	$\mathbf{x}_5 - \mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_3 - \mathbf{x}_4$
9	$1, 8, 4$	$\mathbf{x}_1$	$\mathbf{x}_8 - \mathbf{x}_4$	$\mathbf{x}_1$	$\mathbf{x}_8 - \mathbf{x}_4$

Tabla 6.2: Sorteo de índices y vector base y diferencia de vectores, para ambas mutaciones.

i	$\mathbf{x}_i$	$\phi(\mathbf{x}_i)$	$g_1(\mathbf{x}_i)$	$g_2(\mathbf{x}_i)$
0	[0,8632855; 1,8093881]	−1,0641261	0,0191475	0,1367144
1	[2,2462919; 4,4706843]	0,5751432	21,03284	0
2	[0,8072058; 0,0095228]	0,6420584	0	0,1927941
3	[0,5484478; 1,2195055]	−0,9187104	0	0,4515521
4	[−0,5316090; 0,7925354]	−0,5099272	0	1,5316090
5	[−0,1898005; −0,0179123]	0,0539365	0	1,1898005
6	[0,5572619; 1,4194409]	−1,1089001	0	0,4427380
7	[0,1068314; 0,4559102]	−0,4444973	0	0,8931685
8	[2,3369698; 1,8247078]	3,6367201	4,7909865	0
9	[2,5410392; 2,6737468]	3,7831337	9,6058027	0

Tabla 6.3: Población Mutante por mutación tradicional.

muestra en las tablas 6.3 y 6.4. Es importante destacar que los vectores 1, 2, 4 *y* 8 *de la población mutante generada por la mutación selectiva son mejores que los equivalentes mutantes creados por la mutación tradicional. Si bien ninguno de ellos es factible, en los cuatro casos la violación a las restricciones se da en menor medida.*

Para asegurar qué vectores formarán parte de la población en la epoch $k + 1$ *es necesario contar con la información del operador de selección, puesto que pueden modificar el resultado del* fitness *o seleccionar por factibilidad, entre otros criterios. Es importante remarcar que la población mutante, para la* epoch k*, generada por la mutación selectiva presenta menor grado de violación que la población mutante generada por el operador tradicional de mutación. Ésta es la idea principal detrás de la mutación selectiva, lograr una población factible o con el menor grado de violación posible.*

6.4. Algoritmo ED+HC2

Al momento de generar el algoritmo ED+HC2, es posible implementar varios esquemas sobre el operador de mutación:

- Mutación selectiva a lo largo de toda la evolución.
- Mutación selectiva en la primera etapa de la evolución.

i	$\mathbf{x}_i$	$\phi(\mathbf{x}_i)$	$g_1(\mathbf{x}_i)$	$g_2(\mathbf{x}_i)$
0	[0,8632855; 1,8093881]	−1,0641261	0,0191475	0,1367144
1	[2,1525427; 1,6004539]	3,0329861	3,1948929	0
2	[0,9705400; 0,0457429]	0,8962050	0	0,0294599
3	[0,5484478; 1,2195055]	−0,9187104	0	0,4515521
4	[−0,1734410; 0,6711759]	−0,6410941	0	1,1734410
5	[−0,1898005; −0,0179123]	0,0539365	0	1,1898005
6	[0,5572619; 1,4194409]	−1,1089001	0	0,4427380
7	[0,1068314; 0,4559102]	−0,4444973	0	0,8931685
8	[1,8419645; 1,0040860]	2,3887472	0,4010221	0
9	[2,5410392; 2,6737468]	3,7831337	9,6058027	0

Tabla 6.4: Población Mutación por mutación selectiva.

- Mutación selectiva de acuerdo a la factibilidad de la población.

Además de la modificación con respecto a la mutación, se mantiene el esquema anterior de búsqueda local por *Hill Climbing* modificado sobre el mejor individuo de cada *epoch*, a lo largo de toda la evolución.

Aplicar sólo mutación selectiva restringe la exploración del espacio de búsqueda al priorizar el uso de los individuos factibles (o con menor grado de violación de restricciones) como vector base. Es posible que este esquema lleve a una convergencia prematura de la población, sobre todo en el caso de tratar de optimizar funciones con varios mínimos locales. Emplear en forma alternada a la mutación selectiva y la tradicional, en forma dependiente de la factibilidad de la población es tentador, pero para ello es necesario poder decidir en qué momento la explotación será intensiva y cuándo la exploración será primordial, lo cual es complicado si no se conoce la función sobre la cual se aplica el algoritmo. Por lo antes mencionado una opción lógica es aplicar la mutación selectiva sobre la primera mitad del tiempo de evolución, para lograr que la población se encuentre dentro del espacio factible de búsqueda y el tiempo restante aplicar la mutación tradicional, que conjuga las capacidades de exploración y explotación en forma equilibrada.

6.5. Experimentos y resultados

El algoritmo ED+HC2, hibridación de ED con HCMod junto con los dos operadores de mutación aplicados en etapas, presenta las siguientes características al momento de ser ejecutado:

- **Inicialización de la población**: aleatoria. Se utilizó un generador de números pseudoaleatorios que siguen una distribución uniforme y con una transformación lineal se obtuvieron valores dentro del rango permitido en cada variable.
- **Mutación**:
  - selectiva, aplicada sobre la primera mitad del tiempo de evolución. Para generar el vector mutante se utilizó el esquema

$$\mathbf{v}_{i,g} = \mathbf{x}_{A,g} + F \cdot (\mathbf{x}_{B,g} - \mathbf{x}_{C,g}),$$

donde A es el índice del *mejor* vector de los tres seleccionados, de acuerdo a la factibilidad como primera valoración y al *fitness* como valoración secundaria, B y C son los índices de los dos restantes, todos índices diferentes entre sí.

  - clásica, aplicada sobre la segunda mitad del tiempo de evolución. Para generar el vector mutante se utilizó el esquema

$$\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) ,$$

    donde $r0$, $r1$ y $r2$ son índices escogidos al azar, distintos entre sí.

- **Cruzamiento**: uniforme. También conocido como recombinación discreta ya que recombina dos vectores diferentes seleccionando, para crear el nuevo vector, una componente a la vez.
- **Selección**: clásica. Dados dos individuos a comparar, permanece en la población aquel que tenga mejor valor de ajuste.
- **Hibridación**: con HCMod. Se aplicó el buscador local al mejor individuo obtenido en cada *epoch*.
- **Manejo de restricciones**: penalidad estática, con la nueva función de penalidad

$$\phi(\mathbf{x}) = f(\mathbf{x}) + c_p \left(\sum_{i=1}^{m} G_i + \sum_{j=1}^{p} L_j \right) ,$$

  con $c_p = 5$. Es posible utilizar un valor pequeño como coeficiente de penalidad puesto que la principal encargada de *castigar* a los individuos no factibles es la mutación selectiva. Las restricciones de igualdad, $h_j(\mathbf{x})$, fueron convertidas en restricciones de desigualdad utilizando la tolerancia $\varepsilon = 1 \times 10^{-4}$, como se plantea en la ecuación 3.3.

- **Parámetros del algoritmo para 10D**:
  - Tamaño de la población: 41 individuos.
  - Tiempo de evolución: 4000 *epochs*.
  - Probabilidad de cruzamiento: 0.6, valor sugerido por los autores de ED.
  - Factor de escala en mutación: 0.6, valor sugerido por los autores de ED.
  - Exploraciones por HCMod: 3
  - Variables exploradas por HCMod en cada *epoch*: 2
- **Parámetros del algoritmo para 30D**:
  - Tamaño de la población: 55 individuos.
  - Tiempo de evolución: 7000 *epochs*.
  - Probabilidad de cruzamiento: 0.6, valor sugerido por los autores de ED.
  - Factor de escala en mutación: 0.6, valor sugerido por los autores de ED.
  - Exploraciones por HCMod: 3
  - Variables exploradas por HCMod en cada *epoch*: 6

Para comprobar la eficiencia del algoritmo ED+HC2 se utilizó el suite de funciones del *Single Objective Constrained Real-Parameter Optimization* del CEC2010 [29], con 10D y 30D. Todas las ejecuciones de esta propuesta fueron ejecutadas bajo MatLab 7.9 x64, con una HP EliteBook (Intel Core2Duo P8600, 2.40GHz, 4Gb RAM). Los resultados obtenidos se comparan con el algoritmo ganador de la sesión especial, ϵDEag, presentado por Takahama y Sakai [52]. A fin de equiparar condiciones de ejecución, se consideró un máximo

	F01		**F02**	
	ED+HC2	**εDEag**	**ED+HC2**	**εDEag**
Mejor	-7,473103E-01	-7,473104E-01	-2,277707E+00	-2,277702E+00
Mediana	-7,405572E-01	-7,473104E-01	-2,271573E+00	-2,269502E+00
Peor	-5,697009E-01	-7,405572E-01	-2,259051E+00	-2,174499E+00
Promedio	-7,308432E-01	-7,470102E-01	-2,271062E+00	-2,258870E+00
Desviación	3,731569E-02	1,323339E-03	3,991102E-03	2,389779E-02
	F03		**F04**	
Mejor	0,000000E+00	0,000000E+00	-9,998582E-06	-9,992345E-06
Mediana	0,000000E+00	0,000000E+00	-9,984340E-06	-9,977276E-06
Peor	0,000000E+00	0,000000E+00	9,853494E-01	-9,282295E-06
Promedio	0,000000E+00	0,000000E+00	3,940443E-02	-9,918452E-06
Desviación	0,000000E+00	0,000000E+00	1,970719E-01	1,546730E-07

Tabla 6.5: Resultados de los experimentos para 10D con las funciones F01 a F04.

de 6×10^5 evaluaciones de función para 30D y 2×10^5 para 10D. En las tablas 6.5 y 6.6 se presentan los resultados para 10D, mientras que en las tablas 6.7 y 6.8 se muestran los resultados para 30D, para ambas dimensiones se ejecutó 25 veces el algoritmo ED+HC2.

A modo de resumen, el mejor desempeño en los diversos tópicos fue obtenido por:

- Para 10D:
  - Mejor: ED+HC2 (6), εDEag (2), Empate (10)
  - Mediana: ED+HC2 (3), εDEag (8), Empate (7)
  - Peor: ED+HC2 (5), εDEag (9), Empate (4)
  - Promedio: ED+HC2 (2), εDEag (12), Empate (4)
  - Desviación: ED+HC2 (5), εDEag (10), Empate (3)
- Para 30D:
  - Mejor: ED+HC2 (15), εDEag (3), Empate (0)
  - Mediana: ED+HC2 (14), εDEag (4), Empate (0)
  - Peor: ED+HC2 (10), εDEag (8), Empate (0)
  - Promedio: ED+HC2 (12), εDEag (6), Empate (0)
  - Desviación: ED+HC2 (8), εDEag (10), Empate (0)

6.6. Conclusiones

Se presentó el algoritmo ED+HC2, resultado de aplicar dos versiones de mutación diferencial (selectiva y tradicional) sobre el algoritmo híbrido ED+HC. La mutación selectiva, que hace foco en los individuos factibles o en aquellos no factibles cuyo grado de violación sea el menor, se aplica en la primera mitad del tiempo de evolución favoreciendo la explotación de los individuos factibles o de aquellos que poseen menor grado de violación a las restricciones. La mutación tradicional opera con una combinación de exploración y explotación pero ya dentro de regiones potencialmente factibles, durante la segunda parte del tiempo de evolución. Además de ambas versiones de mutación diferencial, se aplica HCMod al mejor individuo de cada *epoch*. Para ver la calidad de los resultados obtenidos

	F05		**F06**	
	ED+HC2	**εDEag**	**ED+HC2**	**εDEag**
Mejor	-4,836106E+02	-4,836106E+02	-5,786625E+02	-5,786581E+02
Mediana	-4,835963E+02	-4,836106E+02	-5,786624E+02	-5,786533E+02
Peor	-4,764231E+02	-4,836106E+02	-5,786462E+02	-5,786448E+02
Promedio	-4,832203E+02	-4,836106E+02	-5,786607E+02	-5,786528E+02
Desviación	1,433711E+00	3,890350E-13	4,111062E-03	3,627169E-03
	F07		**F08**	
Mejor	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00
Mediana	0,000000E+00	0,000000E+00	1,094154E+01	1,094154E+01
Peor	0,000000E+00	0,000000E+00	1,094154E+01	1,537535E+01
Promedio	0,000000E+00	0,000000E+00	8,131174E+00	6,727528E+00
Desviación	0,000000E+00	0,000000E+00	4,726336E+00	5,560648E+00
	F09		**F10**	
Mejor	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00
Mediana	0,000000E+00	0,000000E+00	2,807824E+01	0,000000E+00
Peor	3,435308E+01	0,000000E+00	1,398129E+02	0,000000E+00
Promedio	4,369718E+00	0,000000E+00	3,193498E+01	0,000000E+00
Desviación	1,038857E+01	0,000000E+00	2,764394E+01	0,000000E+00
	F11		**F12**	
Mejor	-1,522713E-03	-1,522713E-03	-5,701310E+02	-5,700899E+02
Mediana	-1,522713E-03	-1,522713E-03	-2,099679E-01	-4,231332E+02
Peor	-1,522713E-03	-1,522713E-03	-2,099679E-01	-1,989129E-01
Promedio	-1,522713E-03	-1,522713E-03	-9,890186E+01	-3,367349E+02
Desviación	2,193770E-12	6,3410350E-11	1,495104E+02	1,782166E+02
	F13		**F14**	
Mejor	-6,842937E+01	-6,842937E+01	0,000000E+00	0,000000E+00
Mediana	-6,842936E+01	-6,842936E+01	0,000000E+00	0,000000E+00
Peor	-6,227639E+01	-6,842936E+01	0,000000E+00	0,000000E+00
Promedio	-6,671313E+01	-6,842936E+01	0,000000E+00	0,000000E+00
Desviación	2,135940E+00	1,025960E-06	0,000000E+00	0,000000E+00
	F15		**F16**	
Mejor	0,000000E+00	0,000000E+00	6,072207E-01	0,000000E+00
Mediana	3,673239E+00	0,000000E+00	1,029367E+00	2,819841E-01
Peor	4,497444E+00	4,497445E+00	1,091488E+00	1,018265E+00
Promedio	2,383841E+00	1,798978E-01	9,995397E-01	3,702054E-01
Desviación	1,832005E+00	8,813156E-01	1,052056E-01	3,710479E-01
	F17		**F18**	
Mejor	0,000000E+00	1,463180E-17	1,688506E-09	3,7314390E-20
Mediana	1,088416E+00	5,653326E-03	3,599014E-06	4,0979090E-19
Peor	1,088416E+00	7,301765E-01	7,973679E-05	9,2270270E-18
Promedio	8,271967E-01	1,249561E-01	1,268129E-05	9,6787650E-19
Desviación	4,744298E-01	1,937197E-01	2,100227E-05	1,8112340E-18

Tabla 6.6: Resultados de los experimentos para 10D con las funciones F05 a F18.

	F01		**F02**	
	ED+HC2	**εDEag**	**ED+HC2**	**εDEag**
Mejor	-8,218818E-01	-8,218255E-01	-2,222520E+00	-2,169248E+00
Mediana	-7,953880E-01	-8,206172E-01	-2,186000E+00	-2,152145E+00
Peor	-7,018661E-01	-8,195466E-01	-2,127790E+00	-2,117096E+00
Promedio	-7,831150-01	-8,208687E-01	-2,186381E+00	-2,151424E+00
Desviación	3,215000E-02	7,103893E-04	2,362188E-02	1,197582E-02
	F03		**F04**	
Mejor	0,000000E+00	2,867347E+01	2,079489E-03	4,698111E-03
Mediana	2,610183E-25	2,867347E+01	5,786755E-03	6,947614E-03
Peor	7,922700E-24	3,278014E+01	3,451393E-02	1,777889E-02
Promedio	2,019099E-24	2,883785E+01	8,363904E-03	8,162973E-03
Desviación	2,605635E-24	8,047159E-01	7,123662E-03	3,067785E-03
	F05		**F06**	
Mejor	-4,633097E+02	-4,531307E+02	-5,275465E+02	-5,285750E+02
Mediana	-4,563934E+02	-4,500404E+02	-5,246824E+02	-5,280407E+02
Peor	-4,317171E+02	-4,421590E+02	-5,164730E+02	-5,264539E+02
Promedio	-4,546790E+02	-4,495460E+02	-5,244224E+02	-5,279068E+02
Desviación	7,405241E+00	2,899105E+00	2,781389E+00	4,748378E-01
	F07		**F08**	
Mejor	2,470120E-29	1,147112E-15	2,470120E-29	2,518693E-14
Mediana	2,470120E-29	2,114429E-15	2,470120E-29	6,511508E-14
Peor	2,470120E-29	5,481915E-15	2,470120E-29	2,578112E-13
Promedio	2,470120E-29	2,603632E-15	2,470120E-29	7,831464E-14
Desviación	0,000000E+00	1,233430E-15	0,000000E+00	4,855177E-14
	F09		**F10**	
Mejor	0,000000E+00	2,770665E-16	3,043152E+01	3,252002E+01
Mediana	0,000000E+00	1,124608E-08	3,130917E+01	3,328903E+01
Peor	1,520340E+02	1,052759E+02	2,007061E+02	3,463243E+01
Promedio	6,081360E+00	1,072140E+01	5,780908E+01	3,326175E+01
Desviación	3,040680E+01	2,821923E+01	5,259428E+01	4,545577E-01
	F11		**F12**	
Mejor	-3,923425E-04	-3,268462E-04	-5,623711E+02	-1,991453E-01
Mediana	-3,923085E-04	-2,843296E-04	-2,110883E-01	5,337125E+02
Peor	-3,920672E-04	-2,236338E-04	-2,110883E-01	5,461723E+02
Promedio	-3,922902E-04	-2,863882E-04	-8,769951E+01	3,562330E+02
Desviación	6,004102E-08	2,707605E-05	1,792221E+02	2,889253E+02
	F13		**F14**	
Mejor	-6,490832E+01	-6,642473E+01	0,000000E+00	5,015863E-14
Mediana	-6,231389E+01	-6,531507E+01	0,000000E+00	1,359306E-13
Peor	-4,693716E+01	-6,429690E+01	0,000000E+00	2,923513E-12
Promedio	-6,024183E+01	-6,535310E+01	0,000000E+00	3,089407E-13
Desviación	5,619378E+00	5,733005E-01	0,000000E+00	5,608409E-13

Tabla 6.7: Resultados de los experimentos para 30D con las función F01 a F14.

	F15		F16	
	ED+HC2	**εDEag**	**ED+HC2**	**εDEag**
Mejor	2,160323E+01	2,160345E+01	1,096930E+00	0,000000E+00
Mediana	2,160333E+01	2,160375E+01	1,166631E+00	0,000000E+00
Peor	2,160370E+01	2,160403E+01	1,213863E+00	5,421011E-20
Promedio	2,160336E+01	2,160376E+01	1,159892E+00	2,168404E-21
Desviación	1,289045E-04	1,104834E-04	3,165018E-02	1,062297E-20
	F17		**F18**	
Mejor	3,120006E-32	2,165719E-01	3,626268E-03	1,226054E+00
Mediana	3,513466E-01	5,315949E+00	9,556747E-01	2,679497E+01
Peor	3,514799E-01	1,889064E+01	3,375154E+00	7,375363E+02
Promedio	2,810995E-01	6,326487E+00	1,112360E+00	8,754569E+01
Desviación	1,434480E-01	4,986691E+00	9,276525E-01	1,664753E+02

Tabla 6.8: Resultados de los experimentos para 30D con las función F15 a F18.

por ED+HC2 se realiza la comparación contra εDEag, de Takahama y Sakai. Para 10D se observa que ED+HC2 es superior a εDEag 21 veces y se logran 28 empates, sobre un total de 90 comparaciones; mientras que para 30D la diferencia es notoria a favor del nuevo algoritmo, 59 veces es superior y no se obtienen empates. Esta propuesta de modificación a ED presenta mejores resultados que su versión anterior, ED+HC, por lo que se continuará con esta línea de trabajo para poder crear algún otro operador que, en conjunto con los aportes realizados, obtenga resultados de mayor calidad.

Capítulo 7

Propuesta III - ED+HC3

El algoritmo ED+HC3 es la evolución de ED+HC2 [15], siendo un algoritmo híbrido (se utiliza un buscador local sobre cada *epoch* de la evolución), se aplican dos versiones de mutación diferencial (selectiva y tradicional) en etapas y al que se agregan dos nuevos operadores: el primero de ellos, *reparación*, y una nueva función de selección, por torneo. El operador de reparación selecciona un individuo no factible y se intenta reducir su nivel violación a las restricciones, sin que se convierta en un individuo factible. La selección por torneo prioriza a los individuos factibles sobre los no factibles, o el menor grado de violación a las restricciones en el caso de no operar con individuos factibles.

7.1. Introducción

El algoritmo ED+HC2 demostró un buen desempeño en las ejecuciones realizadas, sin embargo es posible mejorar aún más su rendimiento. Para lograr este objetivo será necesaria la introducción de nuevos operadores. La versión clásica de ED incluye tres operadores dentro de la evolución: mutación, cruzamiento y selección, encargados de agregar diversidad a la población y además permitiendo que sobrevivan aquellos que sean más aptos [53]. Pero ningún operador considera la resolución de problemas restringidos en forma directa, con lo que es necesaria una estrategia de manejo de restricciones. El operador reparación conjuga capacidades de exploración y manejo de restricciones. A su vez, la selección por torneo puede ser utilizada en forma independiente a la técnica de manejo de restricciones elegida, porque opera en base a la factibilidad de los individuos o su grado de violación en caso de comparar individuos no factibles.

7.2. Operador Reparación

El operador de reparación, propuesto para ser utilizado en ED+HC3, aprovecha la no factibilidad de un individuo para explorar sólo sobre una de sus restricciones no satisfechas. La idea central es reducir su grado de violación sólo para esa restricción, sin lograr que el individuo en cuestión se transforme en un individuo factible. La forma de operar es simple, primero se recorre la población actual en forma aleatoria y se comprueba si la restricción k-ésima es satisfecha, con $1 \leq k \leq m + p$ siendo $m + p$ el total de restricciones y k elegido en forma aleatoria. Si existe ese individuo, es necesario elegir un individuo factible para la restricción k. Con ambos identificados comienza el proceso de reparación. Se generan q vectores test de la siguiente forma:

$$\mathbf{x}_{T_i} = (\mathbf{x}_F - \mathbf{x}_{NF})rand + \mathbf{x}_{NF},$$

i	$\mathbf{x}_i$	$\phi(\mathbf{x}_i)$	$g_1(\mathbf{x}_i)$	$g_2(\mathbf{x}_i)$	$h_1(\mathbf{x}_i)$
0	$[-1{,}921648; -1{,}899139; -1{,}889120]$	0,869251	0	0	0,868238
1	$[-2{,}303754; -1{,}228373; -2{,}959694]$	5,576208	0	0	5,575978
2	$[0{,}332058; -0{,}867762; 2{,}211049]$	5,159983	0	0,383179	$-4{,}247984$
3	$[-2{,}373234; -2{,}347475; -0{,}513800]$	1,464004	0	0	1,406875
4	$[-2{,}362045; -2{,}120837; -2{,}025087]$	4,178232	0	0	4,178193
5	$[-2{,}215429; -2{,}673303; -0{,}726602]$	2,596157	0	0	2,582633
6	$[-2{,}198161; 2{,}392667; 2{,}266776]$	41,49240	31,72098	4,076362	5,695047
7	$[-2{,}039852; -2{,}885225; -2{,}251047]$	7,552738	0	0	7,552736
8	$[2{,}467925; 0{,}632253; -0{,}292589]$	20,34148	0	16,28402	$-3{,}423990$
9	$[0{,}216656; -2{,}127955; -2{,}067355]$	10,44913	6,704423	0	$-1{,}150902$

Tabla 7.1: Población luego de u *epochs*.

con $i = 1, 2, \ldots, q$ y donde $\mathbf{x}_F$ y $\mathbf{x}_{NF}$ representan a los vectores factible y no factible para la restricción k, respectivamente. Como $rand \in [0; 1]$ entonces $\mathbf{x}_{T_i}$ es una transformación lineal entre los vectores seleccionados y en algún valor interior al intervalo $[0; 1]$ pasará de ser no factible a factible para la restricción k-ésima. De los m vectores test será escogido para reemplazar a $\mathbf{x}_{NF}$ en la población a aquel que, siendo no factible para la restricción k-ésima, tenga un valor de ajuste menor que $\phi(\mathbf{x}_{NF})$. También es posible que en esa exploración aleatoria, no se consiga un vector test mejor que $\mathbf{x}_{NF}$. En ese caso, $\mathbf{x}_{NF}$ no será reemplazado y seguirá en la población por, al menos, una *epoch* más. Es importante recalcar que nunca se permite que el $\mathbf{x}_{NF}$ sea reemplazado por un individuo factible, sino que la idea es explorar sobre la no factibilidad de la restricción k sin importar en qué estado se encuentran las demás.

Ejemplo 3. *Se desea optimizar la función de tres dimensiones*

$$f(\mathbf{x}) = e^{x_1 x_2 x_3},$$

en el espacio $-3 \leq x_i \leq 3$ *para* $i = 1, 2, 3$*, sujeta a las restricciones*

$$g_1(\mathbf{x}) = x_2 x_3 - 5 x_1 x_2 \leq 0, \qquad g_2(\mathbf{x}) = x_1^3 + x_2^3 - 1 \leq 0, \qquad h_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 = 10.$$

Luego de u epochs*, se tiene la población de 10 vectores que se muestra en la tabla 7.1, donde además figuran los valores de* fitness *y de las restricciones* g_1*,* g_2 *y* h_1*.*

Luego, se sortea el orden en que será recorrida la población en busca de un individuo no factible:

$$SorteoPos = [2; 6; 9; 1; 0; 8; 7; 4; 3; 5],$$

y el orden de exploración de restricciones:

$$SorteoRest = [1; 3; 2],$$

con lo que se busca un individuo que sea no factible para la primera restricción, g_1*. De acuerdo al orden establecido, se selecciona* $\mathbf{x}_{NF} = \mathbf{x}_6$*. Ahora se recorre en el orden habitual a los individuos de la población, para determinar si existe alguno que tenga la primera restricción satisfecha. Con este proceso se selecciona* $\mathbf{x}_F = \mathbf{x}_0$*. Se procederá a realizar la reparación entre* $\mathbf{x}_{NF}$ *y* $\mathbf{x}_F$ *de acuerdo a la cantidad de exploraciones solicitadas, en este caso 3:*

$$\mathbf{x}_{T_i} = (\mathbf{x}_F - \mathbf{x}_{NF}) rand + \mathbf{x}_{NF}.$$

Como $rand = [0{,}453257; 0{,}618069; 0{,}799639]$ *entonces:*

- $\mathbf{x}_{T_1} = [-2{,}072829; 0{,}447375; 0{,}383087]$
- $\mathbf{x}_{T_2} = [-2{,}027256; -0{,}259965; -0{,}301854]$
- $\mathbf{x}_{T_3} = [-1{,}977050; -1{,}039228; -1{,}056440]$

Al evaluar a los tres nuevos individuos test, para decidir cuál de ellos es mejor que $\mathbf{x}_{NF}$ *(en el caso de que alguno cumpla con las condiciones), debe tenerse en cuenta que su valor de ajuste debe ser menor al de* $\mathbf{x}_6$ *y además debe ser no factible para* g_1*. Luego de evaluarlos, se obtiene:*

- $\phi(\mathbf{x}_{T_1}) = 0{,}700998$, $g_1(\mathbf{x}_{T_1}) = 4{,}808053$
- $\phi(\mathbf{x}_{T_2}) = 0{,}852926$, $g_1(\mathbf{x}_{T_2}) = -2{,}556609$
- $\phi(\mathbf{x}_{T_3}) = 0{,}114112$, $g_1(\mathbf{x}_{T_3}) = -9{,}175152$

Dados los resultados obtenidos, se selecciona a $\mathbf{x}_{T_1}$ *para que ocupe el lugar de* $\mathbf{x}_6$ *en la población. Esto es así a pesar de que el valor de ajuste de* $\mathbf{x}_{T_3}$ *es menor que el de* $\mathbf{x}_{T_1}$*, dado que la selección se sustenta porque* $\mathbf{x}_{T_3}$ *es factible para* g_1 *y* $\mathbf{x}_{T_1}$ *no.*

7.3. Operador Selección por torneo

La técnica de manejo de restricciones denominada *torneo binario*, fue presentada por Deb en el año 2000 como un operador de selección para algoritmos genéticos [7]. La mecánica es simple, puesto que se basa en la factiblidad como primer criterio y luego en el ajuste de los individuos que compiten en el torneo:

- Cuando se comparan dos individuos factibles, es elegido para permanecer en la población aquel que tenga menor valor de ajuste.
- Cuando el torneo es entre un individuo factible y otro no factible, se selecciona el individuo factible.
- Cuando la comparación es entre dos individuos no factibles, es elegido aquel que tenga menor grado de violación de las restricciones. El grado de violación de las restricciones se calcula de la siguiente forma:

$$\phi(\mathbf{x}) = \sum_{i=1}^{m} \text{máx}(0, g_i(\mathbf{x}))^2 + \sum_{j=1}^{p} |h_j(\mathbf{x})|, \tag{7.1}$$

  donde los valores de las restricciones de desigualdad, g_i, y las restricciones de igualdad, h_i están normalizados.

Para utilizar esta técnica como operador de selección dentro de ED+HC3 se realizaron los siguientes cambios:

- Cuando los dos individuos participantes en el torneo tienen la misma condición de factibilidad, es elegido para permanecer en la población aquel que tenga menor valor de ajuste, independiente del grado de violación en el caso de individuos no factibles.
- La función de ajuste no es la planteada en la ecuación 7.1, sino que en este caso además involucra una penalidad dinámica en el cálculo del ajuste de los individuos:

$$\phi(\mathbf{x}, t) = f(\mathbf{x}) + \frac{kt}{T}\left(\sum_{i=1}^{m} \text{máx}[g_i(\mathbf{x}), 0] + \sum_{j=1}^{p} |h_j(\mathbf{x})|\right), \tag{7.2}$$

  donde $k = 4$ para 10D y $k = 2$ para 30D.

exp	$\mathbf{x}_{best}$	$\phi(\mathbf{x}_{best})$	estado	promedio	nofac	promnofac
1	[1,985443; 1,004734]	2,019878	F	2,037780	1	$5{,}370E-03$
2	[1,993592; 0,999344]	2,014168	F	2,055482	3	$3{,}855E-02$
3	[2,002619; 0,994167]	2,006466	F	2,063534	2	$4{,}416E-02$
4	[2,005261; 0,988569]	2,012495	F	2,063534	1	$2{,}181E-02$
5	[1,996848; 1,000803]	2,004705	F	2,077474	1	$6{,}493E-02$

Tabla 7.2: Resultados de aplicar selección por torneo original.

exp	$\mathbf{x}_{best}$	$\phi(\mathbf{x}_{best})$	estado	promedio	nofac	promnofac
1	[1,967155; 0,999268]	2,068230	F	2,073271	0	$--$
2	[1,997421; 0,997118]	2,010935	F	2,038482	1	$8{,}167E-03$
3	[1,992650; 0,967194]	2,081441	F	2,106116	1	$1{,}944E-02$
4	[1,913360; 0,997603]	2,185583	F	2,307037	0	$--$
5	[1,984120; 0,992634]	2,046796	F	2,079039	0	$--$

Tabla 7.3: Resultados de aplicar selección por torneo modificada.

Ejemplo 4. *Se desea optimizar la función de dos dimensiones:*

$$f(\mathbf{x}) = (x_1 - 3)^2 + (x_2 - 2)^2 ,$$

en el espacio $0 \leq x_i \leq 10$ *para* $i = 1, 2$*, sujeta a las restricciones:*

$$g_1(\mathbf{x}) = x_1^2 + x_2^2 - 5 \leq 0; \qquad g_2(\mathbf{x}) = 2x_1 + x_2 - 6 \leq 0; \qquad g_3(\mathbf{x}) = x_1 + 2x_2 - 4 \leq 0$$

Con el fin de ver el resultado de aplicar cada uno de los operadores mencionados, se creará una versión simple de ED con los siguientes parámetros:

- *Población: 10 individuos*
- *Evolución: 75* epochs
- *Factor de escala para la mutación: 0.6*
- *Probabilidad de cruzamiento: 0.6*

Este algoritmo se ejecutó 5 veces con la selección por torneo planteada por Deb y luego la misma cantidad de veces con la selección por torneo modificada. Los resultados se muestran en las tablas 7.2 y 7.3 respectivamente, donde se explicitan para cada experimento: el mejor individuo obtenido, $\mathbf{x}_{best}$*; su valor de ajuste,* $\phi(\mathbf{x}_{best})$*; el estado del mejor individuo, Factible o No Factible; el promedio de los ajustes, la cantidad de invididuos no factibles y el promedio de las violaciones a las restricciones de los individuos no factibles, éstos últimos tres valores calculados sobre la población final.*

Luego de la ejecución de prueba de los algoritmos con las diferentes versiones de selección por torneo es posible realizar los siguientes comentarios:

- *Ambas versiones operan bien, aunque la versión original logra mejores resultados finales, posiblemente porque mantiene mayor diversidad poblacional que la versión modificada.*
- *La versión modificada mostró poblaciones finales con pocos o ningún individuo no factible, esto tal vez se deba a que una de las diferencias que tiene con respecto a la versión original es al comparar dos individuos no factibles, la comparación se realiza en base a los valores de ajuste.*

- *Es posible suponer que la selección por torneo modificada permitirá una amplia explotación de las regiones promisorias, al interactuar con el operador de reparación y la función de penalidad dinámica.*
- *El problema presentado es de baja dimensión y posee restricciones simples. Tal vez el comportamiento de ambos algoritmos no sea el mismo ante problemas de mayor dimensionalidad.*

7.4. Algoritmo ED+HC3

Para generar el algoritmo ED+HC3 se tuvieron en cuenta los lineamientos en los que se basaron ED+HC [14] y ED+HC2 [15]:

- Hibridación, aplicándose un buscador local con selección aleatoria basado en Hill Climbing, denominado *HCMod*, sobre el mejor individuo de cada *epoch*.
- Doble operador de mutación: utilizando durante la primera mitad del tiempo de evolución la *mutación selectiva*, que hace foco en los individuos factibles o en el caso de no haber factibles, sobre aquellos cuyo grado de violación a las restricciones sea el menor; para luego usar la *mutación tradicional*, creada por Storn y Price, donde se conjuga la explotación y exploración sobre la población actual.

ED+HC3 se define al agregar dos nuevos operadores: *reparación*, que explora el espacio de búsqueda alrededor de la no factibilidad en una restricción, escogida al azar, de algún individuo de la población con el objetivo de bajar su grado de violación y *selección por torneo*, que prefiere a los individuos factibles sobre los no factibles al momento de decidir cuál de dos individuos permanecerá en la población. Como estos dos operadores favorecen la explotación más que la exploración, sumados al uso de la mutación selectiva en la primera mitad del tiempo de evolución, no es conveniente el uso de penalidad estática para realizar el manejo de restricciones. Por ello se utilizó un factor de penalidad dinámica, que tomó valores muy bajos para no lograr convergencia prematura.

7.5. Experimentos y resultados

El algoritmo ED+HC3 presenta las siguientes características al momento de ser ejecutado:

- **Inicialización de la población**: aleatoria. Se utilizó un generador de números pseudoaleatorios que siguen una distribución uniforme y con una transformación lineal se obtuvieron valores dentro del rango permitido en cada variable.
- **Mutación**:
  - selectiva, aplicada sobre la primera mitad del tiempo de evolución. Para generar el vector mutante se utilizó el esquema

    $$\mathbf{v}_{i,g} = \mathbf{x}_{A,g} + F \cdot (\mathbf{x}_{B,g} - \mathbf{x}_{C,g}) ,$$

    donde A es el índice del *mejor* vector, de acuerdo a la factibilidad como primera valoración y al *fitness* como valoración secundaria, de los tres seleccionados y B y C son los dos restantes, todos índices diferentes entre sí.
  - clásica, aplicada sobre la segunda mitad del tiempo de evolución. Para generar el vector mutante se utilizó el esquema

    $$\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) ,$$

    donde $r0$, $r1$ y $r2$ son índices escogidos al azar, distintos entre sí.

- **Cruzamiento**: uniforme. También conocido como recombinación discreta ya que recombina dos vectores diferentes seleccionando, para crear el nuevo vector, una componente a la vez.
- **Selección**: por torneo. Dados dos individuos, se tendrá en cuenta su pertenencia al espacio factible:
  - Ambos factibles, se considera superior a aquel que tenga el menor valor de ajuste.
  - Uno factible, es superior al no factible, independiente de su valor de ajuste.
  - Ninguno factible, se considera superior a quien tenga menor valor de ajuste.
- **Hibridación**: con HCMod. Se aplicó el buscador local al mejor individuo obtenido en cada *epoch*.
- **Manejo de restricciones**: penalidad dinámica. Se considera el factor dinámico $\frac{4t}{T}$ para 10D y $\frac{2t}{T}$ para 30D, donde t es la *epoch* actual y T el tiempo completo de evolución. Se utiliza este valor pequeño puesto que la mutación selectiva y la selección por torneo ya priorizan a los individuos factibles. Se supone que su aplicación será total cuando se esté en la segunda parte del tiempo de evolución, cuando se aplique mutación tradicional y la población se espera que sea compacta. Las restricciones de desigualdad, $h_j(\mathbf{x})$, fueron convertidas en restricciones de igualdad utilizando una tolerancia $\varepsilon = 1 \times 10^{-4}$, como se plantea en la ecuación 3.3.
- **Parámetros del algoritmo para 10D**:
  - Tamaño de la población: 45 individuos.
  - Tiempo de evolución: 3500 *epochs*.
  - Probabilidad de cruzamiento: 0.6, valor sugerido por los autores de ED.
  - Factor de escala en mutación: 0.6, valor sugerido por los autores de ED.
  - Exploraciones por HCMod: 3
  - Variables exploradas por HMMod en cada *epoch*: 2
  - Intentos de reparación por *epoch*: 3
- **Parámetros del algoritmo para 30D**:
  - Tamaño de la población: 60 individuos.
  - Tiempo de evolución: 6250 *epochs*.
  - Probabilidad de cruzamiento: 0.6, valor sugerido por los autores de ED.
  - Factor de escala en mutación: 0.6, valor sugerido por los autores de ED.
  - Exploraciones por HCMod: 9
  - Variables exploradas por HMMod en cada *epoch*: 5
  - Intentos de reparación por *epoch*: 9

Para comprobar la eficiencia del algoritmo ED+HC3 se utilizó el suite de funciones del *Single Objective Constrained Real-Parameter Optimization* del CEC2010 [29], con 10D y 30D. Todas las ejecuciones de esta propuesta fueron ejecutadas bajo MatLab 7.9 x64, con una HP EliteBook (Intel Core2Duo P8600, 2.40GHz, 4Gb RAM). Los resultados obtenidos se comparan con el algoritmo ganador de la sesión especial, ϵDEag, presentado por Takahama y Sakai [52]. A fin de equiparar condiciones de ejecución, se consideró un máximo de 6×10^5 evaluaciones de función para 30D y 2×10^5 para 10D. En las tablas 7.4 y 7.5

	F01		F02	
	ED+HC3	**εDEag**	**ED+HC3**	**εDEag**
Mejor	-7,473104E-01	-7,473104E-01	-2,277711E+00	-2,277702E+00
Mediana	-7,473104E-01	-7,473104E-01	-2,275195E+00	-2,269502E+00
Peor	-7,207228E-01	-7,405572E-01	-2,250734E+00	-2,174499E+00
Promedio	-7,402043E-01	-7,470102E-01	-2,269501E+00	-2,258870E+00
Desviación	9,047869E-03	1,323339E-03	9,057079E-03	2,389779E-02
	F03		**F04**	
Mejor	0,000000E+00	0,000000E+00	-9,999996E-06	-9,992345E-06
Mediana	0,000000E+00	0,000000E+00	-9,999655E-06	-9,977276E-06
Peor	0,000000E+00	0,000000E+00	3,512163E-05	-9,282295E-06
Promedio	0,000000E+00	0,000000E+00	-7,606699E-06	-9,918452E-06
Desviación	0,000000E+00	0,000000E+00	9,122807E-06	1,546730E-07

Tabla 7.4: Resultados de los experimentos para 10D con las funciones F01 a F04.

se presentan los resultados para 10D, mientras que en las tablas 7.6 y 7.7 se muestran los resultados para 30D, para ambas dimensiones se ejecutó 25 veces el algoritmo ED+HC3.

A modo de resumen, el mejor desempeño en los diversos tópicos fue obtenido por:

- Para 10D:
  - Mejor: ED+HC3 (5), εDEag (2), Empate (11)
  - Mediana: ED+HC3 (6), εDEag (4), Empate (8)
  - Peor: ED+HC3 (6), εDEag (6), Empate (6)
  - Promedio: ED+HC3 (4), εDEag (8), Empate (6)
  - Desviación: ED+HC3 (6), εDEag (7), Empate (5)
- Para 30D:
  - Mejor: ED+HC3 (14), εDEag (4), Empate (0)
  - Mediana: ED+HC3 (12), εDEag (6), Empate (0)
  - Peor: ED+HC3 (11), εDEag (7), Empate (0)
  - Promedio: ED+HC3 (12), εDEag (6), Empate (0)
  - Desviación: ED+HC3 (9), εDEag (9), Empate (0)

7.6. Conclusiones

Se presentó el algoritmo ED+HC3, que aparece como la evolución de los algoritmos ED+HC y ED+HC2 puesto que conjuga un buscador local (HCMod), dos versiones de mutación diferencial (selectiva y clásica) aplicadas en etapas y dos nuevos operadores: reparación (aporte propio) y selección por torneo (aporte de Kalyanmoy Deb). Este nuevo algoritmo híbrido cuenta con la capacidad de explotación durante la primera mitad del tiempo de evolución, basándose en la mutación selectiva, el operador de reparación y la selección por torneo. Luego, en la segunda mitad del tiempo de evolución se utiliza la mutación tradicional en conjunto con la reparación y la selección por torneo. Como se supone que durante la primera mitad de la evolución ED+HC3 selecciona regiones promisorias donde los individuos son en su mayoría factibles, el factor de penalidad dinámica es un

	F05		**F06**	
	ED+HC3	**εDEag**	**ED+HC3**	**εDEag**
Mejor	-4,836106E+02	-4,836106E+02	-5,786623E+02	-5,786581E+02
Mediana	-4,836106E+02	-4,836106E+02	-5,786581E+02	-5,786533E+02
Peor	-4,835996E+02	-4,836106E+02	-5,785778E+02	-5,786448E+02
Promedio	-4,836093E+02	-4,836106E+02	-5,786497E+02	-5,786528E+02
Desviación	2,682482E-03	3,890350E-13	1,985450E-02	3,627169E-03
	F07		**F08**	
Mejor	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00
Mediana	0,000000E+00	0,000000E+00	3,986579E+00	1,094154E+01
Peor	0,000000E+00	0,000000E+00	1,094154E+01	1,537535E+01
Promedio	0,000000E+00	0,000000E+00	5,098012E+00	6,727528E+00
Desviación	0,000000E+00	0,000000E+00	4,743257E+00	5,560648E+00
	F09		**F10**	
Mejor	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00
Mediana	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00
Peor	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00
Promedio	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00
Desviación	0,000000E+00	0,000000E+00	0,000000E+00	0,000000E+00
	F11		**F12**	
Mejor	-1,522713E-03	-1,522713E-03	-3,037844E+02	-5,700899E+02
Mediana	-1,522713E-03	-1,522713E-03	-1,992458E-01	-4,231332E+02
Peor	-1,522713E-03	-1,522713E-03	-1,992458E-01	-1,989129E-01
Promedio	-1,522713E-03	-1,522713E-03	-3,180460E+01	-3,367349E+02
Desviación	1,712151E-11	6,3410350E-11	7,093899E+01	1,782166E+02
	F13		**F14**	
Mejor	-6,842937E+01	-6,842937E+01	0,000000E+00	0,000000E+00
Mediana	-6,351751E+01	-6,842936E+01	0,000000E+00	0,000000E+00
Peor	-5,007841E+01	-6,842936E+01	0,000000E+00	0,000000E+00
Promedio	-6,161541E+01	-6,842936E+01	0,000000E+00	0,000000E+00
Desviación	5,741035E+00	1,025960E-06	0,000000E+00	0,000000E+00
	F15		**F16**	
Mejor	0,000000E+00	0,000000E+00	4,701920E-03	0,000000E+00
Mediana	3,673239E+00	0,000000E+00	3,699685E-01	2,819841E-01
Peor	3,673239E+00	4,497445E+00	1,060098E+00	1,018265E+00
Promedio	2,494495E+00	1,798978E-01	4,722849E-01	3,702054E-01
Desviación	1,716216E+00	8,813156E-01	3,737756E-01	3,710479E-01
	F17		**F18**	
Mejor	0,000000E+00	1,463180E-17	0,000000E+00	3,7314390E-20
Mediana	0,000000E+00	5,653326E-03	0,000000E+00	4,0979090E-19
Peor	8,610911E-05	7,301765E-01	5,048709E-29	9,2270270E-18
Promedio	6,991810E-06	1,249561E-01	2,019484E-30	9,6787650E-19
Desviación	2,056091E-05	1,937197E-01	1,009742E-29	1,8112340E-18

Tabla 7.5: Resultados de los experimentos para 10D con las funciones F05 a F18.

	F01		**F02**	
	ED+HC3	**εDEag**	**ED+HC3**	**εDEag**
Mejor	-7,986089E-01	-8,218255E-01	-2,268952E+00	-2,169248E+00
Mediana	-7,653982E-01	-8,206172E-01	-2,239111E+00	-2,152145E+00
Peor	-6,389885E-01	-8,195466E-01	-2,190551E+00	-2,117096E+00
Promedio	-7,407093E-01	-8,208687E-01	-2,238687E+00	-2,151424E+00
Desviación	5,262887E-02	7,103893E-04	2,025430E-02	1,197582E-02
	F03		**F04**	
Mejor	0,000000E+00	2,867347E+01	8,252103E-03	4,698111E-03
Mediana	8,668533E-24	2,867347E+01	2,301251E-02	6,947614E-03
Peor	2,921925E-21	3,278014E+01	9,932350E-01	1,777889E-02
Promedio	2,367006E-22	2,883785E+01	2,493186E-01	8,162973E-03
Desviación	6,911892E-22	8,047159E-01	4,160523E-01	3,067785E-03
	F05		**F06**	
Mejor	-4,799756E+02	-4,531307E+02	-5,304863E+02	-5,285750E+02
Mediana	-3,680132E+02	-4,500404E+02	-5,247260E+02	-5,280407E+02
Peor	-3,399950E+02	-4,421590E+02	-5,034053E+02	-5,264539E+02
Promedio	-4,025294E+02	-4,495460E+02	-5,229468E+02	-5,279068E+02
Desviación	5,943368E+01	2,899105E+00	7,110502E+00	4,748378E-01
	F07		**F08**	
Mejor	2,470121E-29	1,147112E-15	2,470121E-29	2,518693E-14
Mediana	2,470121E-29	2,114429E-15	2,470121E-29	6,511508E-14
Peor	2,470121E-29	5,481915E-15	6,434404E-26	2,578112E-13
Promedio	2,470121E-29	2,603632E-15	1,172105E-26	7,831464E-14
Desviación	0,000000E+00	1,233430E-15	1,933567E-26	4,855177E-14
	F09		**F10**	
Mejor	0,000000E+00	2,770665E-16	0,000000E+00	3,252002E+01
Mediana	1,064336E-26	1,124608E-08	1,602116E-24	3,328903E+01
Peor	7,701479E-22	1,052759E+02	6,547005E+00	3,463243E+01
Promedio	6,117230E-23	1,072140E+01	2,618802E-01	3,326175E+01
Desviación	1,827553E-22	2,821923E+01	1,309401E+00	4,545577E-01
	F11		**F12**	
Mejor	-3,923426E-04	-3,268462E-04	-1,992634E-01	-1,991453E-01
Mediana	-3,923406E-04	-2,843296E-04	-1,992634E-01	5,337125E+02
Peor	-3,923369E-04	-2,236338E-04	-1,973228E-01	5,461723E+02
Promedio	-3,923404E-04	-2,863882E-04	-1,991622E-01	3,562330E+02
Desviación	1,479352E-09	2,707605E-05	3,889318E-04	2,889253E+02
	F13		**F14**	
Mejor	-6,503705E+01	-6,642473E+01	0,000000E+00	5,015863E-14
Mediana	-5,942728E+01	-6,531507E+01	2,222613E-24	1,359306E-13
Peor	-2,538647E+01	-6,429690E+01	5,575369E-15	2,923513E-12
Promedio	-5,802046E+01	-6,535310E+01	2,230154E-16	3,089407E-13
Desviación	7,850674E+00	5,733005E-01	1,115074E-15	5,608409E-13

Tabla 7.6: Resultados de los experimentos para 30D con las función F01 a F14.

	F15		**F16**	
	ED+HC3	**εDEag**	**ED+HC3**	**εDEag**
Mejor	2,150618E+00	2,160345E+01	1,027481E+00	0,000000E+00
Mediana	2,160334E+01	2,160375E+01	1,121996E+00	0,000000E+00
Peor	2,160409E+01	2,160403E+01	1,248617E+00	5,421011E-20
Promedio	2,082529E+01	2,160376E+01	1,121447E+00	2,168404E-21
Desviación	3,890557E+00	1,104834E-04	5,412224E-02	1,062297E-20
	F17		**F18**	
Mejor	6,064842E-03	2,165719E-01	5,527208E-05	1,226054E+00
Mediana	1,538058E+00	5,315949E+00	1,452524E-02	2,679497E+01
Peor	4,488074E+00	1,889064E+01	6,726466E-02	7,375363E+02
Promedio	1,193312E+00	6,326487E+00	2,309097E-02	8,754569E+01
Desviación	1,151180E+00	4,986691E+00	2,133382E-02	1,664753E+02

Tabla 7.7: Resultados de los experimentos para 30D con las función F15 a F18.

valor pequeño. Para ver la calidad de los resultados obtenidos por ED+HC3 se realiza la comparación contra εDEag, de Takahama y Sakai. Para 10D se observa que ED+HC es superior a εDEag 27 veces y se logran 36 empates, sobre un total de 90 comparaciones, lo que equipara los rendimientos de ambos algoritmos; mientras que para 30D la diferencia es notoria a favor del nuevo algoritmo, 58 veces es superior y no se obtienen empates. Esta propuesta de modificación a ED presenta mejores resultados que las versiones anteriores puesto que hay mejoras en el test para 10D y se obtienen casi los mismos valores para 30D que con ED+HC2.

Capítulo 8

Conclusiones Generales

En los tres capítulos anteriores se presentaron propuestas de modificaciones a ED con el fin de mejorar su desempeño. En ED+HC se transformó a ED en un algoritmo híbrido, dándole la capacidad de realizar búsquedas locales sobre el mejor individuo en cada *epoch*. Como segunda propuesta, ED+HC2, se agregó un nuevo operador para agregar diversidad poblacional con una fuerte preferencia por los individuos factibles, la mutación selectiva. Finalmente, sobre la base de ED+HC2 se generó ED+HC3 al utilizar una propuesta de selección con preferencia sobre los individuos factibles, en conjunto con un nuevo operador cuyo objetivo es mejorar la factiblidad de la población, tratando de no disminuir la diversidad poblacional.

A modo de resumen se analizó en detalle el desempeño, con las tres propuestas, sobre cuatro de las funciones optimizadas puesto que el comportamiento con las demás funciones es muy similar. Esto pone en evidencia las mejoras logradas a través de las diferentes modificaciones a ED. Las funciones son:

- **F02 sobre 10D**: Min $f(\mathbf{x}) = \text{máx}(\mathbf{z})$, con $\mathbf{z} = \mathbf{x} - \mathbf{o}$, $\mathbf{y} = \mathbf{z} - 0{,}5$
  - $g_1(\mathbf{x}) = 10 - \frac{1}{10}\sum_{i=1}^{10}\left[z_i^2 - 10\cos(2\pi z_i) + 10\right] \leqslant 0$
  - $g_2(\mathbf{x}) = \frac{1}{10}\sum_{i=1}^{10}\left[z_i^2 - 10\cos(2\pi z_i) + 10\right] - 15 \leqslant 0$
  - $h_1(\mathbf{x}) = \frac{1}{10}\sum_{i=1}^{10}\left[y_i^2 - 10\cos(2\pi y_i) + 10\right] - 20 = 0$
  - $\mathbf{x} \in [-5{,}12; 5{,}12]^{10}$
- **F07 sobre 30D**: Min $f(\mathbf{x}) = \sum_{i=1}^{29}\left[100\left(z_i^2 - z_{i+1}\right)^2 + \left(z_i - 1\right)^2\right]$, con $\mathbf{z} = \mathbf{x} + 1 - \mathbf{o}$, $\mathbf{y} = \mathbf{x} - \mathbf{o}$
  - $g_1(\mathbf{x}) = 0{,}5 - \exp\left(-0{,}1\sqrt{\frac{1}{30}\sum_{i=1}^{30} y_i^2}\right) - 3\exp\left(\frac{1}{30}\sum_{i=1}^{30}\cos(0{,}1y_i)\right) + \exp(1) \leqslant 0$
  - $\mathbf{x} \in [-140; 140]^{30}$
- **F10 sobre 10D**: Min $f(\mathbf{x}) = \sum_{i=1}^{9}\left[100\left(z_i^2 - z_{i+1}\right)^2 + \left(z_i - 1\right)^2\right]$, con $\mathbf{z} = \mathbf{x} + 1 - \mathbf{o}$, $\mathbf{y} = (\mathbf{x} - \mathbf{o})\mathbf{M}$
  - $h_1(\mathbf{x}) = \sum_{i=1}^{10}\left[y_i \sin\left(\sqrt{|y_i|}\right)\right] = 0$

Propuesta	tiempo (seg)	*fitness*	desviación
ED+HC	14.8571	-2.024122E-01	1.124E-16
ED+HC2	19.6161	-2.267465E+00	3.442E-05
ED+HC3	21.4205	-2.277041E+00	5.357E-06

Tabla 8.1: Comparativa de los experimentos para 10D con F02.

- $\mathbf{x} \in [-500; 500]^{10}$

- **F18 sobre 30D**: Min $f(\mathbf{x}) = \sum_{i=1}^{29} [z_i - z_{i+1}]^2$, con $\mathbf{z} = \mathbf{x} - \mathbf{o}$
  - $g_1(\mathbf{x}) = \frac{1}{30} \sum_{i=1}^{30} \left[-z_i \sin\left(\sqrt{|z_i|}\right) \right] \leqslant 0$
  - $h_1(\mathbf{x}) = \frac{1}{30} \sum_{i=1}^{30} \left[z_i \sin\left(\sqrt{|z_i|}\right) \right] = 0$
  - $\mathbf{x} \in [-50; 50]^{30}$

siendo $\mathbf{o}$ y $\mathbf{M}$ vector y matriz de rotación respectivamente, introducidos por los creadores del *Benchmark* con el objetivo de no favorecer el desempeño de los algoritmos que realizan trasformaciones lineales. Cada función posee sus propias constantes de rotación. El resto de las características de las funciones del *Benchmark* puede consultarse en el apéndice que se incluye al final de esta tesis.

8.1. Comparación de desempeños

8.1.1. F02 sobre 10D

La función $F02$ se define en el intervalo $[-5{,}12; 5{,}12]$ para las 10 variables que la componen, es una función separable con tres restricciones. Para este caso, las restricciones de desigualdad son separables mientras que la restricción de igualdad es no separable.

De las ejecuciones mostradas en la tablas 5.1, 6.5 y 7.4, se seleccionaron tres al azar por cada propuesta. Esta muestra del desempeño de las tres propuestas se visualiza en la tabla 8.1. Si bien la población final obtenida en ED+HC es la más compacta de las tres mostradas, hay una diferencia de un orden de magnitud con respecto a los resultados obtenidos luego de la evolución completa. En la figura 8.1 se muestra el ajuste del mejor individuo de cada propuesta, a lo largo de la evolución.

8.1.2. F07 sobre 30D

La función $F07$ se define en el intervalo $[-140; 140]$ para las 30 variables que la componen, es una función no separable con una sola restricción de desigualdad, separable..

De las ejecuciones mostradas en la tablas 5.3, 6.7 y 7.6, se seleccionaron tres al azar por cada propuesta. Esta muestra del desempeño de las tres propuestas se visualiza en la tabla 8.2. Todas las poblaciones finales son compactas, con el mismo resultado final para ED+HC2 y ED+HC3. Es de notar que ED+HC3 logró la convergencia en forma más rápida que ED+HC2, esto se aprecia en la figura 8.2, que muestra el ajuste del mejor individuo para cada una de las propuestas.

Propuesta	tiempo (seg)	*fitness*	desviación
ED+HC	35.1706	8.725793E-26	3.186E-26
ED+HC2	46.4537	2.470121E-29	0.000E+00
ED+HC3	56.1126	2.470121E-29	0.000E+00

Tabla 8.2: Comparativa de los experimentos para 30D con F07.

Propuesta	tiempo (seg)	*fitness*	desviación
ED+HC	15.7966	41.72588E+00	2.136E-12
ED+HC2	20.4985	41.72588E+00	1.118E-12
ED+HC3	20.6709	0.00000E+00	0.000E+00

Tabla 8.3: Comparativa de los experimentos para 10D con F10.

8.1.3. F10 sobre 10D

La función $F10$ se define en el intervalo $[-500; 500]$ para las 10 variables que la componen, es una función no separable con una única restricción de igualdad, que está rotada con respecto a los ejes.

De las ejecuciones mostradas en la tablas 5.2, 6.6 y 7.5, se seleccionaron tres al azar por cada propuesta. Esta muestra del desempeño de las tres propuestas se visualiza en la tabla 8.3. El resultado final obtenido con ED+HC y ED+HC2 es muy parecido, difiriendo sólo en la desviación estándar, aunque en muy poca medida. Es de notar que ED+HC3 logra una población con un único valor para todos los individuos. La figura 8.3 muestra el ajuste del mejor individuo para cada una de las propuestas. La línea que representa el desempeño de ED+HC3 se muestra incompleta alrededor de las 1700 *epochs*, puesto que alcanzó el valor cero antes terminar la evolución y la escala vertical es logarítmica.

8.1.4. F18 sobre 30D

La función $F13$ se define en el intervalo $[-50; 50]$ para las 30 variables que la componen, es una función no separable con una restricción de igualdad y una restricción de desigualdad, ambas separables.

De las ejecuciones mostradas en la tablas 5.4, 6.8 y 7.7, se seleccionaron tres al azar por cada propuesta. Esta muestra del desempeño de las tres propuestas se visualiza en la tabla 8.4. En la figura 8.4 claramente se observa la velocidad de convergencia y la capacidad de optimización que posee ED+HC3.

8.2. Análisis de complejidad

Las tres propuestas presentadas en esta tesis operan sobre la misma base, el algoritmo de Evolución Diferencial. Desde el punto de vista de la complejidad algorítmica la principal

Propuesta	tiempo (seg)	*fitness*	desviación
ED+HC	37.1738	12.94873E+00	1.318E-03
ED+HC2	48.6586	3.82607E-02	4.816E-05
ED+HC3	61.3180	1.28071E-04	3.336E-07

Tabla 8.4: Comparativa de los experimentos para 30D con F18.

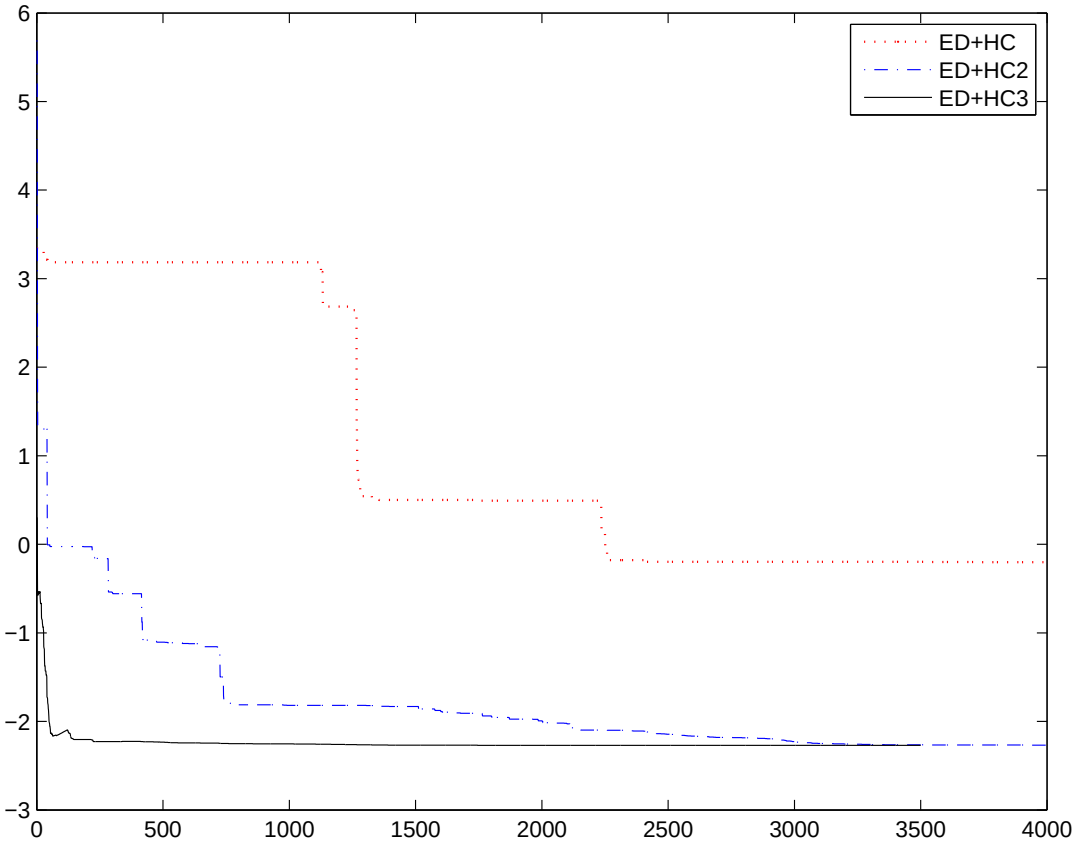

Figura 8.1: Mejor individuo de cada propuesta para F02.

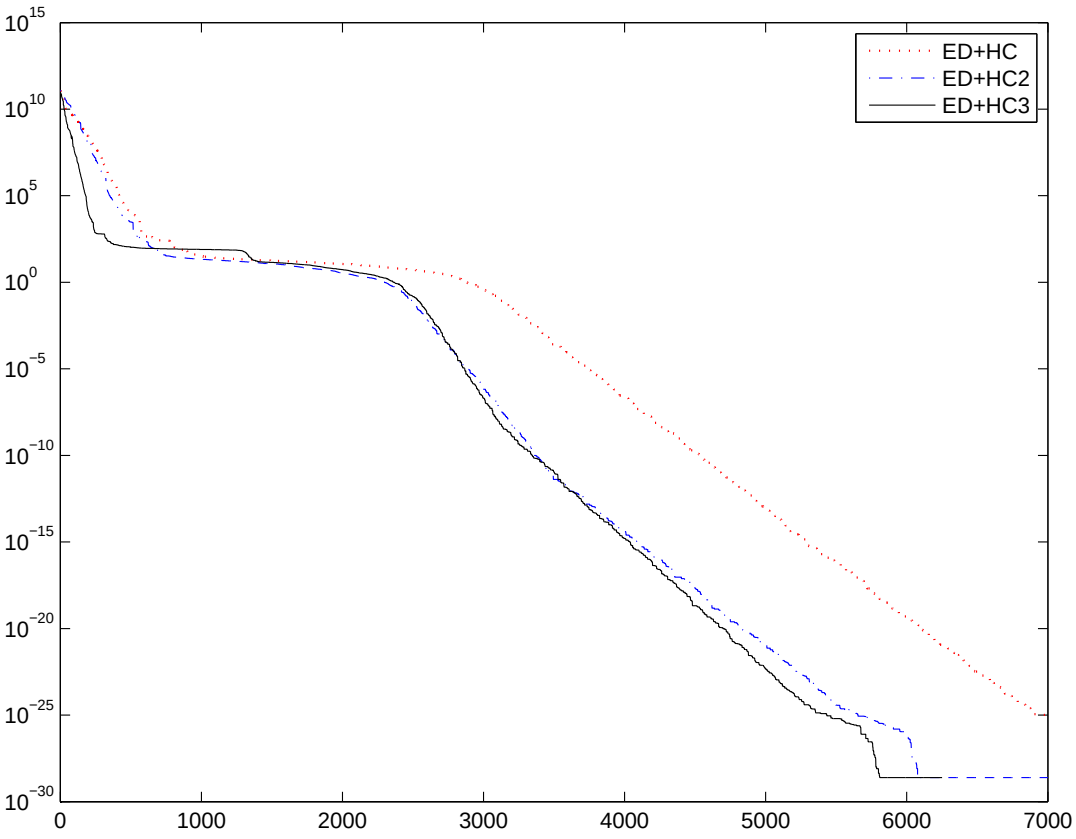

Figura 8.2: Mejor individuo de cada propuesta para F07.

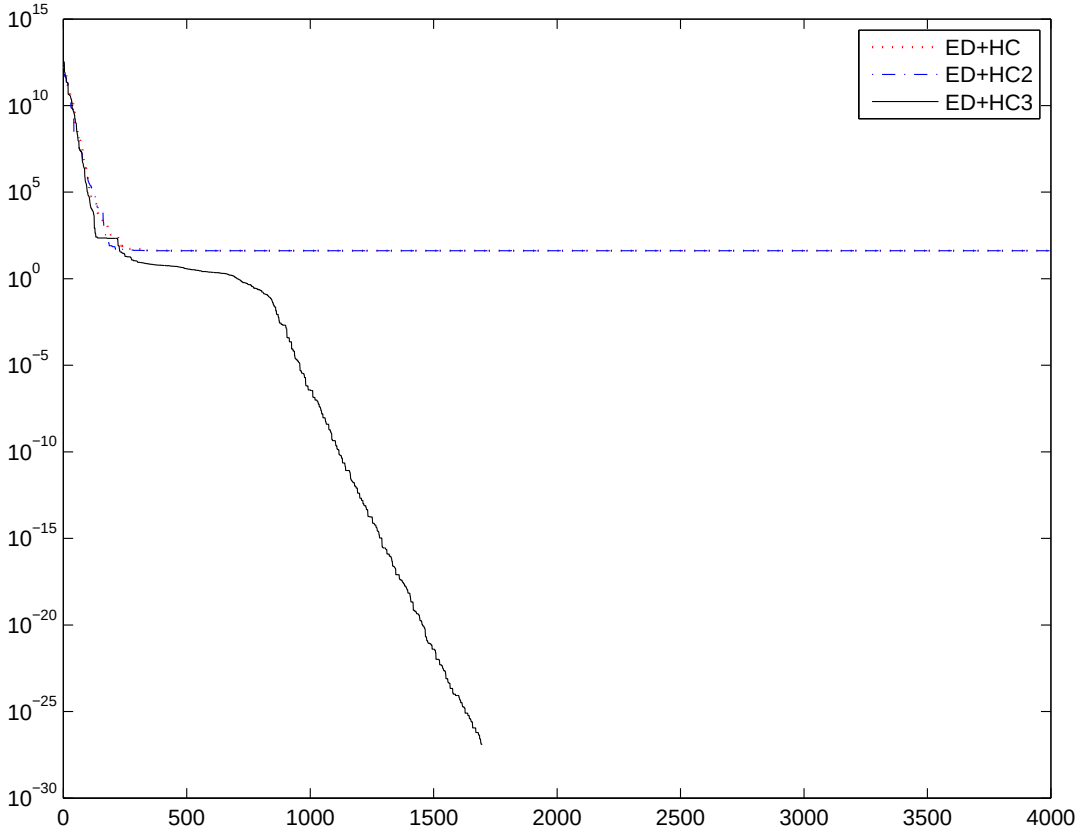

Figura 8.3: Mejor individuo de cada propuesta para F10.

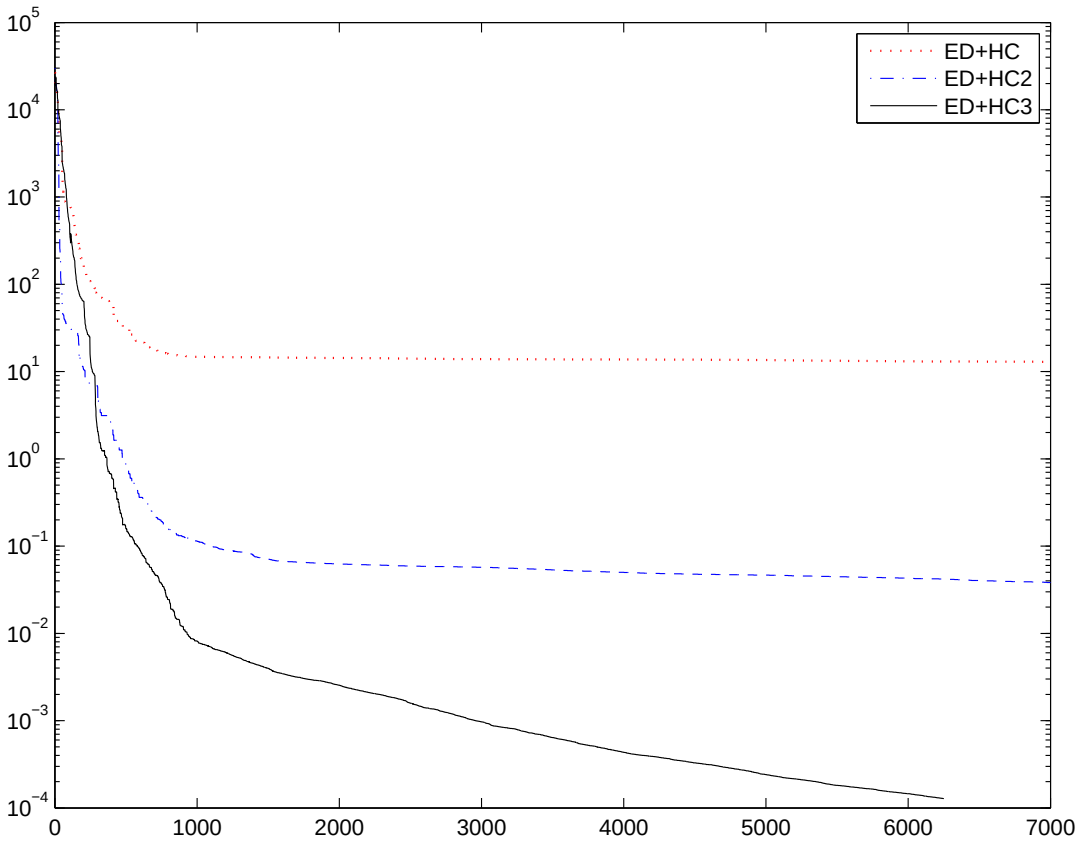

Figura 8.4: Mejor individuo de cada propuesta para F18.

	ED+HC		**ED+HC2**		**ED+HC3**	
dim	$\overline{\mathbf{t}}$(seg)	σ_t	$\overline{\mathbf{t}}$(seg)	σ_t	$\overline{\mathbf{t}}$(seg)	σ_t
10D	25,736	6,26E-02	33,734	1,14E-01	36,087	1,53E-01
15D	29,428	1,19E-01	37,885	4,28E-02	38,951	5,05E-02
20D	35,152	2,17E-01	43,002	1,60E-02	43,811	6,53E-02
25D	39,804	2,10E-01	46,652	3,70E-02	48,726	1,77E-01
30D	48,132	2,66E-01	50,914	3,75E-01	52,545	9,78E-02

Tabla 8.5: Tiempos de ejecución para $F01$

diferencia entre el tiempo de cómputo para la optimización de un problema y otra depende de la estructura de la función objetivo y sus restricciones, puesto que se aplican los mismos operadores en toda ejecución y las comparaciones booleanas (*fitness* padre vs *fitness* hijo, *random* vs probabilidad de cruzamiento) no implican mayor cantidad de operaciones por ser **V** o **F**.

En cambio, para las propuestas híbridas hay diferencias en la cantidad de operaciones de punto flotante puesto que se aplican operadores nuevos, como HCMod que se aplica en los tres algoritmos presentados, o Reparación que es exclusivo de ED+HC3. A continuación se realizará un breve cálculo de la complejidad temporal *a posteriori*.

Cada una de las propuestas fue ejecutada 25 veces con los siguientes parámetros comunes:

- Tamaño de la población: 50 individuos.
- Tiempo de evolución: 5000 *epochs*.
- Probabilidad de cruzamiento: 0,6.
- Factor de escala en mutación: 0,6.
- Tamaño del problema: 10D, 15D, 20D, 25D, 30D.
- Exploraciones por HCMod: $\lfloor 0{,}3 \times dim \rfloor$.
- Variables exploradas por HCMod en cada *epoch*: $\lfloor 0{,}3 \times dim \rfloor$.
- Penalidad: estática de factor 50.

Además, se tuvo en cuenta que:

- Se aplicó mutación selectiva en las primeras 2500 *epochs* para ED+HC2 y ED+HC3, luego mutación tradicional.
- Se realizaron $\lfloor 0{,}3 \times dim \rfloor$ intentos de reparación en ED+HC3 en cada *epoch*.

La tendencia de los tiempos de ejecución con los parámetros declarados en las 18 funciones del *suite* son similares, por lo que se muestran resultados sólo de $F01$ y $F13$ en las tablas 8.5 y 8.6, donde $\overline{\mathbf{t}}$ es el tiempo promedio, medido en segundos, de ejecución de cada propuesta y σ_t es el valor de la desviación estándar de los tiempos.

Por la forma en que evolucionan los tiempos de ejecución, es posible que se esté ante un algoritmo cuya complejidad temporal sea $\mathcal{O}(n)$, ya que los datos tabulados siguen la tendencia de un polinomio lineal. Esto se confirmó al calcular la recta de ajuste y el correspondiente índice de determinación, que devuelve valores muy cercanos a 1. Las rectas de regresión y los valores de los índices son los siguientes:

	ED+HC		ED+HC2		ED+HC3	
dim	$\bar{\mathbf{t}}$(seg)	σ_t	$\bar{\mathbf{t}}$(seg)	σ_t	$\bar{\mathbf{t}}$(seg)	σ_t
10D	27,062	5,27E-02	34,948	1,44E-01	35,708	1,21E-01
15D	30,318	1,11E-01	37,809	2,24E-02	38,530	9,69E-02
20D	35,406	4,21E-02	42,712	8,49E-02	43,712	3,17E-01
25D	40,742	2,63E-02	47,740	1,23E-01	48,382	1,20E-01
30D	44,289	6,22E-02	51,154	1,71E-01	52,006	2,27E-01

Tabla 8.6: Tiempos de ejecución para $F13$

- **F01**
  - ED+HC: $p_1(t) = 1{,}10336t + 13{,}5832$, $I_d = 0{,}9808$
  - ED+HC2: $p_1(t) = 0{,}86254t + 25{,}1866$, $I_d = 0{,}9977$
  - ED+HC3: $p_1(t) = 0{,}85382t + 26{,}9476$, $I_d = 0{,}9932$
- **F13**
  - ED+HC: $p_1(t) = 0{,}89756t + 17{,}6122$, $I_d = 0{,}9932$
  - ED+HC2: $p_1(t) = 0{,}84686t + 25{,}9354$, $I_d = 0{,}9920$
  - ED+HC3: $p_1(t) = 0{,}84896t + 26{,}6884$, $I_d = 0{,}9927$

En las figuras 8.5 y 8.6 se muestra la información detallada anteriormente. Con color gris se representó el conjunto de datos de ED+HC, con color rojo el de ED+HC2 y con color azul el de ED+HC3.

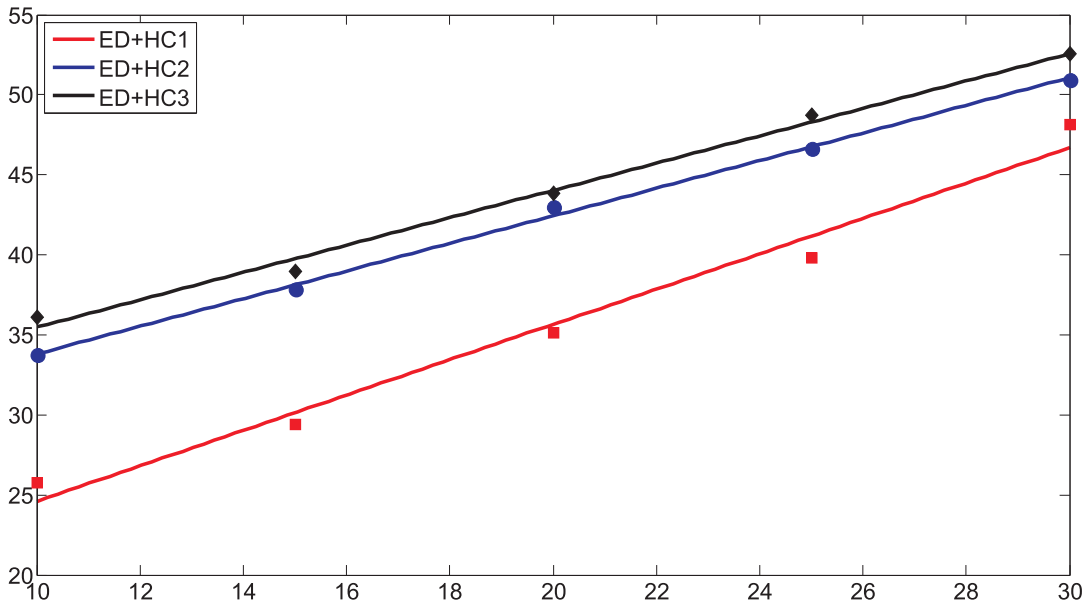

Figura 8.5: Análisis de regresión de los tiempos para F01.

8.3. Desempeño general

El benchmark de la sesión especial del CEC2010 utilizado para mostrar el desempeño de las propuestas de esta tesis, incluye 18 funciones en 10D y 30D, y de cada una de las

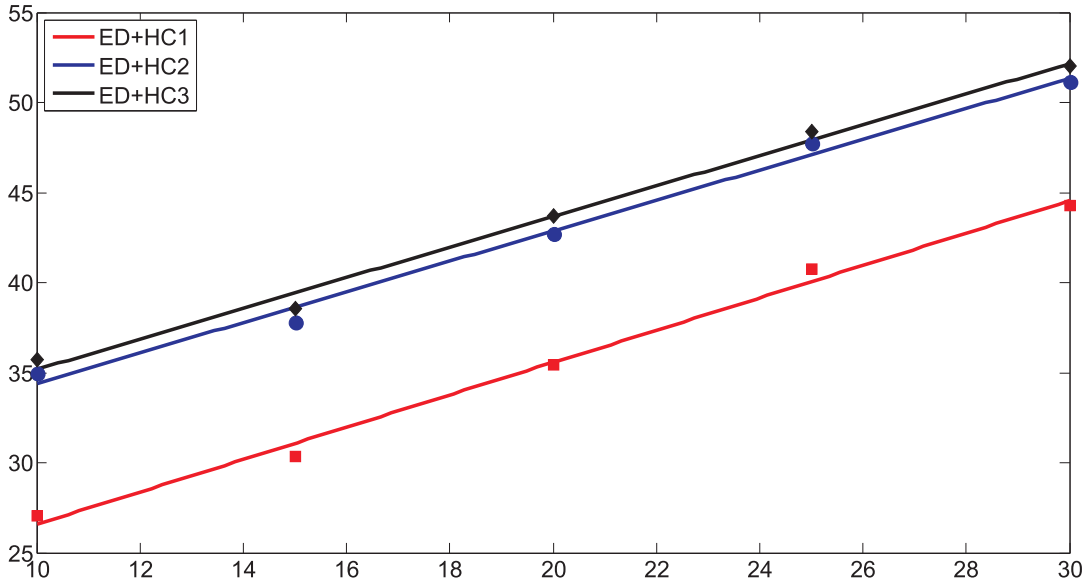

Figura 8.6: Análisis de regresión de los tiempos para F13.

funciones se obtuvo información sobre los individuos *mejor*, *mediana*, *peor*, y los estadísticos *promedio* y *desviación*. En los capítulos 5, 6 y 7 se presentan tablas sobre el desempeño de cada propuesta contra los resultados obtenidos por el algoritmo ganador de la sesión especial, εDEag. En las tablas 8.7 y 8.8 se identifica el algoritmo que obtuvo el mejor valor en cada uno de los ítems comparados, donde 1 representa a ED+HC, 2 representa a ED+HC2, 3 representa a ED+HC3 y ε representa a εDEag. De la comparación general se muestra que:

- Los cuatro algoritmos obtienen el mismo valor en 22 instancias.
- El algoritmo ED+HC obtiene el mejor valor 27 veces, 20 de ellas en forma exclusiva.
- El algoritmo ED+HC2 obtiene el mejor valor 51 veces, 32 de ellas en forma exclusiva.
- El algoritmo ED+HC3 obtiene el mejor valor 66 veces, 36 de ellas en forma exclusiva.
- El algoritmo εDEag obtiene el mejor valor 54 veces, 39 de ellas en forma exclusiva.

8.4. Comentarios finales

A lo largo de esta tesis se explicitaron las principales características y fundamentos de Evolución Diferencial, se indagó sobre el manejo de restricciones y el estado del arte de ED, incluyendo referencias a modificaciones de los operadores básicos, hibridaciones y aplicaciones en las cuales ED se aplica como el método principal de optimización. Luego se hicieron tres propuestas: ED+HC, donde se hibridó ED con un buscador local; ED+HC2, presentación de un nuevo operador de mutación selectiva; y ED+HC3, que se ejecutó con las bases de las dos propuestas anteriores y se anexó un operador de selección por torneo y se aplicó el nuevo operador de reparación.

Los parámetros de *tamaño de población* y *tiempo de evolución* tomaron los mismos valores para ED+HC y ED+HC2. Sin embargo, los experimentos mostraron que ED+HC3 posee una capacidad mayor de explotación, mientras la calidad de exploración se mantiene con respecto a las propuestas iniciales. Por este motivo se disminuyó el tiempo de evolución, y con el fin de mantener constante la cantidad de *evaluaciones de función*, se aumentó el

Función	Mejor	Mediana	Peor	Promedio	Desviación
F01	Empate	1, 3, ε	ε	ε	ε
F02	3	3	2	2	2
F03	Empate	2, 3, ε	2, 3, ε	2, 3, ε	2, 3, ε
F04	1	1	ε	ε	ε
F05	2, 3, ε	3, ε	ε	ε	ε
F06	2	2	2	2	ε
F07	Empate	Empate	Empate	Empate	Empate
F08	Empate	1, 2, ε	1, 2, 3	3	1
F09	Empate	Empate	3, ε	3, ε	3, ε
F10	Empate	3, ε	3, ε	3, ε	3, ε
F11	Empate	Empate	Empate	Empate	2
F12	2	ε	2	ε	3
F13	Empate	1	ε	ε	ε
F14	Empate	Empate	Empate	Empate	Empate
F15	Empate	ε	3	ε	ε
F16	ε	ε	ε	ε	1
F17	2, 3	3	3	3	3
F18	1, 3	1, 3	3	3	3

Tabla 8.7: Mejores resultados de los experimentos para 10D.

Función	Mejor	Mediana	Peor	Promedio	Desviación
F01	2	ε	ε	ε	ε
F02	3	3	3	3	ε
F03	2, 3	2	2	2	2
F04	1	1	1	1	1
F05	3	1	1	1	ε
F06	3	1	1	1	ε
F07	1, 2, 3	2, 3	2, 3	2, 3	2, 3
F08	2, 3	2, 3	2	2	2
F09	2, 3	2	3	3	3
F10	3	3	3	3	ε
F11	2, 3	3	3	3	3
F12	2	2	2	2	3
F13	1	ε	ε	ε	ε
F14	1, 2, 3	2	2	2	2
F15	3	1	1	3	1
F16	ε	ε	ε	ε	ε
F17	2	2	2	2	2
F18	3	3	3	3	3

Tabla 8.8: Mejores resultados de los experimentos para 30D.

tamaño de la población. A pesar de esto, que puede suponer mayor exploración pero menor explotación debido al menor tiempo de evolución, el resultado presentado superó al obtenido por las propuestas ED+HC y ED+HC2.

Si bien el tiempo de cómputo es mayor a medida de que avanzan las propuestas, esto era esperable. En cada una se agregan mayores condiciones de operación y selección, operadores nuevos y mayor verificación de condiciones lógicas.

Las tres propuestas presentadas representan mejoras con respecto al desempeño del algoritmo ED original. También se lograron importantes mejoras con respecto a uno de los algoritmos del estado del arte, cuyos autores son Takahama y Sakai [52], lo que ubica a estas nuevas propuesas como referentes también del estado del arte. El desempeño de ED+HC3 aún no fue publicado, pero se está elaborando un artículo como producto de los resultados del Capítulo 7, que será posiblemente enviado a una revista de divulgación científica.

Apéndice A

Suite de Funciones del CEC2010

Las funciones que se presentan a continuación fueron escogidas para la sesión especial *CEC2010 Competition on Constrained Real-Parameter Optimization*, [29]. Se utilizó como base para este *benchmark* a funciones utilizadas en forma habitual para probar algoritmos de optimización, a las que se agregaron constantes ($\mathbf{o}$ y $\mathbf{M}$) con el fin de efectuar rotaciones sobre el espacio de búsqueda y no favorecer el desempeño de los algoritmos de búsqueda lineal.

Para cada una de las 18 funciones se detalla su estructura, las restricciones, el espacio de búsqueda y el factor ρ, que es la aproximación de la proporción entre espacio factible y espacio de búsqueda.

F01

$$\text{Min } f(\mathbf{x}) = -\left|\frac{\sum_{i=1}^{D}\cos^4(z_i) - 2\prod_{i=1}^{D}\cos^2(z_i)}{\sqrt{\sum_{i=1}^{D} i z_i^2}}\right|, \text{ donde } \mathbf{z} = \mathbf{x} - \mathbf{o},$$

- $g_1(\mathbf{x}) = 0{,}75 - \prod_{i=1}^{D} z_i \leqslant 0$
- $g_2(\mathbf{x}) = \sum_{i=1}^{D} z_i - 7{,}5D \leqslant 0$
- $\mathbf{x} \in [0; 10]^D$

Es una función no separable, ambas restricciones son no separables y $\rho_{10D} \approx 0{,}997689$, $\rho_{30D} \approx 1{,}000000$.

F02

$$\text{Min } f(\mathbf{x}) = \text{máx}(\mathbf{z}), \text{ donde } \mathbf{z} = \mathbf{x} - \mathbf{o} \text{ y } \mathbf{y} = \mathbf{z} - 0{,}5,$$

- $g_1(\mathbf{x}) = 10 - \frac{1}{D}\sum_{i=1}^{D}\left[z_i^2 - 10\cos(2\pi z_i) + 10\right] \leqslant 0$
- $g_2(\mathbf{x}) = \frac{1}{D}\sum_{i=1}^{D}\left[z_i^2 - 10\cos(2\pi z_i) + 10\right] - 15 \leqslant 0$

- $h(\mathbf{x}) = \frac{1}{D}\sum_{i=1}^{D}\left[y_i^2 - 10\cos(2\pi y_i) + 10\right] - 20 = 0$
- $\mathbf{x} \in [-5{,}12; 5{,}12]^D$

Es una función separable, sus tres restricciones son separables y $\rho_{10D} \approx 0{,}000000$, $\rho_{30D} \approx 0{,}000000$.

F03

Min $f(\mathbf{x}) = \sum_{i=1}^{D-1}\left[100\left(z_i^2 - z_{i+1}\right)^2 + (z_i - 1)^2\right]$, donde $\mathbf{z} = \mathbf{x} - \mathbf{o}$,

- $h(\mathbf{x}) = \sum_{i=1}^{D-1}\left[z_i - z_{i+1}\right]^2 = 0$
- $\mathbf{x} \in [-1000; 1000]^D$

Es una función no separable, su restricción es no separable y $\rho_{10D} \approx 0{,}000000$, $\rho_{30D} \approx 0{,}000000$.

F04

Min $f(\mathbf{x}) = \text{máx}(\mathbf{z})$, donde $\mathbf{z} = \mathbf{x} - \mathbf{o}$,

- $h_1(\mathbf{x}) = \frac{1}{D}\sum_{i=1}^{D}\left[z_i \cos\left(\sqrt{|z_i|}\right)\right] = 0$
- $h_2(\mathbf{x}) = \sum_{i=1}^{D/2-1}\left[z_i - z_{i+1}\right]^2 = 0$
- $h_3(\mathbf{x}) = \sum_{i=D/2+1}^{D-1}\left[z_i^2 - z_{i+1}\right]^2 = 0$
- $h_4(\mathbf{x}) = \sum_{i=1}^{D} z_i = 0$
- $\mathbf{x} \in [-50; 50]^D$

Es una función separable, dos restricciones son no separables, las otras dos son separables y $\rho_{10D} \approx 0{,}000000$, $\rho_{30D} \approx 0{,}000000$.

F05

Min $f(\mathbf{x}) = \text{máx}(\mathbf{z})$, donde $\mathbf{z} = \mathbf{x} - \mathbf{o}$,

- $h_1(\mathbf{x}) = \frac{1}{D}\sum_{i=1}^{D}\left[-z_i \sin\left(\sqrt{|z_i|}\right)\right] = 0$
- $h_2(\mathbf{x}) = \frac{1}{D}\sum_{i=1}^{D}\left[-z_i \cos\left(0{,}5\sqrt{|z_i|}\right)\right] = 0$
- $\mathbf{x} \in [-600; 600]^D$

Es una función separable, sus restricciones son separables y $\rho_{10D} \approx 0{,}000000$, $\rho_{30D} \approx 0{,}000000$.

F06

Min $f(\mathbf{x}) = \text{máx}(\mathbf{z})$, donde $\mathbf{z} = \mathbf{x} - \mathbf{o}$, $\mathbf{y} = (\mathbf{x} + 483{,}6106156535 - \mathbf{o})\,\mathbf{M} - 483{,}6106156535$,

- $h_1(\mathbf{x}) = \frac{1}{D}\sum_{i=1}^{D}\left[-y_i \sin\left(\sqrt{|y_i|}\right)\right] = 0$
- $h_2(\mathbf{x}) = \frac{1}{D}\sum_{i=1}^{D}\left[-y_i \cos\left(0{,}5\sqrt{|y_i|}\right)\right] = 0$
- $\mathbf{x} \in [-600; 600]^D$

Es una función separable, sus restricciones son rotaciones y $\rho_{10D} \approx 0{,}000000$, $\rho_{30D} \approx 0{,}000000$.

F07

Min $f(\mathbf{x}) = \sum_{i=1}^{D-1}\left[100\left(z_i^2 - z_{i+1}\right)^2 + (z_i - 1)^2\right]$, donde $\mathbf{z} = \mathbf{x} + 1 - \mathbf{o}$ y $\mathbf{y} = \mathbf{x} - \mathbf{o}$,

- $g(\mathbf{x}) = 0{,}5 - \exp\left(-0{,}1\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}\right) - 3\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(0{,}1y_i)\right) + \exp(1) \leqslant 0$
- $\mathbf{x} \in [-140; 140]^D$

Es una función no separable, su restricción es separables y $\rho_{10D} \approx 0{,}505123$, $\rho_{30D} \approx 0{,}503725$.

F08

Min $f(\mathbf{x}) = \sum_{i=1}^{D-1}\left[100\left(z_i^2 - z_{i+1}\right)^2 + (z_i - 1)^2\right]$, donde $\mathbf{z} = \mathbf{x} + 1 - \mathbf{o}$, $\mathbf{y} = (\mathbf{x} - \mathbf{o})\,\mathbf{M}$,

- $g(\mathbf{x}) = 0{,}5 - \exp\left(-0{,}1\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}\right) - 3\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(0{,}1y_i)\right) + \exp(1) \leqslant 0$
- $\mathbf{x} \in [-140; 140]^D$

Es una función no separable, su restricción es una rotación y $\rho_{10D} \approx 0{,}379512$, $\rho_{30D} \approx 0{,}375278$.

F09

Min $f(\mathbf{x}) = \sum_{i=1}^{D-1}\left[100\left(z_i^2 - z_{i+1}\right)^2 + (z_i - 1)^2\right]$, donde $\mathbf{z} = \mathbf{x} + 1 - \mathbf{o}$, $\mathbf{y} = \mathbf{x} - \mathbf{o}$,

- $h(\mathbf{x}) = \sum_{i=1}^{D}\left[y_i \sin\left(\sqrt{|y_i|}\right)\right] = 0$
- $\mathbf{x} \in [-500; 500]^D$

Es una función no separable, su restricción es separable y $\rho_{10D} \approx 0{,}000000$, $\rho_{30D} \approx 0{,}000000$.

F10

Min $f(\mathbf{x}) = \sum_{i=1}^{D-1} \left[100\left(z_i^2 - z_{i+1}\right)^2 + (z_i - 1)^2\right]$, donde $\mathbf{z} = \mathbf{x} + 1 - \mathbf{o}$ y $\mathbf{y} = (\mathbf{x} - \mathbf{o})\mathbf{M}$,

- $h(\mathbf{x}) = \sum_{i=1}^{D} \left[y_i \sin\left(\sqrt{|y_i|}\right)\right] = 0$
- $\mathbf{x} \in [-500; 500]^D$

Es una función no separable, su restricción es una rotación y $\rho_{10D} \approx 0{,}000000$, $\rho_{30D} \approx 0{,}000000$.

F11

Min $f(\mathbf{x}) = \frac{1}{D} \sum_{i=1}^{D} \left[-z_i \cos\left(2\sqrt{|z_i|}\right)\right]$, donde $\mathbf{z} = (\mathbf{x} - \mathbf{o})\,\mathbf{M}$, $\mathbf{y} = \mathbf{x} + 1 - \mathbf{o}$,

- $h(\mathbf{x}) = \sum_{i=1}^{D-1} \left[100\left(y_i^2 - y_{i+1}\right)^2 + (y_i - 1)^2\right] = 0$
- $\mathbf{x} \in [-100; 100]^D$

Es una función rotada, su restricción es no separable y $\rho_{10D} \approx 0{,}000000$, $\rho_{30D} \approx 0{,}000000$.

F12

Min $f(\mathbf{x}) = \sum_{i=1}^{D} \left[z_i \sin\left(\sqrt{|z_i|}\right)\right]$, donde $\mathbf{z} = \mathbf{x} - \mathbf{o}$,

- $g(\mathbf{x}) = \sum_{i=1}^{D} [z_i - 100\cos(0{,}1 z_i) + 10] \leqslant 0$
- $h(\mathbf{x}) = \sum_{i=1}^{D-1} \left[z_i^2 - z_{i+1}\right]^2 = 0$
- $\mathbf{x} \in [-1000; 1000]^D$

Es una función separable, una de sus restricciones es no separable, la otra restricción es separable y $\rho_{10D} \approx 0{,}000000$, $\rho_{30D} \approx 0{,}000000$.

F13

Min $f(\mathbf{x}) = \frac{1}{D} \sum_{i=1}^{D} \left[-z_i \sin\left(\sqrt{|z_i|}\right)\right]$, donde $\mathbf{z} = \mathbf{x} - \mathbf{o}$,

- $g_1(\mathbf{x}) = -50 + \frac{1}{100D} \sum_{i=1}^{D} z_i^2 \leqslant 0$
- $g_2(\mathbf{x}) = \frac{50}{D} \sum_{i=1}^{D} \sin\left(\frac{1}{50}\pi z_i\right) \leqslant 0$
- $g_3(\mathbf{x}) = 75 - 50\left(\sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1\right) \leqslant 0$

- $\mathbf{x} \in [-500; 500]^D$

Es una función separable, dos de sus restricciones son separables, la restante es no separable y $\rho_{10D} \approx 0{,}000000$, $\rho_{30D} \approx 0{,}000000$.

F14

Min $f(\mathbf{x}) = \sum_{i=1}^{D-1} \left[100\left(z_i^2 - z_{i+1}\right)^2 + (z_i - 1)^2\right]$, donde $\mathbf{z} = \mathbf{x} + 1 - \mathbf{o}$, $\mathbf{y} = \mathbf{x} - \mathbf{o}$,

- $g_1(\mathbf{x}) = \sum_{i=1}^{D} \left[-y_i \cos\left(\sqrt{|y_i|}\right)\right] - D \leqslant 0$
- $g_2(\mathbf{x}) = \sum_{i=1}^{D} \left[y_i \cos\left(\sqrt{|y_i|}\right)\right] - D \leqslant 0$
- $g_3(\mathbf{x}) = \sum_{i=1}^{D} \left[y_i \sin\left(\sqrt{|y_i|}\right)\right] - 10D \leqslant 0$
- $\mathbf{x} \in [-1000; 1000]^D$

Es una función no separable, sus restricciones son separables y $\rho_{10D} \approx 0{,}003112$, $\rho_{30D} \approx 0{,}006123$.

F15

Min $f(\mathbf{x}) = \sum_{i=1}^{D-1} \left[100\left(z_i^2 - z_{i+1}\right)^2 + (z_i - 1)^2\right]$, donde $\mathbf{z} = \mathbf{x} + 1 - \mathbf{o}$, $\mathbf{y} = (\mathbf{x} - \mathbf{o})\,\mathbf{M}$,

- $g_1(\mathbf{x}) = \sum_{i=1}^{D} \left[-y_i \cos\left(\sqrt{|y_i|}\right)\right] - D \leqslant 0$
- $g_2(\mathbf{x}) = \sum_{i=1}^{D} \left[y_i \cos\left(\sqrt{|y_i|}\right)\right] - D \leqslant 0$
- $g_3(\mathbf{x}) = \sum_{i=1}^{D} \left[y_i \sin\left(\sqrt{|y_i|}\right)\right] - 10D \leqslant 0$
- $\mathbf{x} \in [-1000; 1000]^D$

Es una función no separable, sus restricciones son rotaciones y $\rho_{10D} \approx 0{,}003210$, $\rho_{30D} \approx 0{,}006023$.

F16

Min $f(\mathbf{x}) = \sum_{i=!}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1$, donde $\mathbf{z} = \mathbf{x} - \mathbf{o}$,

- $g_1(\mathbf{x}) = \sum_{i=1}^{D} \left[z_i^2 - 100\cos(\pi z_i) + 10\right] \leqslant 0$
- $g_2(\mathbf{x}) = \prod_{i=1}^{D} z_i \leqslant 0$

- $h_1(\mathbf{x}) = \sum_{i=1}^{D} \left[z_i \sin\left(\sqrt{|z_i|}\right) \right] = 0$
- $h_2(\mathbf{x}) = \sum_{i=1}^{D} \left[-z_i \sin\left(\sqrt{|z_i|}\right) \right] = 0$
- $\mathbf{x} \in [-10; 10]^D$

Es una función no separable, una restricción es no separable, el resto de las restricciones son separables y $\rho_{10D} \approx 0{,}000000$, $\rho_{30D} \approx 0{,}000000$.

F17

Min $f(\mathbf{x}) = \sum_{i=1}^{D-1} [z_i - z_{i+1}]^2$, donde $\mathbf{z} = \mathbf{x} - \mathbf{o}$,

- $g_1(\mathbf{x}) = \prod_{i=1}^{D} z_i \leqslant 0$
- $g_2(\mathbf{x}) = \sum_{i=1}^{D} z_i \leqslant 0$
- $h(\mathbf{x}) = \sum_{i=!}^{D} \left[z_i \sin\left(4\sqrt{|z_i|}\right) \right] = 0$
- $\mathbf{x} \in [-10; 10]^D$

Es una función no separable, una restricción es separable, las dos restricciones restantes son no separables y $\rho_{10D} \approx 0{,}000000$, $\rho_{30D} \approx 0{,}000000$.

F18

Min $f(\mathbf{x}) = \sum_{i=1}^{D-1} [z_i - z_{i+1}]^2$, donde $\mathbf{z} = \mathbf{x} - \mathbf{o}$,

- $g(\mathbf{x}) = \frac{1}{D} \sum_{i=1}^{D} \left[-z_i \sin\left(\sqrt{|z_i|}\right) \right] \leqslant 0$
- $h(\mathbf{x}) = \frac{1}{D} \sum_{i=1}^{D} \left[z_i \sin\left(\sqrt{|z_i|}\right) \right] = 0$
- $\mathbf{x} \in [-50; 50]^D$

Es una función no separable, ambas restricciones son separables y $\rho_{10D} \approx 0{,}000010$, $\rho_{30D} \approx 0{,}000000$.

Bibliografía

[1] Baatar, N., Jeong, K.-Y., and Koh, C.-S. Adaptive parameter controlling non-dominated ranking differential evolution for multi-objective optimization of electromagnetic problems. *IEEE Transactions on Magnetics 50* (2015), 709–712.

[2] Coello Coello, C. A. Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry 41* (2000), 113–127.

[3] Coello Coello, C. A. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering 191* (2002), 1245–1287.

[4] Croes, G. A. A method for solving traveling-salesman problems. *Operations Research 6* (1958), 791–812.

[5] De Falco, I., Cioppa, A. D., Maisto, D., Scafuri, U., and Tarantino, E. An adaptive invasion-based model for distributed differential evolution. *Information Sciences 278* (2014), 653–672.

[6] Deb, A., Roy, J. S., and Gupta, B. Performance comparison of differential evolution, particle swarm optimization and genetic algorithm in the design of circularly polarized microstrip antennas. *IEEE Transactions on Antennas and Propagation 62* (2014), 3920–3928.

[7] Deb, K. An efficient constraint handling method for genetic algorithms. *Computer Methods in Appplied Mechanics and Engineering 186* (2000), 311–338.

[8] Deuflhard, P. *Newton Methods for Nonlinear Problems.* Springer, 2004.

[9] Domínguez-Isidro, S., Mezura-Montes, E., and Leguizamón, G. Memetic differential evolution for constrained numerical optimization problems. *IEEE Congress on Evolutionary Computation* (2013), 2996–3003.

[10] Eiben, A., and Smith, J. *Introduction to Evolutionary Computing.* Springer, 2003.

[11] Gong, W., and Cai, Z. Differential evolution with ranking-based mutation operators. *IEEE Transactions on Cybernetics 43* (2013), 2066–2081.

[12] Gong, W., Cai, Z., and Wang, Y. Repairing the crossover rate in adaptive differential evolution. *Applied Soft Computing 15* (2014), 149–168.

[13] Hadj-Alouane, A. B., and Bean, J. C. A genetic algorithm for the multiple-choice integer program. *Operations Research 45* (1997), 92–101.

[14] Hernández, S., Leguizamón, G., and Mezura-Montes, E. Hibridación de evolución diferencial utilizando hill climbing para resolver problemas de optimización con restricciones. *Congreso Argentino de Ciencias de la Computación* (2012), 60–69.

[15] Hernández, S., Leguizamón, G., and Mezura-Montes, E. A hybrid version of differential evolution with two differential mutation operators applied by stages. *IEEE Congress on Evolutionary Computation* (2013), 2895–2901.

[16] Holland, J. *Adaptation in Natural and Artificial Systems.* MIT Press, 1992.

[17] Homaifar, A., Qi, C. X., and Lai, S. H. Constrained optimization via genetic algorithms. *Simulation 62* (1994), 242–253.

[18] Hooke, R., and Jeeves, T. Direct search: solution of numerical and statistical problems. *Journal of the Association for Computing Machinery 8* (1961), 212–229.

[19] Jan, M. A., and Khanum, R. A. A study of two penalty-parameterless constraint handling techniques in the framework of moea/d. *Applied Soft Computing 13* (2013), 128–148.

[20] Jia, G., Wang, Y., Cai, Z., and Jin, Y. An improved $(\mu + \lambda)$-constrained differential evolution for constrained optimization. *Information Sciences 222* (2013), 302–322.

[21] Joines, J., and Houck, C. R. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's. *IEEE Conference on Evolutionary Computation* (1994), 579–584.

[22] Kazimipour, B., Li, X., and Qin, A. K. Effects of population initialization on differential evolution for large scale optimization. *IEEE Congress on Evolutionary Computation* (2014), 2404–2411.

[23] Kelley, C. *Solving Nonlinear Equations with Newton's Method.* SIAM, 2003.

[24] Kennedy, J., and Eberhart, R. Particle swarm optimization. *IEEE International Conference on Neural Networks 4* (1995), 1942–1948.

[25] Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science 220* (1983), 671–680.

[26] Li, X., and Yin, M. Parameter estimation for chaotic systems by hybrid differential evolution algorithm and artificial bee colony algorithm. *Nonlinear Dynamics 77* (2014), 61–71.

[27] Liepins, G. E., and Vose, M. D. Representational issues in genetic optimization. *Journal Of Experimental & Theoretical Artificial Intelligence 2* (1990), 101–115.

[28] Lourenço, H., Martin, O., and Stützle, T. Iterated local search. *Handbook of Metaheuristics 57* (2003), 321–353.

[29] Mallipeddi, R., and Suganthan, P. N. Problem definitions and evaluation criteria for the cec 2010 competition on constrained real parameter optimization. Tech. rep., Nanyang Technological University, Singapore, 2010.

[30] McMinn, P. Search-based software testing: Past, present and future. *IEEE Fourth International Conference on Software Testing* (2011), 153–163.

[31] Mezura-Montes, E., and Coello Coello, C. A. Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation 1*, 4 (2011), 173–194.

[32] Mezura-Montes, E., Velázquez-Reyes, J., and Coello Coello, C. A. A comparative study of differential evolution variants for global optimization. *Proceedings of the Genetic and Evolutionary Computation Conference 1* (2006), 485–492.

[33] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, 1996.

[34] Michalewicz, Z. *Handbook of Evolutionary Computation.* Oxford University Press, 1997.

[35] Michiels, W., Aarts, E., and Korst, J. *Theoretical Aspects of Local Search.* Sp, 2007.

[36] Mohamed, A. W. Rdel: Restart differential evolution algorithm with local search mutation for global numerical optimization. *Egyptian Informatics Journal 15* (2014), 175–188.

[37] Mohanty, B., Panda, S., and Hota, P. Controller parameters tuning of differential evolution algorithm and its application to load frequency control of multi-source power system. *Electrical Power and Energy Systems 54* (2014), 77–85.

[38] Nelder, J. A., and Mead, R. A simplex method for function minimization. *The Computer Journal 4* (1965), 308–313.

[39] Pearson, K. The problem of the random walk. *Nature 72* (1905), 294.

[40] Powell, D., and Skolnick, M. Using genetic algorithms in engineering design optimization with non-linear constraints. *Proceedings of the 5th International Conference on Genetic Algorithms* (1993), 424–431.

[41] Price, K., Storn, R. M., and Lampinen, J. A. *Differential evolution: a practical approach to global optimization.* Springer Science & Business Media, 2006.

[42] Qiang, J., and Mitchell, C. An adaptive unified differential evolution algorithm for global optimization. Tech. rep., Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA (US), 2014.

[43] Qin, A. K., Li, X., Pan, H., and Xia, S. Investigation of self-adaptive differential evolution on the cec-2013 real-parameter single-objective optimization testbed. *IEEE Congress on Evolutionary Computation* (2013), 1107–1114.

[44] Rusell, S., and Norvig, P. *Inteligencia Artificial (2da Edición) - Un Enfoque Moderno.* Pearson - Prentice Hall, 2004.

[45] Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2003.

[46] Sarker, R. A., Elsayed, S. M., and Ray, T. Differential evolution with dynamic parameters selection for optimization problems. *IEEE Transactions on Evolutionary Computation 18* (2014), 689–707.

[47] Schoenauer, M., and Xanthakis, S. Constrained ga optimization. *Proceedings of the 5th International Conference on Genetic Algorithms* (1993), 573–580.

[48] Schwefel, H. *Numerical Optimization of Computer Models.* John Wiley & Sons, Inc., 1981.

[49] Son, N. N., and Anh, H. P. H. Adaptive mimo neural network model optimized by differential evolution algorithm for manipulator kinematic system identification. *International Conference on Automatic Control Theory and Application 1* (2014), 23–26.

[50] Su, S.-C., Lin, C.-J., and Ting, C.-K. An effective hybrid of hill climbing and genetic algorithm for 2d triangular protein structure prediction. *Proteome Science 9* (2011), 1–8.

[51] Sun, W., and Yuan, Y.-X. *Optimization Theory and Methods: Nonlinear Programming.* Springer, 2006.

[52] Takahama, T., and Sakai, S. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. *IEEE World Congress on Computational Intelligence I* (2010), 1680–1688.

[53] Vitaliy, F. *Differential Evolution: In Search of Solutions.* Springer, 2006.

[54] Wang, M., Song, Z., Dai, G., Peng, L., and Zheng, C. Asteroids exploration trajectory optimal design with differential evolution based on mixed coding. *International Journal of Distributed Sensor Networks Article ID 827987* (2015), in press.

[55] Yi, W., Gao, L., Li, X., and Zhou, Y. A new differential evolution algorithm with a hybrid mutation operator and self-adapting control parameters for global optimization problems. *Applied Intelligence 42* (2015), 642–660.

[56] Yuret, D., and de la Maza, M. Dynamic hill climbing: Overcoming the limitations of optimization techniques. Tech. rep., Numinous Noetics Group, Artificial Intelligence Laboratory, MIT, 1993.

[57] Zanakis, S., and Evans, J. Heuristic optimization: why, when, and how to use it. *Interfaces 11* (1981), 84–91.

[58] Zhang, X., and Duan, H. An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning. *Applied Soft Computing 26* (2015), 270–284.