Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Evolución Diferencial Compacta Multi-Objetivo

Tesis que presenta

Jesús Moisés Osorio Velázquez

para obtener el Grado de

Maestro en Ciencias

en Computación

Director de Tesis

Dr. Carlos A. Coello Coello

México, Distrito Federal Septiembre, 2012

Resumen

Una vasta cantidad de problemas en ingeniería requieren la optimización de varios objetivos de manera simultánea. Dado el origen de dichos problemas, muchas veces la optimización debe realizarse desde un dispositivo con recursos computacionales limitados. Aunque actualmente existen algoritmos compactos exitosos que facilitan la implementación en dispositivos limitados, estos algoritmos resuelven problemas con una sola función objetivo.

Los algoritmos evolutivos multi-objetivo han sido muy exitosos en la solución de problemas de optimización con más de una función objetivo, debido, sobre todo, a su simplicidad y facilidad de uso. Sin embargo, la mayor parte de estos algoritmos utilizan una población de individuos, lo que obliga a hacer uso de más recursos de memoria tan sólo por almacenar toda esta información.

En esta tesis se propone un algoritmo multi-objetivo compacto basado en evolución diferencial, el cual busca resolver problemas de optimización con un menor uso de recursos de memoria. El algoritmo propuesto obtiene resultados competitivos, e incluso mejores, que los algoritmos representativos del estado del arte, al mismo tiempo que requiere menos recursos de memoria al adoptar una representación estadística de la población.

Abstract

A wide range of problems in enginnering require the optimization of several objectives at the same time. Given the nature of such problems, it is often the case that the optimization process needs to take place from a device with very limited resources. Although there exist compact algorithms that are more suitable for being implemented in devices with limited computing resources, such algorithms only deal with single-objective problems.

Multi-objetive evolutionary algorithms have been successful in solving optimization problems with more than one objective function, mainly due to their simplicity and ease of use. Nevertheless, most of these algorithms use a population of individuals, which requires more memory resources only to save all this information.

In this thesis, a multi-objective compact algorithm based on differential evolution is presented. The proposed algorithm obtains competitive (and even better in some cases) results than state-of-the-art multi-objective evolutionary algorithms while using less memory resources because of its statistical representation of the population.

Agradecimientos

Primero que nada deseo agradecer a mis padres, gracias a sus sacrificios estoy donde estoy y he tenido la oportunidad de forjarme para el futuro.

Gracias a mi esposa Cristina, quien se ha encargado de presionarme para no rezagarme y ser de los mejores. Gracias por toda su paciencia e impulsos positivos.

Muchas gracias a mi asesor el Dr. Carlos Coello, por guiarme más de un año a través del mundo de la investigación científica, permitiéndome realizar este trabajo y completar mis estudios.

Gracias a mis sinodales, el Dr. Luis Gerardo de la Fraga y el Dr. Francisco Rodríguez Henríquez, porque sus comentarios hicieron de esta tesis un mejor trabajo.

En general, quiero agradecer a todo el personal y estudiantes del CINVESTAV, por permitirme compartir todo este tiempo a su lado y hacer de mi estadía algo de valor.

También agradezco a toda mi familia y amigos, por estar siempre a mi lado y hacerme sentir que puedo hacer lo que me proponga, gracias por su apoyo incondicional.

Finalmente, agradezco a CONACYT, por apoyarme económicamente en mis estudios.

Este trabajo de tesis se derivó del proyecto de investigación titulado "Escalabilidad y Nuevos Esquemas Híbridos en Optimización Evolutiva Multiobjetivo" (Ref. 103570), cuyo responsable es el Dr. Carlos A. Coello Coello.

Contenido

Lista de figuras

Lista de tablas

Capítulo 1

Introducción

Muchas áreas de la ingeniería requieren de la optimización de dos o más objetivos de manera simultánea. Dichos problemas, llamados problemas multi-objetivo, suelen tener un conjunto de soluciones óptimas, y no una sola.

Para resolver problemas de optimización multi-objetivo, se han desarrollado distintos métodos de programación matemática. Sin embargo, estos métodos tienen ciertas limitantes y generalmente necesitan de información adicional sobre el problema a resolverse.

Esto ha motivado el uso de metaheurísticas que buscan resolver problemas multi-objetivo sin las limitantes que poseen los métodos de programación matemática. Entre dichas metaheurísticas, la computación evolutiva ha aportado algoritmos fáciles de entender y simples de usar, además de obtener soluciones razonablemente buenas.

Sin embargo, los problemas de optimización multi-objetivo se pueden encontrar en aplicaciones muy variadas, incluyendo aquellas en las que no se cuenta con el poder de cómputo necesario para utilizar apropiadamente los algoritmos evolutivos.

En esta tesis, se propone un algoritmo evolutivo compacto multi-objetivo, el cual tiene la finalidad de optimizar problemas multi-objetivo, obteniendo una calidad de soluciones igual o mayor a las soluciones de los algoritmos evolutivos representativos del estado del arte, minimizando los recursos de memoria necesarios para su ejecución.

Esta tesis está organizada de la siguiente manera:

- En el capítulo 2 se mencionan algunos conceptos básicos para comprender el trabajo realizado. Además, se describen los métodos de programación matemática más populares y algunos algoritmos representativos del estado del arte de la computación evolutiva para optimización multi-objetivo.

- El capítulo 3 presenta los trabajos anteriores que han motivado esta tesis, la cual se basa en algunos principios básicos establecidos por dichos trabajos.

- En el capítulo 4 se describe de manera detallada el diseño de mocDE, el algoritmo propuesto en esta tesis.

- El capítulo 5 muestra los resultados obtenidos al comparar distintas variantes de mocDE entre sí, así como una de dichas variantes contra algoritmos del estado del arte.
- Al final, en el capítulo 6 se enumeran las conclusiones obtenidas a partir de los resultados de esta tesis, así como algunos aspectos dignos de investigar en el futuro.

Capítulo 2

Conceptos básicos

La optimización tiene diversas aplicaciones en muchos campos como el diseño de procesos y productos, las finanzas, la industria petrolera, y muchas otras donde es necesario hacer el mejor uso posible de los recursos disponibles. Su objetivo es encontrar la mejor solución posible a un problema, lo que ha llevado a desarrollar diversos métodos de programación matemática. Sin embargo, dichos métodos tienen ciertas limitantes y necesitan de información adicional al modelo del problema, es decir, sólo operan sobre problemas continuos donde se conoce su gradiente.

Por lo tanto, las técnicas heurísticas han surgido como una alternativa para resolver problemas de optimización, ya que son capaces de obtener buenas soluciones, aunque no necesariamente óptimas, con un costo computacional razonable, pero sin requerir información adicional al problema.

La computación evolutiva es un tipo de heurística que simula la evolución de los seres vivos, viendo a las soluciones de un problema como individuos y aplicándoles mecanismos evolutivos para mejorarlos. Los algoritmos utilizados dentro de la computación evolutiva suelen ser fáciles de entender y usar, y suelen obtener soluciones razonablemente buenas en tiempos razonablemente cortos.

En la sección 2.1 se mencionan los conceptos básicos de optimización, mientras que en la sección 2.2 se discuten los métodos de programación matemática más utilizados. Finalmente, en las secciones 2.3 y 2.4 se hace una breve introducción a la computación evolutiva y los algoritmos evolutivos multi-objetivo del estado del arte.

2.1. Optimización

De manera general, la optimización se define como el proceso para minimizar o maximizar el valor de una o varias funciones que representan los objetivos de un problema. Cuando un problema tiene solo una función objetivo se le denomina mono-objetivo, y cuando tiene dos o más se le llama multi-objetivo.

En problemas mono-objetivo, la noción de óptimo (la mejor solución posible, dadas las restricciones del problema) es un tanto intuitiva. Sin embargo, esto no ocurre en problemas multi-objetivo, donde normalmente sus funciones objetivo están en con-

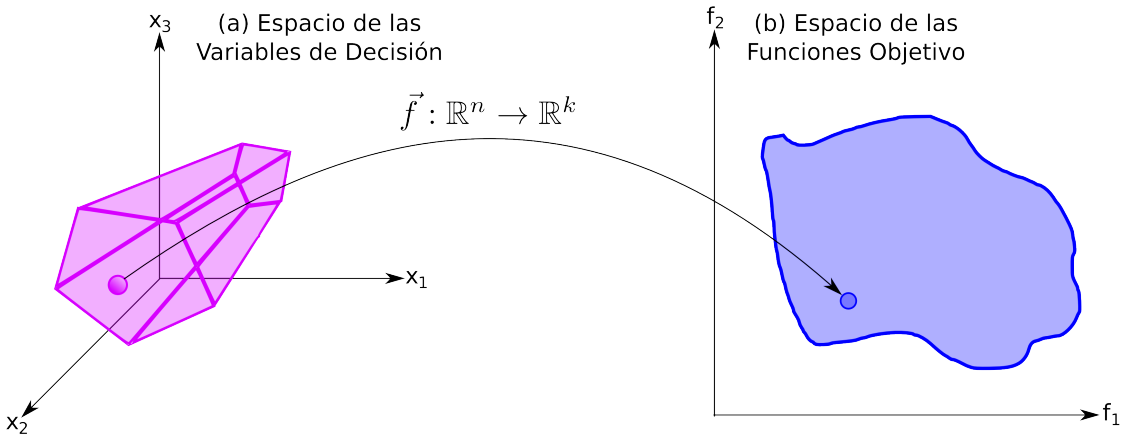


Figura 2.1: Espacios de búsqueda en un problema de optimización.

flicto, es decir, optimizar una función implica empeorar otra. En consecuencia, en los problemas multi-objetivo se genera un conjunto de soluciones que representan los mejores compromisos posibles entre los objetivos del problema.

Definición 1 (Variables de decisión)**.** Las *variables de decisión* x_i, $i = 1, 2, ..., n$ son los n parámetros que determinan una solución al problema.

Las variables de decisión se denotan por el vector:

$$\vec{x} = [x_1, x_2, ..., x_n]^T, \quad \vec{x} \in \mathbb{R}^n \tag{2.1}$$

Al conjunto $\mathbb{R}^n$ se le conoce como *espacio de las variables de decisión* (véase figura 2.1).

Definición 2 (Funciones objetivo)**.** Las *funciones objetivo* $f_i(\vec{x})$, $i = 1, 2, ..., k$ son las k funciones vectoriales que determinan qué tan buena es una solución al problema.

Las funciones objetivo son denotadas por el vector:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})]^T, \quad \vec{f}(\vec{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^k \tag{2.2}$$

Al conjunto $\mathbb{R}^k$ se le conoce como *espacio de las funciones objetivo* (véase figura 2.1).

Definición 3 (Región factible)**.** La *región factible* $\mathcal{F}$ es el conjunto de soluciones $\vec{x} \in \mathbb{R}^n$ que cumplen con las restricciones del problema.

Dichas restricciones están definidas por l ecuaciones de desigualdad y m ecuaciones de igualdad:

$$g_i(\vec{x}) \leq 0, \quad i = 1, 2, ..., l \tag{2.3}$$

$$h_j(\vec{x}) = 0, \quad j = 1, 2, ..., m \tag{2.4}$$

Definición 4 (Problema de optimización)**.** De manera formal, un *problema de optimización* es expresado como:

$$\min_{\vec{x}\in\mathcal{F}} \vec{f}(\vec{x}) \tag{2.5}$$

donde $\vec{f}: \mathbb{R}^n \to \mathbb{R}^k$ es el vector de funciones objetivo, $\vec{x} \in \mathbb{R}^n$ es el vector de variables de decisión y $\mathcal{F} \subseteq \mathbb{R}^n$ es la región factible.

En problemas mono-objetivo ($k = 1$), la relación *igual o menor que* ($\leq$) permite que distintas soluciones se comparen, manteniendo un orden total en $\mathbb{R}$. Por lo tanto, una solución $\vec{x}* \in \mathcal{F}$ es óptima si y sólo si:

$$f(\vec{x}*) \leq \vec{y}, \quad \forall \vec{y} \in \mathcal{F} \tag{2.6}$$

En problemas multi-objetivo ($k \geq 2$), se utiliza la *dominancia de Pareto* para comparar distintas soluciones.

Definición 5 (Dominancia de Pareto)**.** Dadas dos soluciones $\vec{x}, \vec{y} \in \mathcal{F}$, se dice que $\vec{x}$ domina (en el sentido de Pareto) a $\vec{y}$ ($\vec{x} \prec \vec{y}$) si y sólo si:

$$f_i(\vec{x}) \leq f_i(\vec{y}), \quad i = 1, 2.., k, \quad \text{y} \tag{2.7}$$
$$f_j(\vec{x}) < f_j(\vec{y}), \quad j = 1, 2.., k \tag{2.8}$$

La dominancia de Pareto no impone un orden total en $\mathbb{R}^k$ ya que algunas soluciones pueden resultar incomparables. Por lo tanto, la mayoría de los problemas multi-objetivo no tienen una solución única, sino un conjunto de ellas (véase figura 2.2).

Definición 6 (Conjunto de óptimos de Pareto)**.** El *conjunto de óptimos de Pareto* $\mathcal{P}$ se define como:

$$\mathcal{P} = \{\vec{x}* \in \mathcal{F} \mid \forall \vec{y} \in \mathcal{F} \quad \vec{y} \not\preceq \vec{x}*\} \tag{2.9}$$

Definición 7 (Frente de Pareto)**.** El *frente de Pareto* $\mathcal{P}_F$, mostrado en la figura 2.3, es definido como:

$$\mathcal{P}_F = \{\vec{f}(\vec{x}*) \mid \forall \vec{x}* \in \mathcal{P}\} \tag{2.10}$$

De esta manera, cuando un algoritmo utiliza la dominancia de Pareto para comparar soluciones, tiene como objetivo, al resolver un problema multi-objetivo, encontrar $\mathcal{P}$ y su correspondiente $\mathcal{P}_F$. Ya que las soluciones a reportar por un algoritmo son finitas, aumenta la importancia de mantener una buena distribución de dichas soluciones. Dicha distribución, más la convergencia de los resultados, se vuelven los indicadores más importantes sobre la calidad de las soluciones reportadas por un algoritmo de optimización multi-objetivo.

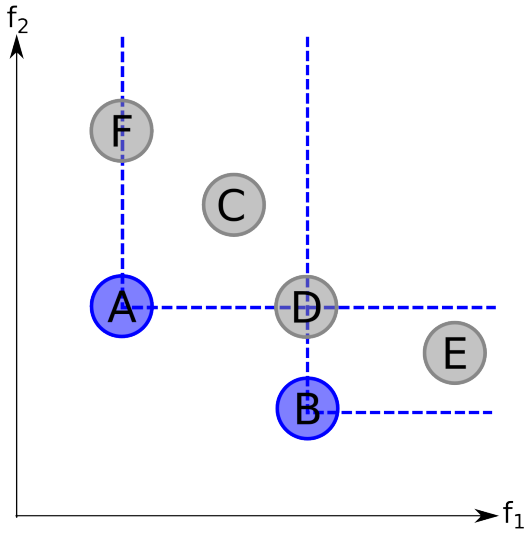


Figura 2.2: En un problema con dos objetivos, la dominancia de Pareto determina que los puntos azules son mejores que los puntos grises.

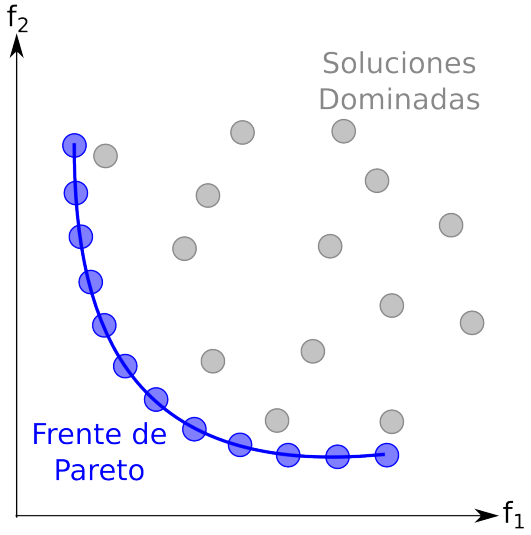


Figura 2.3: El frente de Pareto está formado por todas las soluciones no dominadas.

2.2. Métodos de programación matemática para problemas multi-objetivo

En esta sección se ilustran algunos de los métodos de programación matemática más populares para resolver problemas de optimización multi-objetivo, los cuales se dividen en métodos *a priori*, *a posteriori* e interactivos, dependiendo de los criterios que aplique el usuario (persona o grupo de personas cuya labor es tomar decisiones) respecto a la toma de decisiones y los compromisos de cada problema en particular.

2.2.1. Métodos con articulación de preferencias *a priori*

En los métodos con articulación de preferencias a priori, el usuario debe especificar sus preferencias y expectativas antes de que el método se ejecute.

Método de programación de metas

Propuesto en [1] como un enfoque para usarse en modelos lineales, el *método de programación de metas* es uno de los primeros métodos usados para la solución de problemas de optimización multi-objetivo.

En este método, el usuario debe asignar metas a alcanzar por cada uno de los objetivos del problema. Dichas metas se incorporan al problema en la forma de restricciones adicionales. Finalmente, la función objetivo trata de minimizar la desviación absoluta entre el valor alcanzado y la meta definida. De forma simple, el método es formulado de la siguiente manera:

$$\min_{x \in \mathcal{F}} \sum_{i=1}^{k} |f_i(x) - G_i| \tag{2.11}$$

donde G_i es la meta definida por el usuario para la i-ésima función objetivo f_i.

Cabe destacar que esta técnica no obtiene un óptimo de Pareto si las metas no son definidas correctamente.

Método lexicográfico

En el *método lexicográfico* [2], el usuario ordena las funciones objetivo de acuerdo a su importancia, siendo la primera la más importante. El método inicia optimizando la primera función objetivo, sujeta a las restricciones originales del problema. Después, minimiza la segunda función objetivo agregando como una restricción adicional el resultado de optimizar la primera. El método continúa optimizando las siguientes funciones objetivo tomando como restricciones los resultados previos. El método se

ilustra a continuación:

$$\begin{aligned} \min_{x\in\mathcal{F}} f_1(x) &= \alpha_1 \\ \min_{x\in\mathcal{F}}\{f_2(x)|f_1(x) \leq \alpha_1\} &= \alpha_2 \\ &\vdots \\ \min_{x\in\mathcal{F}}\{f_n(x)|f_1(x) \leq \alpha_1, ..., f_{n-1}(x) \leq \alpha_{n-1}\} &= \alpha_n \end{aligned} \tag{2.12}$$

En [3, 4] se muestra que la solución obtenida es un óptimo de Pareto.

2.2.2. Métodos con articulación de preferencias *a posteriori*

En los métodos con articulación de preferencias a posteriori, el usuario elige las mejores alternativas, de acuerdo a su criterio, de las soluciones que resultan de ejecutar el método.

Método de combinación lineal de pesos

En [5], Zadeh transforma un problema de optimización multi-objetivo a uno mono-objetivo asociando un coeficiente de peso con cada función objetivo y minimizando la suma de todos ellos. El *método de combinación lineal de pesos* se formula de la siguiente manera:

$$\min_{x\in\mathcal{F}} \sum_{i=1}^{k} w_i f_i(x) \tag{2.13}$$

donde $w_i \geq 0$, existiendo al menos un j tal que $w_j > 0$.

Al variar los pesos, es posible obtener cualquier solución óptima de Pareto en un problema convexo. Sin embargo, para obtener un óptimo de Pareto es necesario que todos los pesos sean positivos.

Método de restricción ϵ

El *método de restricción* ϵ [3] es una de las técnicas de escalarización más conocidas para resolver problemas multi-objetivo. Este enfoque trata de minimizar un objetivo mientras los demás son utilizados como restricciones limitadas a ciertos valores ϵ_i. El método se puede formular como se muestra a continuación:

$$\min_{x\in\mathcal{F}}\{f_i(x)|f_j(x) \leq \epsilon_j, \forall j \in \{1, ..., k\} \vee j \neq i\} \tag{2.14}$$

donde ϵ_i son los límites que el usuario define.

Para obtener distintas soluciones, es necesario resolver el problema utilizando diferentes valores de ϵ_i. En [3, 4] se demuestra que la soluciones óptimas obtenidas son óptimos de Pareto.

2.2.3. Métodos con articulación interactiva de preferencias

En estos métodos, el usuario participa de manera activa durante el proceso de optimización, utilizando su conocimiento sobre el problema en cuestión.

Método de Chebyshev

El *método de Chebyshev* [6] transforma un problema de optimización multi-objetivo a uno mono-objetivo minimizando la distancia a un vector objetivo utópico o ideal μ (definido por el usuario), utilizando la métrica ponderada de Chebyshev. Éste vector utópico debe dominar, de acuerdo a la relación de Pareto, a cualquier otro vector solución que se genere en la búsqueda. La formulación del problema se muestra a continuación:

$$\min_{x \in \mathcal{F}} \{ \max_{1 \leq i \leq k} |f_i(x) - \mu_i| \} \tag{2.15}$$

donde μ es el vector objetivo utópico.

Este método tiene la característica de poder obtener cualquier punto óptimo de Pareto.

Método del punto de referencia

El *método del punto de referencia* [7] es una técnica basada en la definición de función de una escalarización de logros.

Definición 8 (Función de escalarización de logros)**.** Una *función de escalarización de logros* es una función parametrizada $s(x, z) : \mathbb{R}^k \to \mathbb{R}$, donde $x \in \mathbb{R}^k$ es el vector a escalar y $z \in \mathbb{R}^k$ es un punto de referencia que representa las preferencias del usuario. Esto convierte al problema multi-objetivo en el problema escalar:

$$\min_{x \in \mathcal{F}} s(x, z) \tag{2.16}$$

En esta técnica, inicialmente se le pide un punto de referencia al usuario. Después se determinan, con la función de escalarización, las soluciones que mejor satisfacen las preferencias del usuario. Si el usuario queda satisfecho con el resultado, el proceso termina; de lo contrario, se pide otro punto de referencia para repetir el proceso.

Método de búsqueda de haz de luz

Propuesto en [8], el *método de búsqueda de haz de luz* combina la idea del punto de referencia con otras utilizadas en el análisis de decisión multi-atributo.

En cada iteración, este método genera un conjunto de soluciones compuesto por un punto llamado *punto medio*, tomado de la iteración anterior, y j puntos no dominados pertenecientes a su vecindario. Para definir el vecindario, el usuario debe definir un modelo de preferencia local en la forma de una relación S, indicando umbrales de

indiferencia, *preferencia* y *veto*. Dicha relación determina si una solución a es mejor que otra solución b (aSb).

Con esta técnica, el usuario puede escanear el espacio objetivo entre dos soluciones en el vecindario, permitiéndole explorarlo con más precisión.

2.3. Computación evolutiva

Los algoritmos evolutivos hacen uso de mecanismos inspirados en la evolución de los seres vivos que, al ser simulados en una computadora, son capaces de resolver problemas de optimización del mundo real y obtener un buen desempeño. Si bien los algoritmos evolutivos no siempre son capaces de converger a la mejor solución posible, sí pueden obtener una buena aproximación sin las limitantes de los métodos de programación matemática.

En cada *generación* o iteración, un algoritmo evolutivo aplica ciertos operadores a una *población* actual con el fin de generar nuevos y mejores *individuos*, los cuales son conservados o descartados de acuerdo a un criterio de *selección*. Los operadores esenciales son la *recombinación* y la *mutación*. La recombinación toma dos o más individuos para generar a uno o más individuos nuevos. La mutación varía, de manera aleatoria, un segmento de un individuo.

2.3.1. Paradigmas

A continuación, se describen brevemente los paradigmas más populares de la computación evolutiva.

Programación evolutiva

La *programación evolutiva* fue propuesta por Lawrence J. Fogel [9], y después adaptada por David Fogel [10] para la optimización numérica. El algoritmo 1 describe su estructura básica.

Algoritmo 1 Programación evolutiva

Entrada: $\mu > 0$
Salida: Conjunto de vectores solución P
  Generar una población aleatoria P de tamaño μ
  Evaluar P
  mientras no se cumpla condición de paro **hacer**
    Generar un conjunto Q de individuos al mutar individuos de P
    Evaluar los hijos en Q
    Reemplazar P con selección por torneo de $P \cup Q$
  devolver P

La programación evolutiva inicia con una población aleatoria de μ individuos. En cada generación, se le aplica a la población actual un operador de mutación, el cual

genera μ nuevos individuos. Después, se aplica un torneo estocástico a la unión de los $\mu + \mu$ individuos para generar la siguiente población. En este paradigma no existe el operador de recombinación.

Estrategias evolutivas

Las *estrategias evolutivas* fueron propuestas por Ingo Rechenberg y Hans-Paul Schwefel [11]. El algoritmo 2 describe su estructura.

Algoritmo 2 Estrategia evolutiva $(\mu + \lambda)$

Entrada: $\mu > 0 \wedge \lambda > 0$
Salida: Conjunto de vectores solución P
  Generar una población aleatoria P de tamaño μ
  Evaluar P
  mientras no se cumpla condición de paro **hacer**
    Seleccionar λ individuos padre de P para crear conjunto Q
    Aplicar recombinación a Q para generar hijos R
    Mutar los individuos en R
    Evaluar R
    Reemplazar P seleccionando μ individuos de $P \cup R$
  devolver P

Al inicio del algoritmo se genera una población aleatoria inicial. En cada generación, se selecciona un subconjunto de la población actual para generar nuevos hijos mediante recombinación. Después, se les aplica a los hijos el operador de mutación, el cual es el más importante, para evaluarlos y seleccionar los más aptos que reemplazarán a la población actual. La selección puede ser realizada de dos maneras, ya sea seleccionando los mejores μ individuos sólo del conjunto de hijos (selección (μ, λ), o de la unión de conjuntos de padres e hijos (selección $(\mu + \lambda)$).

Algoritmos genéticos

Los *algoritmos genéticos* fueron propuestos inicialmente por John Holland [12], y ahora constituyen el paradigma más popular en la computación evolutiva. Su estructura básica se detalla en el algoritmo 3.

Los algoritmos genéticos inician con una población aleatoria inicial. En cada generación, se selecciona un conjunto de individuos padre de la población actual, los cuales se recombinarán para generar nuevos individuos. A estos nuevos individuos, se les aplica el operador de mutación, y el resultado es la nueva población del algoritmo. En los algoritmos genéticos, el operador más importante es la recombinación, ya que guía el proceso de optimización, mientras que la mutación trata de evitar que la población se quede atrapada en óptimos locales.

Algoritmo 3 Algoritmo genético

Entrada: $n > 0$
Salida: Conjunto de vectores solución P
  Generar una población aleatoria P de tamaño n
  Evaluar P
  mientras no se cumpla condición de paro **hacer**
    Seleccionar los individuos padre de P para crear conjunto Q
    Aplicar recombinación a Q para generar hijos R
    Mutar los individuos en R
    Evaluar R
    Reemplazar P con R
  devolver P

Evolución diferencial

La *evolución diferencial* fue propuesta por Kenneth Price y Rainer Storn [13]. En los últimos años, ha ganado mucha popularidad ya que ha demostrado ser una metaheurística muy robusta a pesar de su simplicidad. En el algoritmo 4, se puede observar su estructura básica.

Algoritmo 4 Evolución diferencial

Entrada: $n > 0$
Salida: Conjunto de vectores solución P
  Generar una población aleatoria P de tamaño n
  Evaluar P
  mientras no se cumpla condición de paro **hacer**
    para $i = 1 \rightarrow n$ **hacer**
      Seleccionar el individuo base x' de P
      Seleccionar el conjunto C de y individuos padre de P
      Generar un individuo x agregándole la diferencia vectorial de C a x' de acuerdo a la variante deseada
      si $f(x) < f(P_i)$ **entonces**
        $P_i \leftarrow x$
  devolver P

La evolución diferencial, al igual que la mayoría de los algoritmos evolutivos, inicia con una población generada aleatoriamente. En cada iteración, se trata de mejorar cada individuo en la población al agregar la diferencia vectorial de un conjunto de individuos en P a otro individuo seleccionado.

Existen diversas variantes de la evolución diferencial, denotadas por *DE/x/y/z* donde:

- x especifica el vector a ser mutado, el cual puede ser *cur* (el vector actual), *rand* (un vector aleatorio tomado de la población) o *best* (el vector con el menor costo

de la población).

- y es la cantidad de vectores diferencia a utilizar.
- z denota el esquema de cruza, siendo *bin* (cruza de acuerdo a experimentos binomiales independientes) el más popular.

De acuerdo a dicha notación, las variantes más populares de la evolución diferencial, con su respectiva fórmula de mutación, son:

- *DE/rand/1/bin*: $x = x_t + F \cdot (x_r - x_s)$.
- *DE/best/1/bin*: $x = x_{best} + F \cdot (x_r - x_s)$.
- *DE/cur-to-best/1/bin*: $x = x_k + F \cdot (x_{best} - x_k) + F \cdot (x_r - x_s)$.
- *DE/best/2/bin*: $x = x_{best} + F \cdot (x_r - x_s) + F \cdot (x_u - x_v)$.
- *DE/rand/2/bin*: $x = x_t + F \cdot (x_r - x_s) + F \cdot (x_u - x_v)$.
- *DE/rand-to-best/1/bin*: $x = x_t + F \cdot (x_{best} - x_t) + F \cdot (x_r - x_s)$.
- *DE/rand-to-best/2/bin*: $x = x_t + F \cdot (x_{best} - x_t) + F \cdot (x_r - x_s) + F \cdot (x_u - x_v)$.

donde x es el vector resultante, F es la constante de amplificación de la variación diferencial, x_k es el vector actual, x_{best} es el mejor vector de la población y los vectores x_r, x_s, x_t, x_u y x_v son tomados aleatoriamente de la población.

2.4. Algoritmos evolutivos multi-objetivo

Como se mostró en la sección 2.2, existe una gran variedad de métodos matemáticos diseñados para resolver problemas multi-objetivo. Sin embargo, muchos de estos métodos son susceptibles a la forma y continuidad del frente de Pareto, y a que su solución inicial cumpla con ciertas características. Además, el resultado de una ejecución de estos métodos es normalmente sólo una solución no dominada, por lo que deben ejecutarse varias veces, partiendo de distintas soluciones iniciales, para obtener un conjunto de ellas.

Dado que los algoritmos evolutivos operan sobre un conjunto de individuos, basta una sola ejecución para encontrar un conjunto representativo del frente de Pareto. Además, pueden lidiar con frentes de Pareto cóncavos y discontinuos, y no necesitan información adicional sobre el problema.

Los algoritmos evolutivos mono-objetivo y multi-objetivo poseen la misma estructura, y sus diferencias principales son la manera en cómo se determina la aptitud de un individuo y el mecanismo elitista utilizado. Sus diferencias se deben a que los algoritmos multi-objetivo optimizan más de una función objetivo, y dan como resultado un conjunto de soluciones y no sólo una.

2.4.1. Elitismo

El elitismo es el mecanismo que trata de preservar los mejores individuos durante el proceso de optimización. En la optimización mono-objetivo sólo se tiene que lidiar con un individuo; sin embargo, en el caso multi-objetivo se podría tener un gran número de individuos no dominados entre sí, y su cantidad se incrementa mientras avanza el proceso de optimización. Por lo tanto, es necesario contar con un método de filtrado para mantener un número máximo de mejores individuos.

Existe dos enfoques principales para implementar el elitismo en un algoritmo evolutivo multi-objetivo:

- Implementar un mecanismo de selección que tome los mejores individuos de la unión de la población actual y la nueva población constituida por los nuevos individuos.

- Utilizar una población externa, llamada *archivo*, donde se almacenen las soluciones no dominadas encontradas durante todo el proceso de optimización; se suele limitar el tamaño del archivo de acuerdo a un criterio de filtrado.

2.4.2. Asignación de aptitud

Incluso en los algoritmos evolutivos multi-objetivo, el mecanismo de selección necesita de un valor de aptitud de tipo escalar. Por lo tanto, el valor de aptitud debe ser calculado aún teniendo más de una función objetivo. Existen tres enfoques principales:

- El enfoque *basado en criterios* elige sólo una función objetivo, la cual será utilizada para determinar su aptitud.

- El esquema *basado en agregación* combina las funciones objetivo en una sola función parametrizada, de cuyo valor dependerá la aptitud de los individuos. Para que la optimización genere un conjunto de soluciones no dominadas, los parámetros deben ser variados durante el proceso.

- El esquema *basado en Pareto* utiliza la dominancia de Pareto para comparar individuos. Este esquema es el más utilizado, e incluso se le puede dividir en dos generaciones:

    - Durante la primera generación (desarrollada entre 1984 y 1998) los algoritmos evolutivos multi-objetivo carecían de un mecanismo elitista, su diseño era relativamente simple y sus mecanismos de diversidad eran primitivos.
    - La segunda generación incorporó el mecanismo elitista, así como nuevos estimadores de densidad y diseños algorítimicos más elaborados y eficientes.

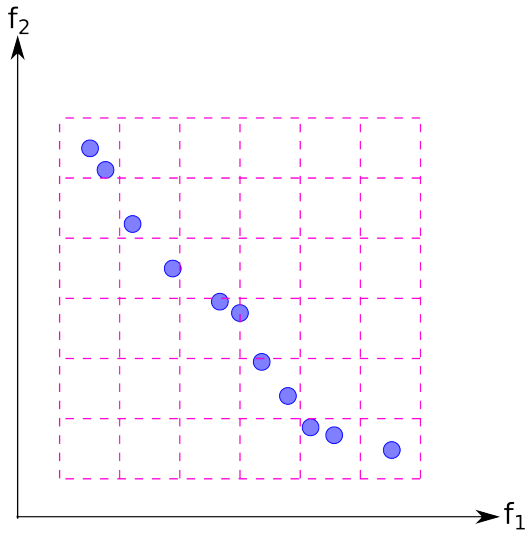


Figura 2.4: Malla adaptativa de PAES con 6 divisiones en un problema con dos objetivos.

2.4.3. Estado del arte

En la actualidad, existen diversos algoritmos evolutivos multi-objetivo que han demostrado tanto su eficiencia como su eficacia, lo cual los ha vuelto populares dentro de la comunidad científica. A continuación, se presentan tres algoritmos contra los cuales se comparan resultados en esta tesis.

Pareto Archived Evolution Strategy (PAES)

Pareto Archived Evolution Strategy (PAES) [14] consiste de una estrategia evolutiva (1+1) que hace uso de un archivo externo para almacenar las soluciones no dominadas encontradas durante el proceso de optimización. Posee una malla adaptativa que divide el espacio objetivo de manera recursiva, permitiendo calcular la densidad de cada región del frente de Pareto.

PAES inicia con un solo individuo, el cual inicializa el archivo de soluciones. En cada iteración, genera un nuevo individuo mutando al actual. Si el nuevo individuo es mejor que el actual, de acuerdo a la dominancia de Pareto, el nuevo individuo lo reemplaza y se agrega al archivo. Si el individuo actual domina al nuevo, no hay cambios en el archivo y se avanza a la siguiente iteración. Si ambos individuos son no dominados entre sí, prevalecerá aquél que se encuentre en la región menos densa del archivo, de acuerdo a la malla adaptativa.

Es evidente que el archivo es utilizado para dos propósitos: mantener y actualizar las soluciones no dominadas, y ayudar en la selección entre el individuo actual y el candidato. La figura 2.4 muestra un ejemplo de la malla adaptativa.

Aunque sus resultados no compiten contra la mayoría de los algoritmos del estado

del arte, es el algoritmo evolutivo multi-objetivo más simple que se pueda concebir, de forma conceptual.

Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)

Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [15] descompone un problema multi-objetivo en p subproblemas escalares. Para ello, hace uso de cualquier método de descomposición, siendo el de Chebyshev el más popular. Para que MOEA/D descomponga un problema, necesita de un conjunto de vectores de pesos $\vec{\lambda}_i, i = 1, 2, ..., p$ uniformemente distribuidos y un vector de referencia $\vec{z}$. El escalamiento se realiza de la siguiente manera:

$$g_i(\vec{x}, \vec{\lambda}_i, \vec{z}) = \max_{1 \leq j \leq k} \{\vec{\lambda}_{i,j}(f_j(\vec{x}) - \vec{z}_j)\} \tag{2.17}$$

MOEA/D minimiza todos estos objetivos de manera simultánea en una sola ejecución.

Hay que notar que g es continuo en λ, por lo que la solución óptima de $g_i(\vec{x}, \vec{\lambda}_i, \vec{z})$ debe estar cerca a $g_j(\vec{x}, \vec{\lambda}_j, \vec{z})$ si $\vec{\lambda}_i$ es cercano a $\vec{\lambda}_j$. Por lo tanto, cualquier información acerca de g con pesos cercanos a $\vec{\lambda}_i$ debe ser de ayuda al optimizar $g_i(\vec{x}, \vec{\lambda}_i, \vec{z})$. Ésta es la mayor motivación de MOEA/D.

En MOEA/D, el vecindario de $\vec{\lambda}_i$ es definido como los s vectores $\vec{\lambda}_j, j = 1, 2, ..., s$ más cercanos a $\vec{\lambda}_i$.

La población está compuesta por la mejor solución encontrada para cada subproblema. Sólo las soluciones vecinas actuales son explotadas para optimizar un subproblema.

En cada generación t, MOEA/D mantiene:

- Una población de p individuos $\vec{x}_i \in \mathcal{F}, i = 1, 2..., p$, donde x_i es la solución actual para el subproblema i.
- Un conjunto de vectores $\vec{F}_i, i = 1, ..., p$, donde $\vec{F}_i = f(\vec{x}_i)$.
- Un vector ideal $\vec{z}$, donde $\vec{z}_i, i = 1, ..., k$ es el mejor valor encontrado para el objetivo f_i.
- Un archivo externo, utilizado para almacenar las soluciones no dominadas encontradas.

MOEA/D incorpora, en su última versión [16], un método de detección de subproblemas difíciles, es decir, aquéllos que toman más esfuerzo computacional para ser optimizados, para de esta manera dedicarles más recursos.

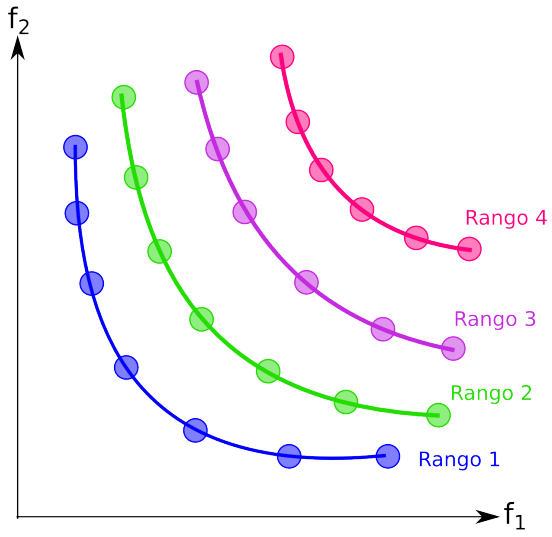


Figura 2.5: Jerarquización de Pareto en un problema con dos objetivos.

Non-Dominated Sorting Genetic Algorithm II (NSGA-II)

Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [17] emplea una *jerarquización de Pareto* y un mecanismo de preservación de la diversidad basada en el agrupamiento.

La jerarquización de Pareto, mostrada en la figura 2.5, consiste en clasificar a los individuos en diversas categorías de acuerdo al esquema siguiente:

- Los individuos no dominados pertenecen a la primera jerarquía.
- Omitiendo los individuos ya clasificados, se determina el nuevo conjunto de individuos no dominados, asignándoles el siguiente nivel de la jerarquía.
- Se repite el proceso hasta que todos los individuos hayan sido jerarquizados.

El NSGA-II estima la densidad de las soluciones alrededor de una solución particular en la población, calculando la distancia promedio hacia los dos puntos a los lados de cada objetivo del problema. Este valor es llamado *distancia de agrupamiento*.

Durante la selección, el NSGA-II utiliza un comparador que toma en consideración tanto la jerarquía de Pareto de un individuo, como su distancia de agrupamiento. De esta manera, se prefieren las soluciones no dominadas, y como criterio de desempate se toma la que se encuentre en la región menos densa del espacio de las funciones objetivo.

El NSGA-II no utiliza un archivo externo como otros algoritmos. Sin embargo, cuenta con un mecanismo elitista que consiste en combinar los mejores padres con los mejores hijos, como en una selección $(\mu + \lambda)$.

Este algoritmo se ha vuelto tan popular que es altamente citado y la gran mayoría de los nuevos algoritmos de optimización se comparan contra él para validar su desempeño. Sin embargo, el desempeño propio del NSGA-II se degrada rápidamente al aumentar el número de objetivos del problema [18].

Capítulo 3

Algoritmos evolutivos compactos mono-objetivo

En el mundo real, existen muchas aplicaciones que involucran la solución de problemas de optimización en las que no se cuenta con mucho poder de cómputo, ya sea por razones de costo o de espacio. Ésta es una situación que continuamente sucede en problemas de robótica y control. Por ejemplo, un robot en una línea de ensamblaje de autos necesita calcular la trayectoria óptima para la colocación de parabrisas, lo que requiere resolver un problema de optimización para no dañar el vidrio y colocarlo correctamente en el menor tiempo posible. Dicho robot no cuenta con un gran poder de cómputo, ya que eso aumentaría su volumen y su consumo de energía. En una aplicación como ésta, el uso de una metaheurística poblacional tradicional (por ejemplo, un algoritmo genético) pudiese resultar inadecuada debido a las limitantes de memoria y poder de cómputo para ejecutarla en un tiempo adecuado.

Para mitigar la falta de poder de cómputo, se han desarrollado los algoritmos evolutivos compactos. Un algoritmo evolutivo compacto pertenece a la clase de algoritmos de estimación de distribución [19], pero adopta principios que también son utilizados en algoritmos evolutivos tradicionales. Los algoritmos de estimación de distribución no guardan ni procesan una población completa de individuos, sino una representación estadística de ella. De esta manera, es necesario guardar menos valores en memoria, lo que lleva a que tengan un consumo de memoria mucho menor a su contraparte tradicional.

A lo largo de todo este capítulo, se describirán algunos algoritmos que fundamentan los principios en los que se basa mocDE, algoritmo presentado en el capítulo 4. Sin embargo, todos los algoritmos presentados se enfocan en resolver problemas de optimización con un solo objetivo, por lo que mocDE resulta al utilizar todos estos principios para aprovechar sus características al resolver problemas multi-objetivo.

3.1. Compact Genetic Algorithm

El *Compact Genetic Algorithm* (cGA) [20] fue el primer algoritmo evolutivo compacto propuesto. El cGA simula un algoritmo genético tradicional de codificación binaria, alcanzando un desempeño equiparable.

El cGA consiste de un vector de probabilidad v de longitud n, el cual es inicializado con n valores iguales a 0.5, y determina la probabilidad de que un gen sea 0 o 1. En cada iteración, se toman dos muestras de v, que fungirán como individuos, y se calcula su aptitud para competir entre sí. El individuo ganador influirá en v con base en el tamaño de la población virtual p. Es decir, si el ganador tiene un 1 en el gen i y el perdedor un 0, la probabilidad de la posición i en v aumenta $1/p$. En caso contrario, si el ganador tiene un 0 y el perdedor un 1, el valor i de v disminuye $1/p$. En caso de que ambos individuos tengan el mismo valor, v no se modifica. El vector v resultante representa la solución final.

El algoritmo 5 muestra cómo opera el cGA.

Algoritmo 5 Compact Genetic Algorithm (cGA)

Entrada: $n > 0$
Salida: Vector solución v

```
para i = 1 → n hacer {Inicializar v}
  v_i ← 0,5
mientras no se cumpla condición de paro hacer
  Generar dos individuos a y b, por medio de v
  si f(a) < f(b) entonces {Elegir al mejor}
    w ← a
    l ← b
  sino
    w ← b
    l ← a
  para i = 1 → n hacer {Afectar distribución de la población}
    si w_i <> l_i entonces
      si w_i = 1 entonces
        v_i ← v_i + 1/p
      sino
        v_i ← v_i − 1/p
devolver v
```

Es posible observar que, con una población de p individuos, el cGA requiere sólo $n \log_2(p+1)$ bits de memoria, mientras que un algoritmo genético tradicional requiere np bits.

Individuo	Cromosoma	f
a	01101	3
b	00110	2

Tabla 3.1: Individuos en el problema *onemax*.

3.1.1. Selección

El proceso de selección del cGA se enfoca en favorecer a los mejores genes, en lugar de los mejores individuos como en un algoritmo genético tradicional. Por ejemplo, consideremos el problema *onemax*, donde se trata de maximizar el número de genes con valor 1. Supongamos que el individuo a compite contra el individuo b (véase tabla 3.1).

Cuando dichos individuos compitan, el individuo a ganará. Sin embargo, a nivel de genes se ha producido un error en la cuarta posición, es decir, la selección prefiere al esquema $***0*$ sobre el esquema $**1*$. En este caso, el papel de la población es amortiguar estos errores y favorecer a los mejores genes. Los autores muestran que este esquema es equivalente a un torneo binario [20].

3.1.2. Recombinación

En un algoritmo genético tradicional, el papel de la recombinación es combinar los bits de buenas soluciones lo cual, después de varias repeticiones, lleva a la determinación de la relación entre los genes de la población. En este estado, la población puede ser representada de manera más compacta con un vector de probabilidad. Por lo tanto, la generación de individuos a partir de este vector puede ser vista como un atajo al objetivo final de la recombinación.

3.2. Persistent Elitist Compact Genetic Algorithm y Non-Persistent Elitist Compact Genetic Algorithm

En [21] se proponen dos variantes del cGA: *Persistent Elitist Compact Genetic Algorithm* (pe-cGA) y *Non-Persistent Elitist Compact Genetic Algorithm* (ne-cGA). Ambas variantes, demuestran tener mejor desempeño que cGA al incorporar elitismo al proceso de optimización.

Tanto pe-cGA como ne-cGA inician el proceso de optimización generando aleatoriamente un individuo extra llamado *élite*, además del vector de probabilidad v. En cada iteración, estos algoritmos sólo generan un individuo más a partir del vector v, y no dos como el cGA. Esto es porque el nuevo individuo competirá contra el individuo élite. Si el ganador de la competencia es el individuo élite, dicho individuo influirá

sobre v, tal como lo hace el cGA. Si el ganador es el nuevo individuo, éste influirá sobre v de la misma manera que el cGA y reemplazará al individuo élite.

La diferencia entre ambas variantes estriba en el momento en que un nuevo individuo se convierte en el individuo élite. El pe-cGA sólo reemplaza al individuo élite cuando el nuevo individuo es mejor. En el caso del ne-cGA, el individuo élite es reemplazado cuando el nuevo individuo lo supera, o cuando no ha podido ser reemplazado después de η iteraciones. Nótese que pe-cGA es equivalente a ne-cGA cuando $\eta = \infty$.

El algoritmo 6 muestra cómo funcionan pe-cGA y ne-cGA.

Algoritmo 6 Persistent y Non-Persistent Compact Genetic Algorithms (pe-cGA y ne-cGA)

Entrada: $n > 0$
Salida: Vector solución v

```
θ ← 0
para i = 1 → n hacer {Inicializar v}
  v_i ← 0,5
Generar individuo élite e, por medio de v
mientras no se cumpla condición de paro hacer
  Generar individuo a, por medio de v
  si f(a) < f(e) o θ ≥ η entonces {La comparación θ ≥ η es omitida en pe-cGA}
    w ← a
    l ← e
    e ← a
    θ ← 0
  sino
    w ← e
    l ← a
    θ ← θ + 1
  para i = 1 → n hacer {Afectar distribución de la población}
    si w_i <> l_i entonces
      si w_i = 1 entonces
        v_i ← v_i + 1/p
      sino
        v_i ← v_i − 1/p
devolver v
```

La función principal del ne-cGA es prevenir la convergencia prematura para que la optimización no quede atrapada en un óptimo local. Esto es aplicable en problemas donde un elitismo muy fuerte, como el del pe-cGA, produzca convergencia prematura.

Cabe mencionar que los autores no han podido establecer una relación entre el tipo de problemas de optimización y el algoritmo adecuado, pero sí muestran que el parámetro η no debe ser mayor al tamaño de la población p, y que dicho parámetro permite que el algoritmo se asemeje a algoritmos genéticos que incorporan mecanismos

de migración de individuos. Además, también muestran que cuando se disminuye η se mejora la calidad de las soluciones, aunque puede afectar la velocidad de convergencia.

También en [21], se muestra que el pe-cGA es equivalente a una estrategia evolutiva (1+1) con mutación auto-adaptable. Esto significa que ambos algoritmos poseen una diversidad genética similar, mientras que el pe-cGA ofrece beneficios adicionales, como el ajuste dinámico de ciertas variables utilizadas y la selección y corrección de la distribución de probabilidad multi-variada, necesarias en la estrategia evolutiva auto-adaptable.

3.3. Real-Valued Compact Genetic Algorithm

El *Real-Valued Compact Genetic Algorithm* (rcGA) [22] es otra variante del cGA que lleva lo compacto del dominio binario al dominio real, convirtiéndose en un algoritmo de gran desempeño y poco uso de memoria.

En rcGA, el vector de probabilidad v no es un vector de bits, sino una matriz de dimensión $n \times 2$:

$$V = [\vec{\mu}, \vec{\sigma}] \tag{3.1}$$

donde $\vec{\mu}$ y $\vec{\sigma}$ son vectores de longitud n. Cada posición i de $\vec{\mu}$ y $\vec{\sigma}$ representa la media y desviación estándar, respectivamente, de la variable de decisión x_i del problema a optimizar. Dichos vectores representan una función de distribución de probabilidad (FDP) gaussiana, truncada en el intervalo [-1, 1] y con altura normalizada para tener un área igual a 1.

Al inicio del algoritmo, los valores de $\vec{\mu}$ se inicializan con 0 y $\vec{\sigma}$ con α, donde α es una constante ($\alpha = 10$) para que la FDP funcione como una distribución uniforme que poco a poco converge a una distribución gaussiana. Al igual que en pe-cGA y ne-cGA, se genera un individuo élite que guía la optimización y, en cada iteración, se genera un nuevo individuo, a partir de V, para competir con el élite. Dado que rcGA trata con valores reales en lugar de binarios, las reglas de actualización de V son distintas:

$$\mu_i^{t+1} = \mu_i^t + \frac{1}{p}(w_i - l_i) \tag{3.2}$$

$$(\sigma_i^{t+1})^2 = (\sigma_i^t)^2 + (\mu_i^t)^2 - (\mu_i^{t+1})^2 + \frac{1}{p}(w_i^2 - l_i^2) \tag{3.3}$$

Las variantes persistente y no persistente del rcGA funcionan de igual manera que pe-cGA y ne-cGA, e incluso presentan las mismas características en relación a la calidad de soluciones y velocidad de convergencia. Los autores de rcGA también muestran que usar una variante u otra es dependiente del problema.

El algoritmo 7 muestra el funcionamiento del rcGA.

Algoritmo 7 Real-Valued Compact Genetic Algorithm (rcGA)

Entrada: $n > 0$
Salida: Vector solución $\vec{\mu}$

$t \leftarrow 0$
$\theta \leftarrow 0$
para $i = 1 \rightarrow n$ **hacer** {Inicializar v}
  $\mu_i \leftarrow 0$
  $\sigma_i \leftarrow \alpha$
Generar individuo élite e, por medio de V
mientras no se cumpla condición de paro **hacer**
  Generar individuo a, por medio de V
  si $f(a) < f(e)$ **o** $\theta \geq \eta$ **entonces** {La comparación $\theta \geq \eta$ es omitida en la variante persistente}
    $w \leftarrow a$
    $l \leftarrow e$
    $e \leftarrow a$
    $\theta \leftarrow 0$
  sino
    $w \leftarrow e$
    $l \leftarrow a$
    $\theta \leftarrow \theta + 1$
  para $i = 1 \rightarrow n$ **hacer** {Afectar distribución de la población}
    $\mu_i^{t+1} = \mu_i^t + \frac{1}{p}(w_i - l_i)$
    $\sigma_i^{t+1} = \sqrt{(\sigma_i^t)^2 + (\mu_i^t)^2 - (\mu_i^{t+1})^2 + \frac{1}{p}(w_i^2 - l_i^2)}$
  $t \leftarrow t + 1$
devolver $\vec{\mu}$

Mecanismo de muestreo

Para generar la variable i de un nuevo individuo x a partir de la FDP gaussiana, el rcGA utiliza la siguiente fórmula:

$$FDP(x_i) = \frac{e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}} \sqrt{\frac{2}{\pi}}}{\sigma_i \left(\operatorname{erf}\left(\frac{\mu_i + 1}{\sqrt{2}\sigma_i} \right) - \operatorname{erf}\left(\frac{\mu_i - 1}{\sqrt{2}\sigma_i} \right) \right)} \tag{3.4}$$

donde erf es la función de error [23], y μ_i y σ_i son la media y la desviación estándar, respectivamente, de la variable i. La deducción de la fórmula se detalla en la sección 4.1.

A partir de la FDP, se define su función de distribución acumulada (FDA) a partir de polinomios de Chebyshev, utilizando el procedimiento descrito en [24]. Dado que el co-dominio de la FDA es $[0, 1]$, para obtener una muestra s_i se necesita un número aleatorio $r_i \in [0, 1)$ tomado de una distribución uniforme, el cual se evalúa por la función inversa de la FDA, es decir, $s_i = FDA^{-1}(r_i)$. La muestra obtenida está en el intervalo $[-1, 1]$, por lo que es necesario regresarla al intervalo original $[a, b]$ de la variable, es decir, $x_i = a + \frac{b-a}{2}(s_i + 1)$.

3.4. Compact Differential Evolution

El algoritmo denominado *Compact Differential Evolution* (cDE) [25] surgió al modificar el rcGA para que utilizara los principios de la evolución diferencial, y no los del algoritmo genético.

Dado que el cDE utiliza los mismos principios para el muestreo de los individuos que el rcGA, ambos algoritmos son muy parecidos. Considerando la implementación más común de la evolución diferencial, DE/rand/1/bin, en cada iteración se toman tres individuos x_r, x_s y x_t a partir de v, con los cuales se genera un nuevo individuo x'_{off}. El nuevo individuo se recombina con el individuo élite para, finalmente, generar al individuo x_{off}, el cual competirá contra el élite mismo. Como puede observarse, el cDE y el rcGA sólo difieren en la recombinación y la mutación.

De igual manera que el rcGA, el cDE también tiene dos variantes: con elitismo persistente (pe-cDE) y con elitismo no persistente (ne-cDE). Ambas variantes funcionan igual que en el rcGA y, por lo tanto, que en el cGA.

En el algoritmo 8 puede observarse cómo funcionan las dos variantes del cDE.

El cDE ha demostrado tener muy buen desempeño en una amplia variedad de problemas, incluso siendo comparado con su contraparte: la evolución diferencial poblacional. Algunas razones de su éxito, detalladas en [25], son:

- El cDE genera los nuevos individuos siguiendo directamente los principios de la evolución diferencial, y no sólo por medio de v, aumentando su poder exploratorio.

Algoritmo 8 Compact Differential Evolution (cDE)

Entrada: $n > 0$
Salida: Vector solución $\vec{\mu}$

$t \leftarrow 0$
$\theta \leftarrow 0$
para $i = 1 \rightarrow n$ **hacer** {Inicializar v}
  $\mu_i \leftarrow 0$
  $\sigma_i \leftarrow \alpha$
Generar individuo élite e, por medio de v
mientras no se cumpla condición de paro **hacer**
  Generar tres individuos x_r, x_s y x_t, por medio de v
  $x'_{\text{off}} \leftarrow x_t + F(x_r - x_s)$
  $x_{\text{off}} = x'_{\text{off}}$
  para $i = 1 \rightarrow n$ **hacer**
    $r \leftarrow \text{rand}(0, 1)$ {Generar un número aleatorio uniforme}
    si $r < C_r$ **entonces**
      $x_{\text{off}_i} \leftarrow e_i$
  si $f(x_{\text{off}}) < f(e)$ **o** $\theta \geq \eta$ **entonces** {La comparación $\theta \geq \eta$ es omitida en la variante persistente}
    $w \leftarrow x_{\text{off}}$
    $l \leftarrow e$
    $e \leftarrow x_{\text{off}}$
    $\theta \leftarrow 0$
  sino
    $w \leftarrow e$
    $l \leftarrow x_{\text{off}}$
    $\theta \leftarrow \theta + 1$
  para $i = 1 \rightarrow n$ **hacer** {Afectar distribución de la población}
    $$\mu_i^{t+1} = \mu_i^t + \frac{1}{p}(w_i - l_i)$$
    $$\sigma_i^{t+1} = \sqrt{(\sigma_i^t)^2 + (\mu_i^t)^2 - (\mu_i^{t+1})^2 + \frac{1}{p}(w_i^2 - l_i^2)}$$
  $t \leftarrow t + 1$
devolver $\vec{\mu}$

- A diferencia del cGA y rcGA, el cDE permite implementar, de manera directa, el esquema de supervivencia de los individuos de la evolución diferencial, impidiendo que se degrade.
- Dado que el cDE se basa en una estructura probabilística, no es necesario recaer en la aleatoriedad que imponen los parámetros de control, ya que dicha aleatoriedad es introducida por el cDE directamente en las soluciones generadas por medio de v.
- Aunque no intenta superar a la evolución diferencial, el cDE es una excelente versión ligera, ya que reproduce su desempeño, pero impone menos requerimientos de memoria, reduciendo el consumo de $2p$ individuos a sólo 4.

Capítulo 4

Algoritmo propuesto

En este capítulo se presenta el algoritmo *Multi-Objective Compact Differential Evolution (mocDE)*, el cual se diseñó para resolver problemas de optimización continuos multi-objetivo sin restricciones. Sin embargo, también cuenta con las siguientes características:

- Robustez al definir dos versiones de elitismo para controlar la convergencia: elitismo persistente y elitismo no persistente.
- Ahorro de recursos de memoria al utilizar una representación estadística de la población.
- Buena distribución de las soluciones al usar un archivo basado en el escalamiento de Chebyshev.
- Poder explorativo al implementar la evolución diferencial.

Dichas características se obtienen a partir de implementar algunos principios tomados de los algoritmos descritos en el capítulo 3.

En la sección 4.1 se detalla cómo mocDE representa su población y cómo nuevas soluciones influyen sobre ella. La sección 4.2 define el uso de elitismo y las variantes que se origina con ello. En la sección 4.3 se describe el comportamiento del archivo externo utilizado para mejorar la distribución de las soluciones encontradas. Finalmente, la sección 4.4 muestra el algoritmo obtenido, así como el detalle de su funcionamiento.

4.1. Representación de la población

Al igual que cDE, mocDE opera sobre una representación estadística de la población y no sobre un conjunto de individuos como tal. Esto le permite utilizar sólo cinco vectores de longitud n (cantidad de variables de decisión), en lugar de p (tamaño de la población) vectores, logrando un ahorro de memoria de orden lineal.

La representación estadística de la población, denotada por v, está integrada por dos vectores μ y σ, ambos de longitud n, que representan la media y la desviación

estándar, respectivamente, de cada variable de decisión. Los valores μ_i y σ_i definen la función de distribución de probabilidad (FDP)[1] gaussiana de la variable x_i, truncada en el intervalo [-1, 1] y con área normalizada, como se muestra a continuación:

$$\text{FDP}(x_i) = \frac{e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}\sqrt{\frac{2}{\pi}}}{\sigma_i\left(\text{erf}\left(\frac{\mu_i+1}{\sqrt{2}\sigma_i}\right) - \text{erf}\left(\frac{\mu_i-1}{\sqrt{2}\sigma_i}\right)\right)} \tag{4.1}$$

A partir de la FDP, se obtiene su función de distribución acumulada (FDA)[2], la cual, al ser invertida, permite tomar una muestra s_i a partir de un número aleatorio tomado de una distribución uniforme. A continuación, se denota la forma general de FDA^{-1}:

$$\text{FDA}^{-1}(\mu_i, \sigma_i) = \mu_i - \sqrt{2}\sigma_i\ \text{erf}^{-1}\left((x-C)\left(\text{erf}\left(\frac{\mu_i-1}{\sqrt{2}\sigma_i}\right) - \text{erf}\left(\frac{\mu_i+1}{\sqrt{2}\sigma_i}\right)\right)\right) \tag{4.2}$$

donde erf es la función de error, y C es la constante de integración obtenida al calcular la FDA:

$$C = -\text{FDA}(-1) = -\frac{\text{erf}\left(\frac{\mu_i+1}{\sqrt{2}\sigma_i}\right)}{\text{erf}\left(\frac{\mu_i-1}{\sqrt{2}\sigma_i}\right) - \text{erf}\left(\frac{\mu_i+1}{\sqrt{2}\sigma_i}\right)} \tag{4.3}$$

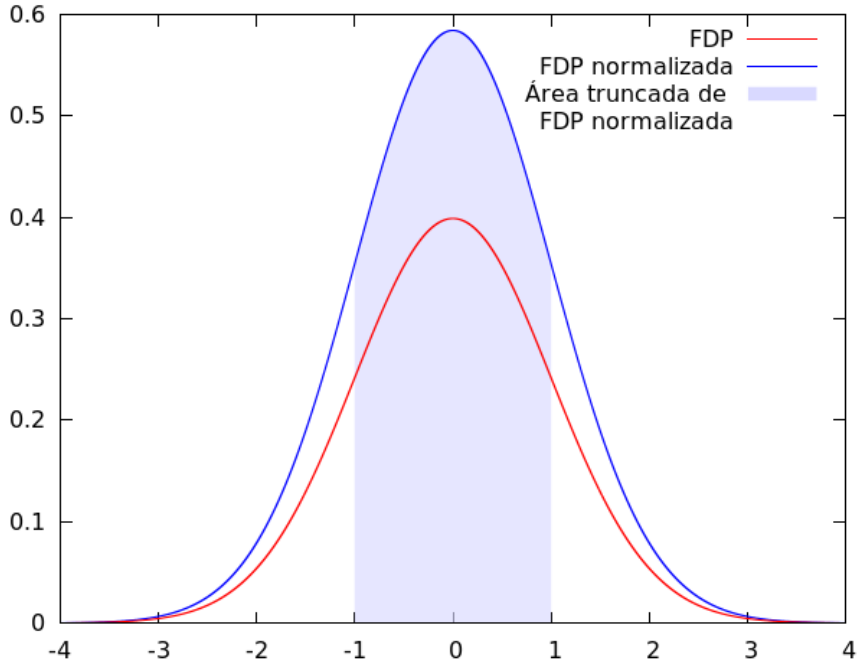
En la figura 4.1, se puede observar la normalización de dos curvas gaussianas truncadas para obtener su FDP. Mientras que en la figura 4.2, se muestra un ejemplo de una FDP y su FDA correspondiente.

Dado que la muestra s_i obtenida está en el intervalo $[-1, 1]$, es necesario regresarla al rango intervalo $[a, b]$ de la variable, es decir, $x_i' = a + \frac{b-a}{2}(s_i + 1)$.

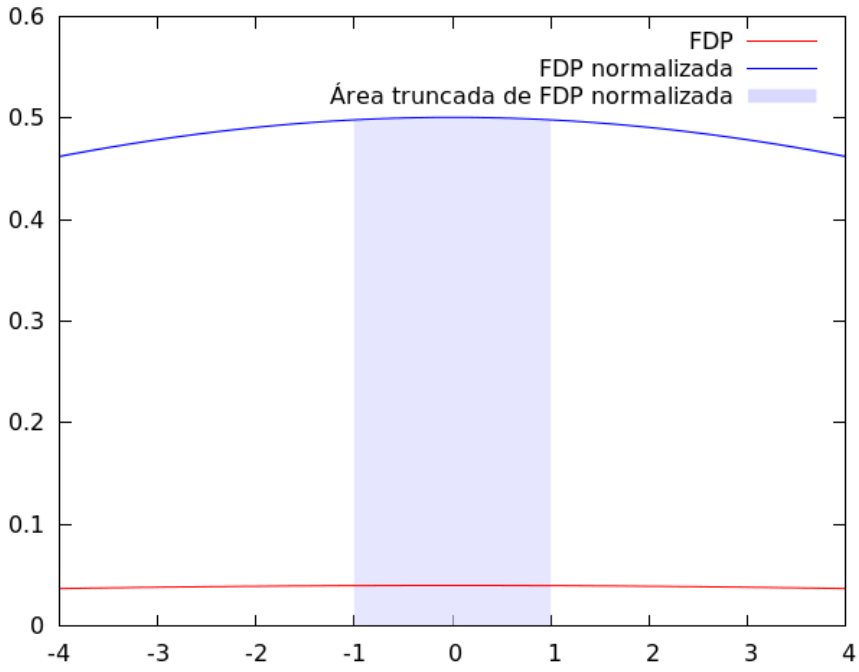
Durante el proceso de optimización, mocDE mantiene una solución élite y, en cada iteración, genera una nueva solución para ser comparadas entre sí. Después, la solución vencedora es agregada a la población, es decir, afecta a su representación estadística.

Supongamos que t es la iteración actual, w es la solución vencedora, l la solución perdedora, x una solución que ya es parte de la población y w reemplazará a l en la

[1] Conocida como PDF por sus siglas en inglés.
[2] Conocida como CDF por sus siglas en inglés.

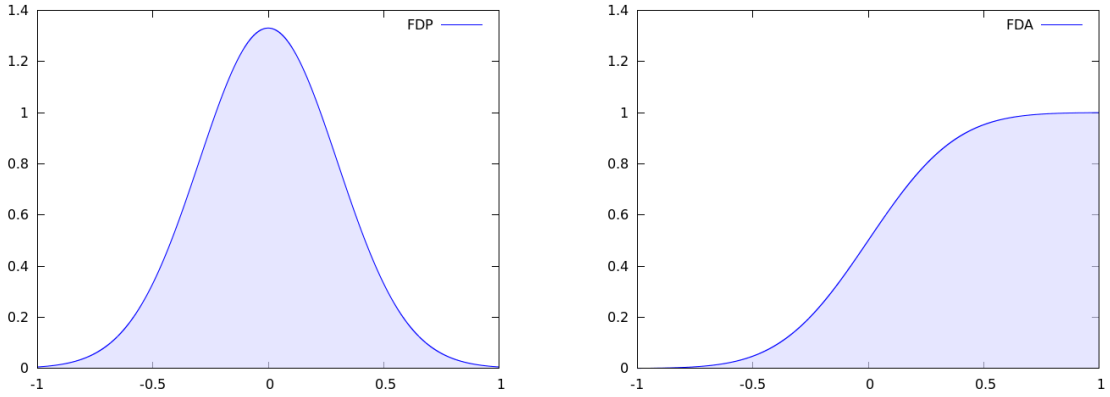


(a)



(b)

Figura 4.1: Normalización de curvas gaussianas truncadas en $[-1, 1]$, con $\mu = 0$ y (a) $\sigma = 1$ y (b) $\sigma = 10$.



(a) (b)

Figura 4.2: Correspondencia de una (a) función de distribución de probabilidad (con $\mu = 0$ y $\sigma = 0{,}3$) y su (b) función de distribución acumulada.

población, entonces tenemos que la media se calcula de la siguiente manera:

$$\mu_i^t = \frac{1}{p}(x_{1_i} + ... + x_{p-1_i} + l_i) \tag{4.4}$$

$$\mu_i^{t+1} = \frac{1}{p}(x_{1_i} + ... + x_{p-1_i} + w_i) \tag{4.5}$$

$$= \mu_i^t + \frac{1}{p}(w_i - l_i) \tag{4.6}$$

donde p es el tamaño de la población.

Bajo las mismas condiciones, el cálculo de la desviación estándar se realiza como

sigue:

$$\sigma_i^t = \sqrt{\frac{(x_{1_i} - \mu^t)^2}{p} + ... + \frac{(x_{p-1_i} - \mu^t)^2}{p} + \frac{(l_i - \mu^t)^2}{p}} \tag{4.7}$$

$$\sigma_i^{t+1} = \sqrt{\frac{(x_{1_i} - \mu^{t+1})^2}{p} + ... + \frac{(x_{p-1_i} - \mu^{t+1})^2}{p} + \frac{(w_i - \mu^{t+1})^2}{p}} \tag{4.8}$$

$$= \sqrt{\sum_{k=1}^{p-1} \frac{(x_{k_i} - \mu^{t+1})^2}{p} + \frac{(w_i - \mu^{t+1})^2}{p}} \tag{4.9}$$

$$= \sqrt{\sum_{k=1}^{p-1} \frac{(x_{k_i} - (\mu^t + \frac{1}{p}(w_i - l_i)))^2}{p} + \frac{(w_i - (\mu^t + \frac{1}{p}(w_i - l_i)))^2}{p}} \tag{4.10}$$

$$= \sqrt{(\sigma_i^t)^2 + (\mu_i^t)^2 - (\mu_i^{t+1})^2 + \frac{1}{p}(w_i^2 - l_i^2)} \tag{4.11}$$

4.2. Elitismo

En [21] se muestra que al incorporar elitismo en el proceso de optimización se obtiene un mejor desempeño. Por lo tanto, mocDE también implementa los mismos tipos de elitismo: persistente y no persistente. Ésto origina dos variantes: *Persistent Elitist Multi-Objective Compact Differential Evolution* (pe-mocDE) y *Non-Persistent Elitist Multi-Objective Compact Differential Evolution* (ne-mocDE).

Ambas variantes incorporan un individuo extra llamado élite, el cual es utilizado, durante el proceso de optimización, para ser comparado contra la nueva solución generada y determinar si dicha solución mejora a la población actual o no. Si la solución élite es mejor que la nueva solución, ésta afecta a la representación estadística de la población, sino la nueva solución la afecta y reemplaza a la solución élite. Dicha sustitución sólo se lleva a cabo cuando:

- La nueva solución domina a la solución élite.
- Ambas soluciones son no dominadas entre sí pero la nueva solución se puede agregar al archivo externo (véase sección 4.3).
- Han pasado η iteraciones desde la última vez que se actualizó la solución élite (en el caso de ne-mocDE).

Como se puede ver, la gran diferencia entre ambas variantes es sólo el último caso de sustitución. Esto mismo origina que ne-mocDE necesite de un parámetro de inicialización extra η, el cual debe ser menor al tamaño de la población, es decir, $\eta < p$ de acuerdo a [21]. Nótese que pe-mocDE es equivalente a ne-mocDE cuando $\eta = \infty$.

La función principal de ne-mocDE es prevenir la convergencia prematura para que la optimización no quede atrapada en un óptimo local. Esto es aplicable en problemas donde un elitismo muy fuerte, como el de pe-mocDE, produce convergencia prematura.

En [21] no se concluye cuándo es mejor utilizar la variante persistente o la no persistente. Sin embargo, dichos algoritmos resuelven problemas de optimización de un solo objetivo, por lo que el capítulo 5 vuelve a comparar ambas variantes pero con problemas multi-objetivo.

4.3. Archivo de soluciones no dominadas

Para mantener una buena distribución de las soluciones obtenidas, mocDE implementa un archivo externo con un esquema parecido al de MOEA/D. Dicho archivo utiliza el escalamiento ponderado de Chebyshev, donde los vectores de pesos $\lambda_i, i = 1, 2, ..., p$ son definidos por el usuario y representan la distribución deseada. Este archivo ayuda a determinar cuándo una solución mejora la distribución y debe ser declarada ganadora al momento de influir en la población. Dicho escalamiento tiene la gran característica de poder generar todas las soluciones óptimas de Pareto [3].

El escalamiento de Chebyshev se realiza de la siguiente manera:

$$g_i(x) = \max_{1 \leq j \leq k} \lambda_{i,j}(f_j(x) - z_j) \tag{4.12}$$

donde z es un vector de referencia, y λ es un conjunto de vectores que definen la distribución deseada.

El vector de referencia z es el vector ideal encontrado hasta el momento, es decir, $z_i = \min f_i(x)$ para toda x generada durante el proceso de optimización.

El tamaño máximo del archivo debe ser definido por el usuario, y representa la cantidad de soluciones que se desea como resultado.

En el algoritmo 9 se observa cómo se archiva una nueva solución x al archivo A utilizando el vector de referencia z, regresando *cierto* si se agregó y *falso* en caso contrario.

Algoritmo 9 Función *archivar* de mocDE

Entrada: $n > 0 \wedge k > 1 \wedge p > 0 \wedge x \in \mathbb{R}^n \wedge z \in \mathbb{R}^k \wedge f : \mathbb{R}^n \to \mathbb{R}^k \wedge A \in \mathbb{R}^{p \times n} \wedge \lambda \in \mathbb{R}^{p \times k}$
Salida: Si se agregó la solución x al archivo A

```
para i = 1 → k hacer {Actualizar vector ideal}
  si f_i(x) < z_i entonces
    z_i ← f_i(x)
agregado ← falso
para i = 1 → p hacer {Actualizar archivo}
  f_1 ← máx_{1≤j≤k} λ_{i,j}|f_j(x) − z_j|
  f_2 ← máx_{1≤j≤k} λ_{i,j}|f_j(A_i) − z_j|
  si f_1 < f_2 entonces {Si x es mejor que A_i}
    agregado ← cierto
    A_i ← x
devolver agregado
```

4.4. Algoritmo

Al inicio del algoritmo, los valores de μ se inicializan con 0 y σ con α, donde α es una constante ($\alpha = 10$) que hace que la FDP funcione como una distribución uniforme que va convergiendo hacia una distribución gaussiana. También se crea un individuo élite e inicial que guiará el proceso de optimización, y que será el primero en incluirse en el archivo de soluciones no dominadas e inicializará al vector ideal z, el cual será la referencia para el escalamiento de Chebyshev.

En cada iteración, mocDE utiliza la implementación estándar de la evolución diferencial DE/rand-to-best/1/bin, aunque puede reemplazarse por cualquier otra variante, tomando tres individuos x_r, x_s y x_t a partir de v, para generar un nuevo individuo x'_{off}. El nuevo individuo se recombina con el individuo élite para generar al individuo x_{off}, el cual competirá contra el élite mismo.

Para decidir si x_{off} es mejor que e, se determina si $f(x_{\text{off}}) \prec f(e)$. En caso de que ambas soluciones sean no dominadas entre sí, x_{off} es el vencedor si es posible agregarlo al archivo, verificando que exista algún $i \in \{1, .., p\}$ tal que $g_i(A_i) > g_i(x_{\text{off}})$ (vea ecuación 4.12).

De igual manera que cDE, mocDE también tiene dos variantes: con elitismo persistente (pe-mocDE) y con elitismo no persistente (ne-mocDE). Esto permite prevenir la convergencia prematura, y es aplicable donde un elitismo muy fuerte disminuye la velocidad de convergencia, o incluso evita que la población converja. La versión ne-mocDE reemplaza a la solución élite cuando sobrevive η iteraciones.

Una vez determinada la solución ganadora, se procede a actualizar la población y continuar con la siguiente iteración (vea ecuaciones 4.6 y 4.11).

El algoritmo propuesto se puede ver de manera general en el algoritmo 10, donde α es usualmente igual a 10, y la función *archivar(A, z, x)* fue descrita en el algoritmo

9.

Algoritmo 10 Multi-Objective Compact Differential Evolution (mocDE)

Entrada: $n > 0 \wedge k > 1 \wedge p > 0 \wedge C_r > 0 \wedge F > 0 \wedge \eta > 0 \wedge f : \mathbb{R}^n \rightarrow \mathbb{R}^k \wedge \lambda \in \mathbb{R}^{p \times k}$
Salida: Conjunto de vectores solución A

$t \leftarrow 0$ {Iteración actual}
$\theta \leftarrow 0$ {Iteraciones sobrevividas por la solución élite}
para $i = 1 \rightarrow n$ **hacer** {Inicializar $V = \{\mu, \sigma\}$}
  $\mu_i \leftarrow 0$ {Media}
  $\sigma_i \leftarrow \alpha$ {Desviación estándar}
Generar individuo élite e, por medio de v
Inicializar archivo de soluciones no dominadas $A \leftarrow \{e\}^p, z \leftarrow f(e)$
mientras no se cumpla condición de paro **hacer**
  Generar tres individuos x_r, x_s y x_t, por medio de V
  $x'_{\text{off}} \leftarrow x_t + F \cdot (x_r - x_s) + F \cdot (e - x_t)$ {Aplicar mutación}
  para $i = 1 \rightarrow n$ **hacer** {Realizar cruza}
    $r \leftarrow \text{rand}(0, 1)$ {Generar un número aleatorio uniforme}
    si $r < C_r$ **entonces**
      $x_{\text{off}_i} \leftarrow e_i$
    sino
      $x_{\text{off}_i} \leftarrow x'_{\text{off}_i}$
  si $f(x_{\text{off}}) \prec f(e)$ **o** $\theta \geq \eta$ **entonces** {La comparación $\theta \geq \eta$ es omitida en la variante persistente}
    $w \leftarrow x_{\text{off}}$ {Ganador}
    $l \leftarrow e$ {Perdedor}
    $e \leftarrow x_{\text{off}}$
    $\theta \leftarrow 0$
    archivar(A, z, x_{off}) {Agregar al archivo}
  sino
    si $f(e) \not\prec f(x_{\text{off}})$ **and** archivar(A, z, x_{off}) **entonces** {Si son no dominadas entre sí y se puede agregar al archivo}
      $w \leftarrow x_{\text{off}}$ {Ganador}
      $l \leftarrow e$ {Perdedor}
      $e \leftarrow x_{\text{off}}$
      $\theta \leftarrow 0$
    sino
      $w \leftarrow e$ {Ganador}
      $l \leftarrow x_{\text{off}}$ {Perdedor}
      $\theta \leftarrow \theta + 1$
  para $i = 1 \rightarrow n$ **hacer** {Actualizar representación de la población}

$$\mu_i^{t+1} = \mu_i^t + \frac{1}{p}(w_i - l_i)$$

$$\sigma_i^{t+1} = \sqrt{(\sigma_i^t)^2 + (\mu_i^t)^2 - (\mu_i^{t+1})^2 + \frac{1}{p}(w_i^2 - l_i^2)}$$

  $t \leftarrow t + 1$
devolver A

Capítulo 5

Resultados

A lo largo de este capítulo, se presentarán los resultados obtenidos entre distintas variantes de mocDE, definidas en la sección 4.4 (en la página 35), y los algoritmos del estado del arte, descritos en la sección 2.4.3 (en la página 15). En la sección 5.1, se muestran los resultados obtenidos al comparar las siguientes variantes de mocDE: pe-mocDE, ne-mocDE$_{10}$ ($\eta = 10$), ne-mocDE$_{20}$ ($\eta = 20$) y ne-mocDE$_{50}$ ($\eta = 50$). Finalmente, en la sección 5.2 se compara pe-mocDE con PAES, MOEA/D y NSGA-II. Los resultados han sido medidos con los indicadores que se describen en el apéndice A (en la página 73).

Para realizar la comparación entre los algoritmos anteriormente mencionados, se utilizaron las familias de problemas Zitzler-Deb-Thiele (ZDT) [26] y Deb-Thiele-Laumanns-Zitzler (DTLZ) [27], detalladas en el apéndice B (en la página 77).

5.1. Comparación entre las variantes de mocDE

En esta sección se muestran los resultados obtenidos al comparar distintas variantes de mocDE, con la finalidad de determinar las fortalezas y debilidades que cada uno presenta en distintos tipos de problemas. Estas variantes han sido previamente descritas en el capítulo 4.

Los resultados se obtuvieron al realizar 30 ejecuciones independientes, de cada una de las variantes, con un máximo de 20,000 evaluaciones y usando los parámetros detallados en la tabla 5.1. Los resultados gráficos mostrados de cada algoritmo corresponden a la ejecución que se encuentra en la mediana de las 30 ejecuciones independientes, de acuerdo a la distancia de *Hausdorff* promedio (I_{Δ_p}).

Las siguientes subsecciones condensan los resultados obtenidos entre las variantes de mocDE. Los resultados completos pueden observarse en el apéndice C.1 (en la página 85).

Algoritmo	Parámetro	*Valor*
pe-mocDE	Tamaño de población p	100
	Variación diferencial F	1,0
	Probabilidad de cruza C	0,1
ne-mocDE$_{50}$	Tamaño de población p	100
	Variación diferencial F	1,0
	Probabilidad de cruza C	0,1
	Máxima sobrevivencia del individuo élite η	50
ne-mocDE$_{20}$	Tamaño de población p	100
	Variación diferencial F	1,0
	Probabilidad de cruza C	0,1
	Máxima sobrevivencia del individuo élite η	20
ne-mocDE$_{10}$	Tamaño de población p	100
	Variación diferencial F	1,0
	Probabilidad de cruza C	0,1
	Máxima sobrevivencia del individuo élite η	10

Tabla 5.1: Parámetros de las variantes de mocDE.

5.1.1. Problemas de prueba ZDT

La variante pe-mocDE ha mostrado ser muy superior con respecto a las diferentes variantes de ne-mocDE para ZDT1, ZDT2, ZDT3 y ZDT4, obteniendo el mejor resultado para cada uno de los indicadores utilizados. En dichos problemas es posible apreciar cómo los resultados mejoran al aumentar el valor de η para las variantes de ne-mocDE, indicando que se necesita de una alta fuerza elitista.

El único problema que muestra resultados un poco distintos es ZDT6, en el cual pe-mocDE es mejor en los indicadores de distancia generacional invertida e hipervolumen, mientras que ne-mocDE$_{50}$ vence en los demás (véase figura 5.1). Además, el resultado gráfico de ambas variantes es muy similar, como se puede ver en la figura 5.2.

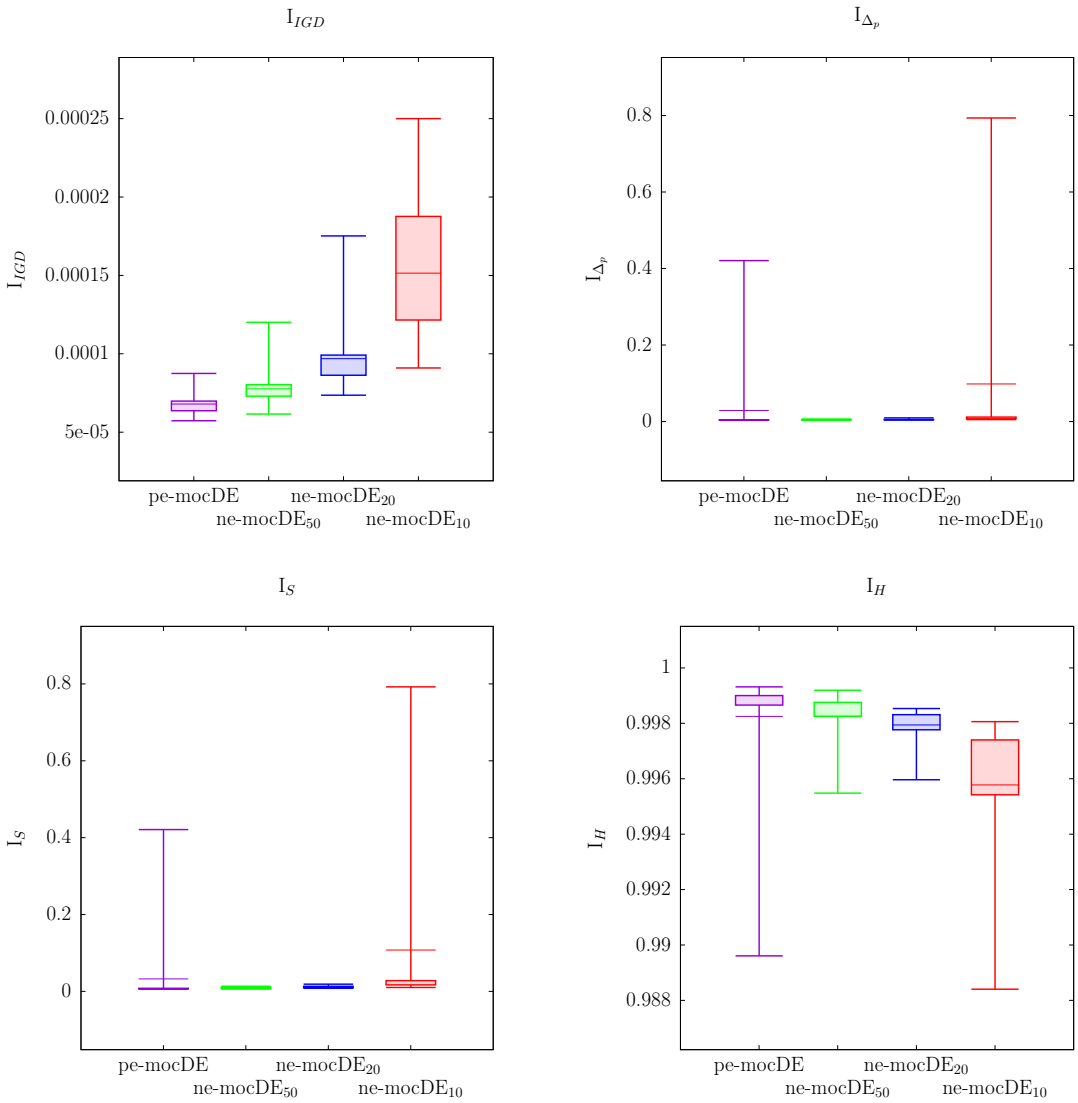


Figura 5.1: Indicadores obtenidos por las variantes de mocDE en el problema ZDT6, donde pe-mocDE no resulta ser el vencedor, dada su inestabilidad, pero sí altamente competitivo.

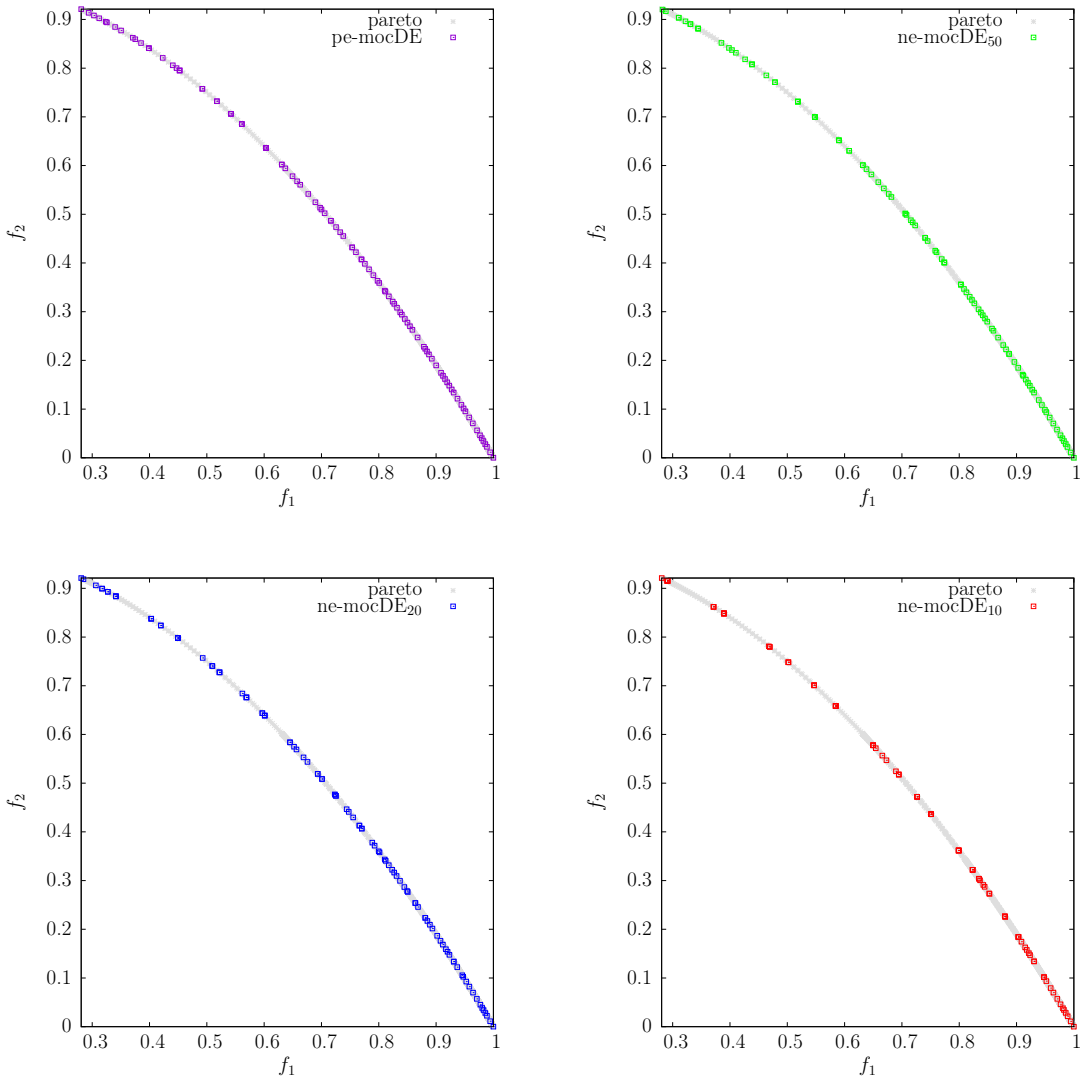


Figura 5.2: Ejecución de las variantes de mocDE al resolver el problema ZDT6, donde pe-mocDE muestra un resultado muy competitivo con su contraparte ne-mocDE$_{50}$.

5.1.2. Problemas de prueba DTLZ

Al igual que en los problemas de prueba ZDT, la variante pe-mocDE ha mostrado ser muy superior en contra de las demás variantes, obteniendo el mejor resultado para cada uno de los indicadores utilizados en DTLZ1, DTLZ2, DTLZ3, DTLZ4 y DTLZ5. De nueva cuenta, es posible apreciar cómo los resultados mejoran al aumentar el valor de η para las variantes de ne-mocDE, indicando que se necesita de una alta fuerza elitista en estos problemas.

El único problema que muestra resultados algo diferentes es DTLZ7, en el cual pe-mocDE y ne-mocDE$_{50}$ tienen resultados muy competitivos (véase figura 5.3). Incluso su respectivo resultado gráfico es muy similar, como se puede ver en la figura 5.4.

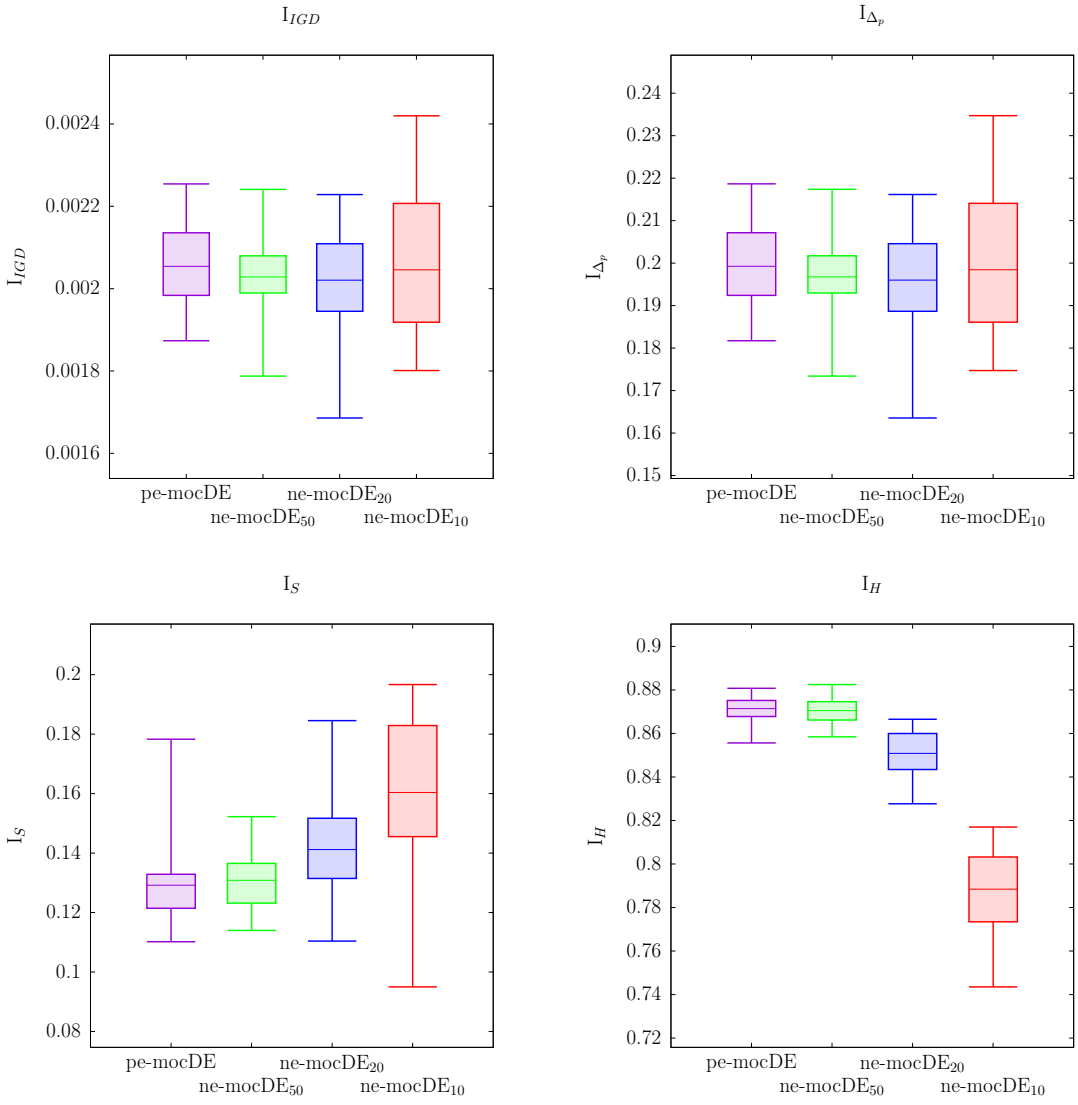


Figura 5.3: Indicadores obtenidos por las variantes de mocDE en el problema DTLZ7, donde pe-mocDE muestra resultados competitivos con el resto de las variantes.

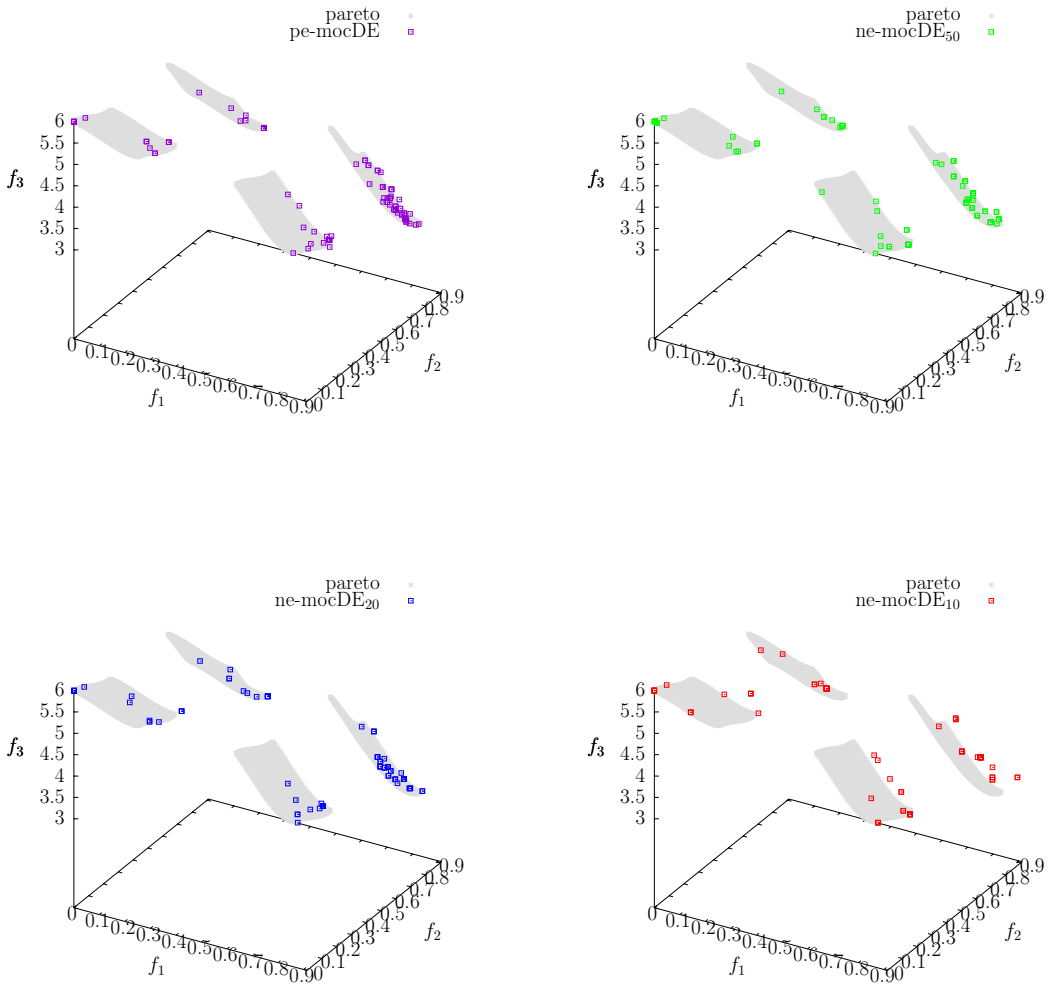


Figura 5.4: Ejecución de las variantes de mocDE al resolver el problema DTLZ7, donde pe-mocDE muestra un resultado muy competitivo con su contraparte ne-mocDE$_{50}$

Algoritmo	Parámetro	*Valor*
mocDE	Variante	pe-mocDE
	Tamaño de población p	100
	Variación diferencial F	1,0
	Probabilidad de cruza C	0,1
NSGA-II	Tamaño de población p	100
	Tasa de mutación m	0,01
	Probabilidad de cruza C	0,9
MOEA/D	Tamaño de población p	100
	Tamaño de nicho	100(2D), 150(3D)
	Límite de actualización	10(2D), 15(3D)
	Variación diferencial F	1,0
	Probabilidad de cruza C	0,5
	Tasa de mutación m	1,0/(número de variables)
	Probabilidad de selección de apareamiento	0,9
PAES	Tamaño de archivo p	100
	Tasa de mutación m	1,0/(número de variables)
	Bisección	5
	Índice de distribución	20

Tabla 5.2: Parámetros de mocDE y los algoritmos del estado del arte.

5.2. Comparación entre mocDE y el estado del arte

En esta sección se muestran los resultados obtenidos al comparar mocDE, NSGA-II, MOEA/D y PAES. Estos algoritmos han sido detallados anteriormente en las secciones 4.4 y 2.4.3.

En la sección 5.1 se determinó que la variante pe-mocDE es más prometedora que el resto de las variantes comparadas, por lo que será la que se utilice para comparar mocDE con el estado del arte.

Los resultados se obtuvieron al realizar 30 ejecuciones independientes, de cada uno de los algoritmos, con un máximo de 20,000 evaluaciones y los parámetros detallados en la tabla 5.2. Los resultados gráficos mostrados de cada algoritmo corresponden a la ejecución que se encuentra en la mediana de las 30 ejecuciones independientes, de acuerdo a la distancia de *Hausdorff* promedio (I_{Δ_p}).

Las siguientes subsecciones condensan los resultados obtenidos entre mocDE y los algoritmos del estado del arte. Los resultados completos pueden observarse en el apéndice C.2.

5.2.1. Problemas de prueba ZDT

En los problemas ZDT1 y ZDT2, mocDE y NSGA-II muestran los mejores resultados de acuerdo a los indicadores unarios (véase figuras 5.5, 5.6, 5.7 y 5.8). Sin

embargo, los binarios determinan que mocDE es definitivamente mejor que NSGA-II.

En ZDT3, NSGA-II tiene los mejores resultados, seguido por mocDE, de acuerdo a los indicadores unarios y épsilon aditivo (I_ϵ^+) (véase figuras 5.9 y 5.10). No obstante, tanto el indicador de cobertura (I_C) como el indicador épsilon multiplicativo (I_ϵ^*) denotan a mocDE como vencedor frente a NSGA-II.

En el problema ZDT4, el algoritmo con mejores resultados es MOEA/D, siendo mejor que mocDE. Sin embargo, mocDE resulta competitivo con respecto a PAES, el cual muestra ser muy inestable, pero mejor que NSGA-II (véase figura 5.11). Sin embargo, el resultado gráfico de PAES es mejor que el de mocDE (véase figura 5.12).

En ZDT6, mocDE vuelve a mostrar los mejores resultados, pero con cierta inestabilidad (véase figuras 5.13 y 5.14).

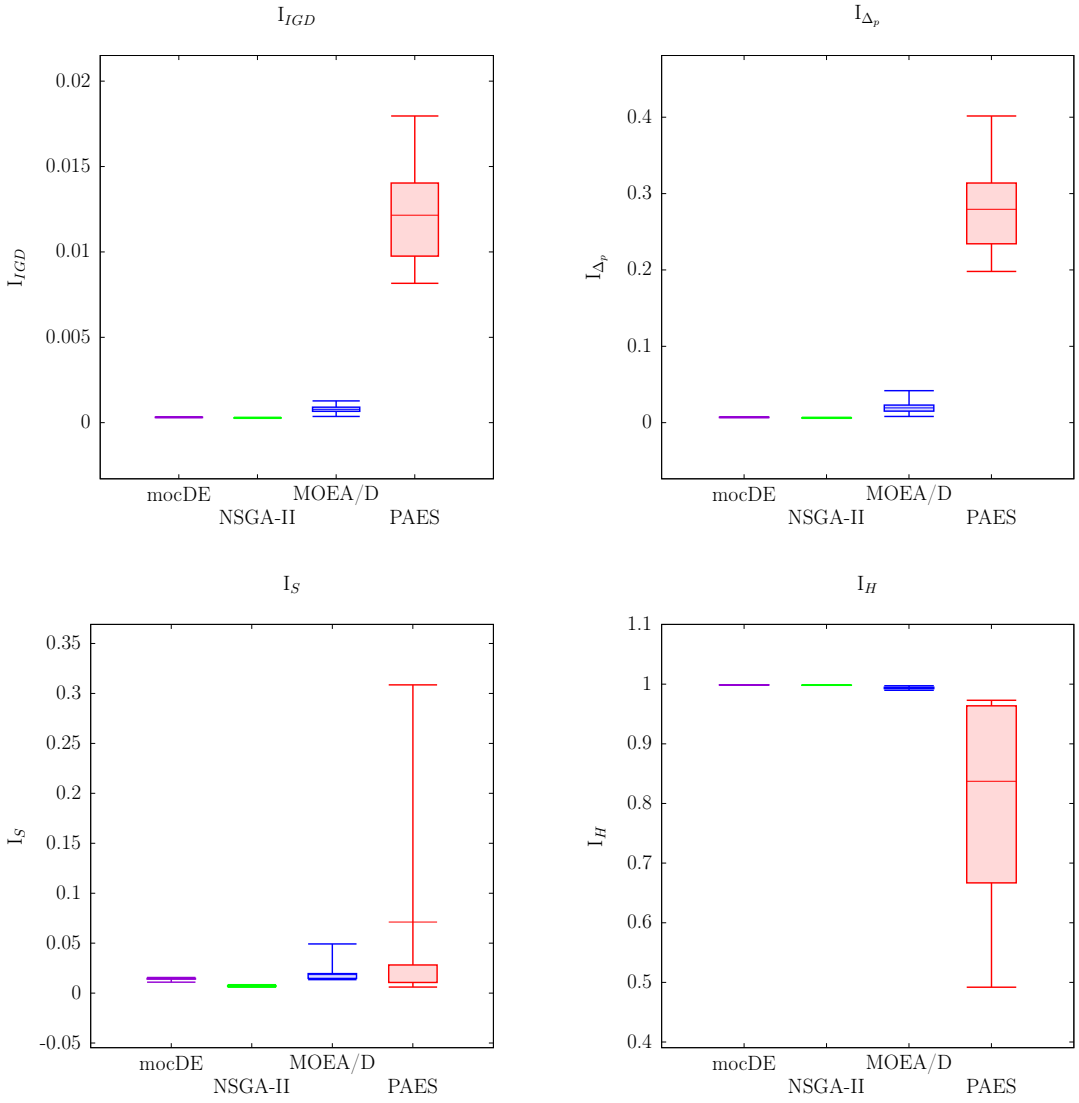


Figura 5.5: Indicadores en el problema ZDT1, donde mocDE muestra los mejores resultados.

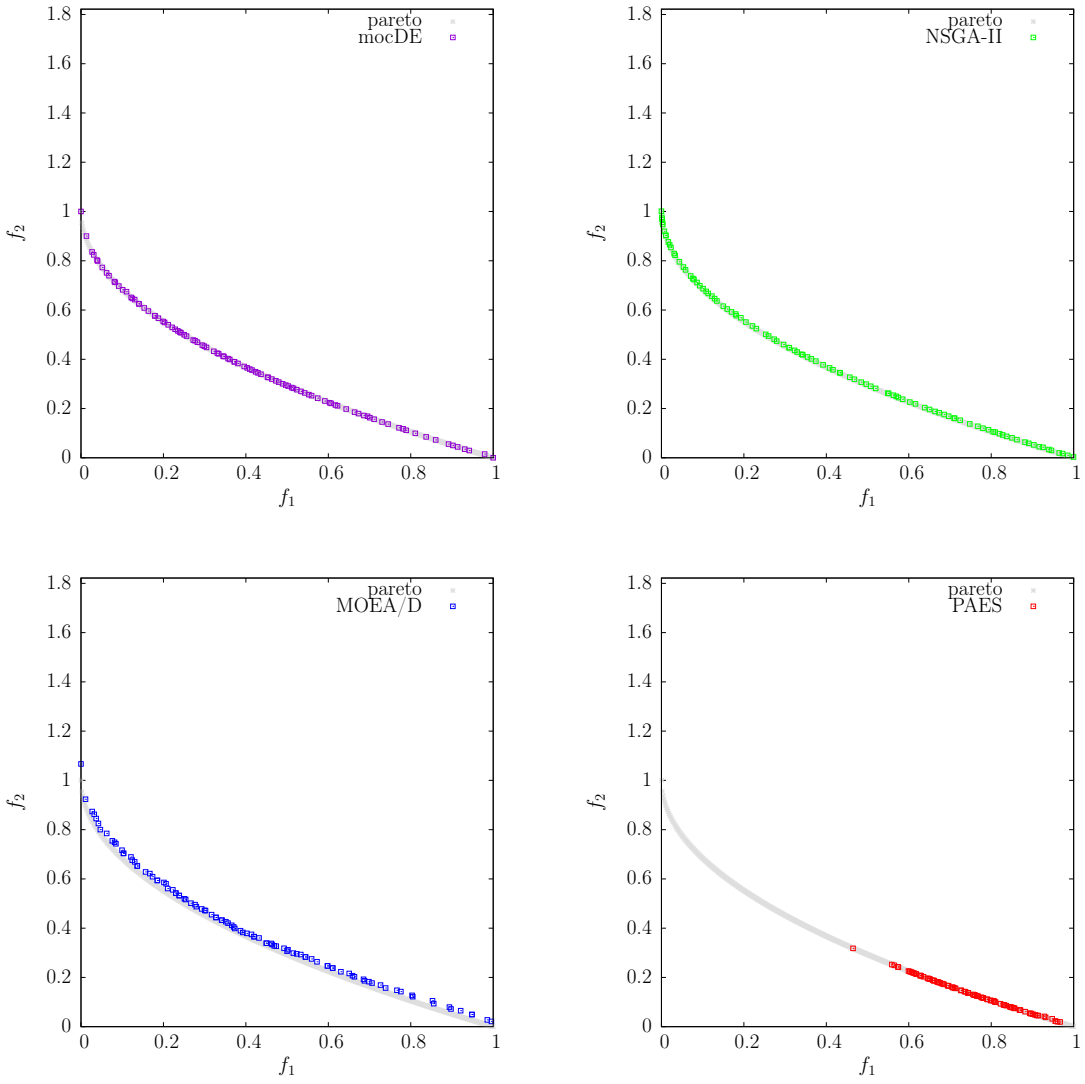


Figura 5.6: Ejecución de todos los algoritmos al resolver el problema ZDT1, donde mocDE muestra un resultado muy competitivo con NSGA-II pero mejor que los demás algoritmos.

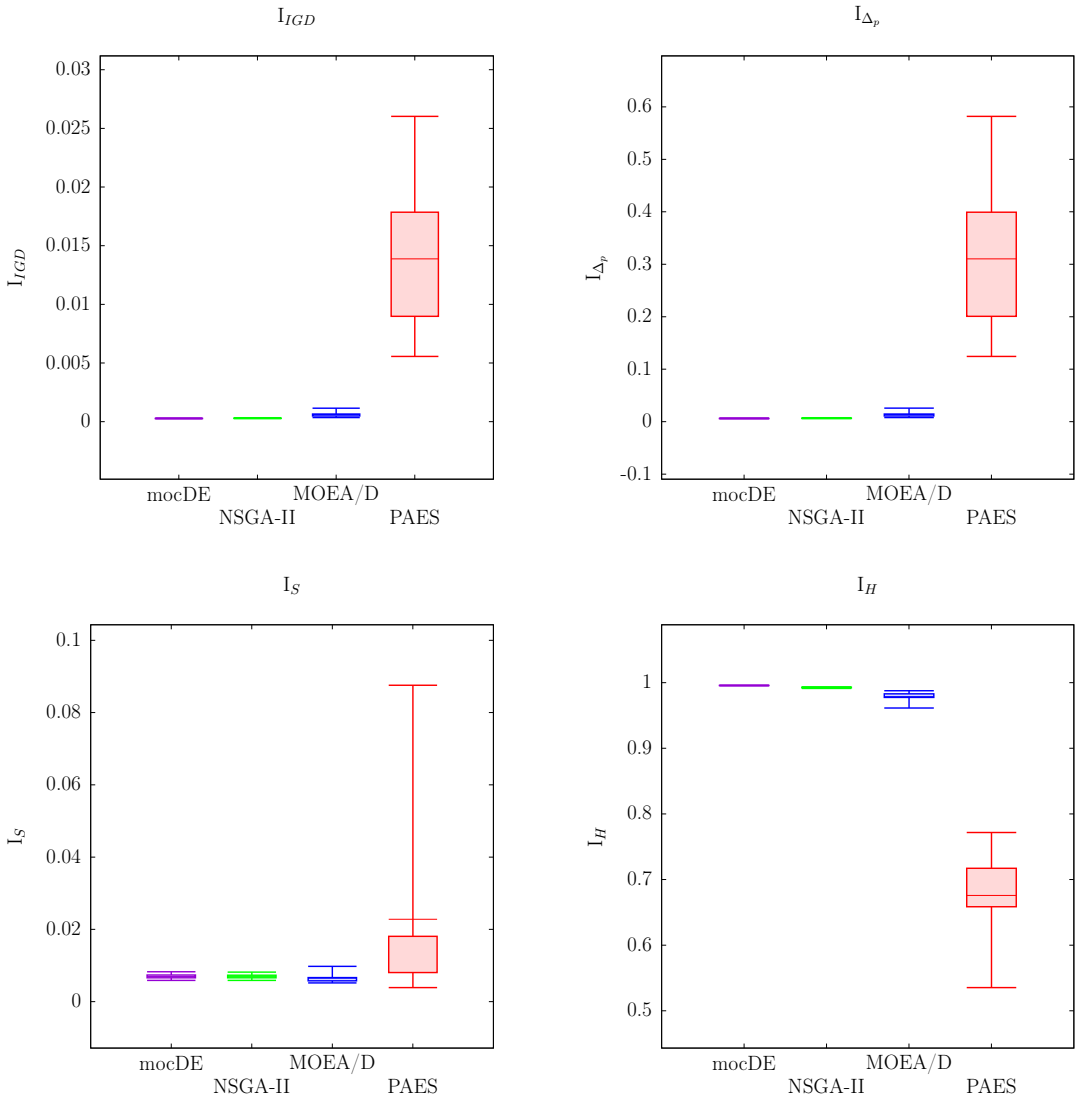


Figura 5.7: Indicadores en el problema ZDT2, donde mocDE muestra mejores resultados que los demás algoritmos.

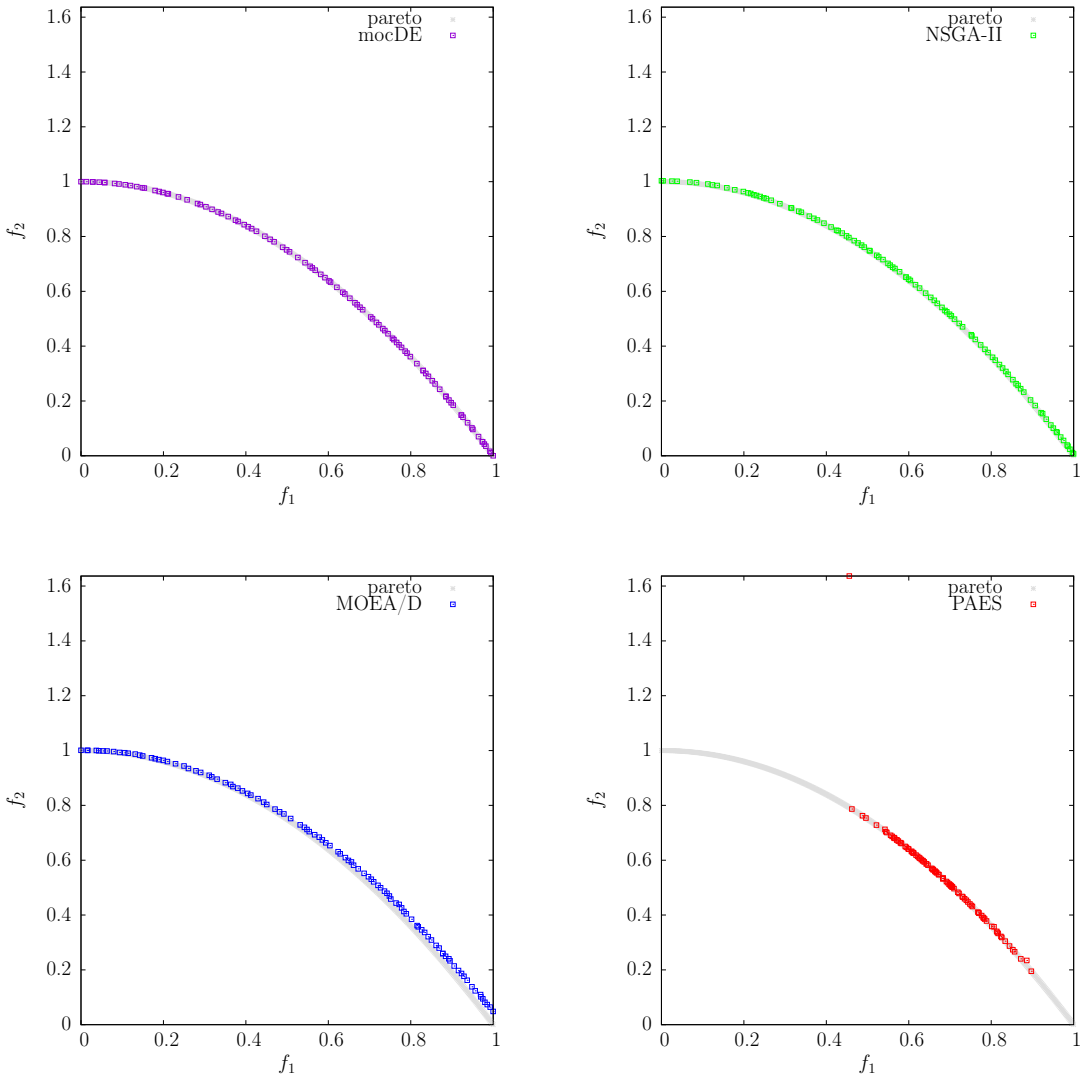


Figura 5.8: Ejecución de todos los algoritmos al resolver el problema ZDT2, donde mocDE muestra un resultado muy competitivo con NSGA-II pero mejor que los demás algoritmos.

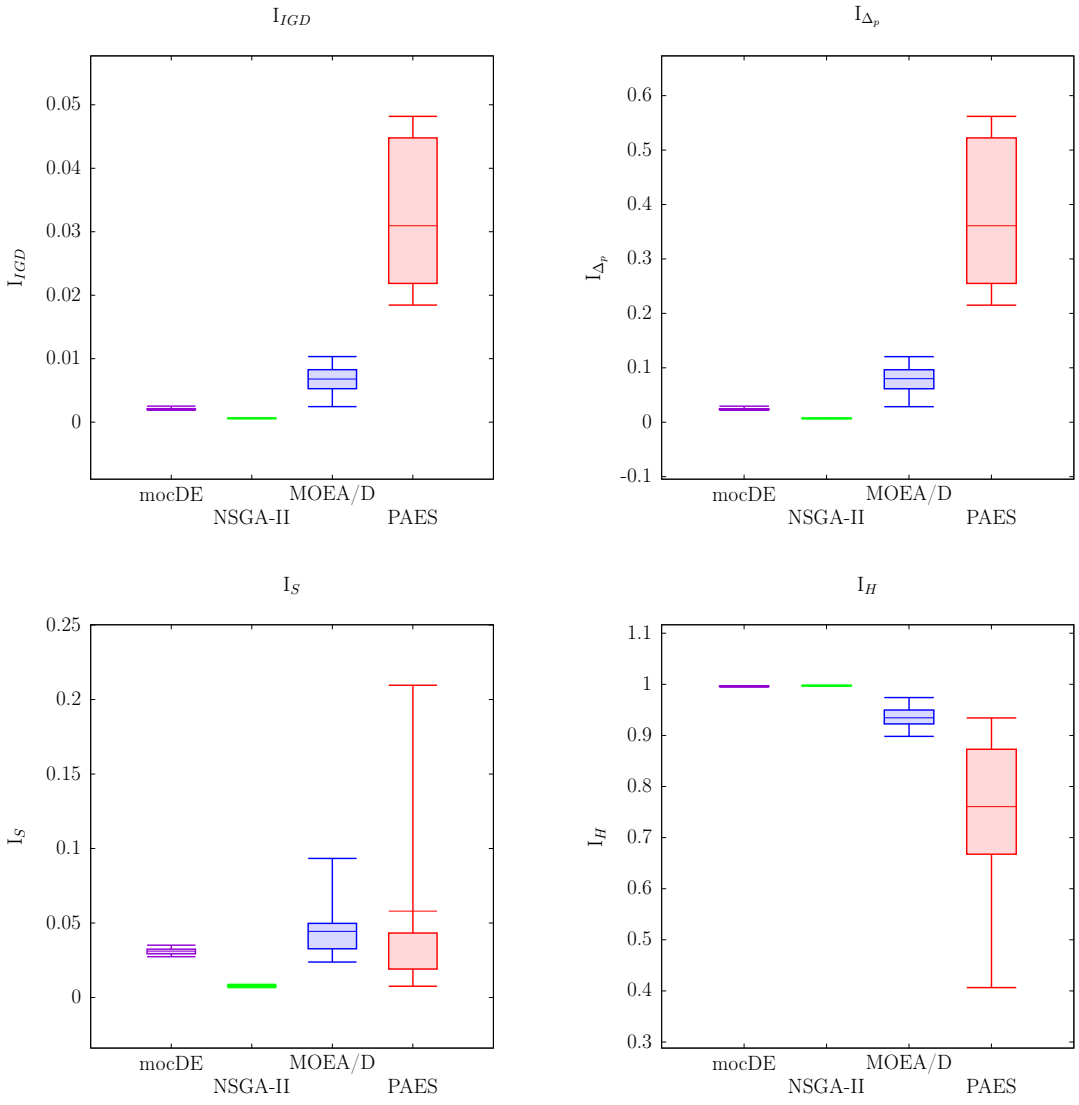


Figura 5.9: Indicadores en el problema ZDT3, donde mocDE muestra resultados inferiores que NSGA-II pero mejores que los demás algoritmos.

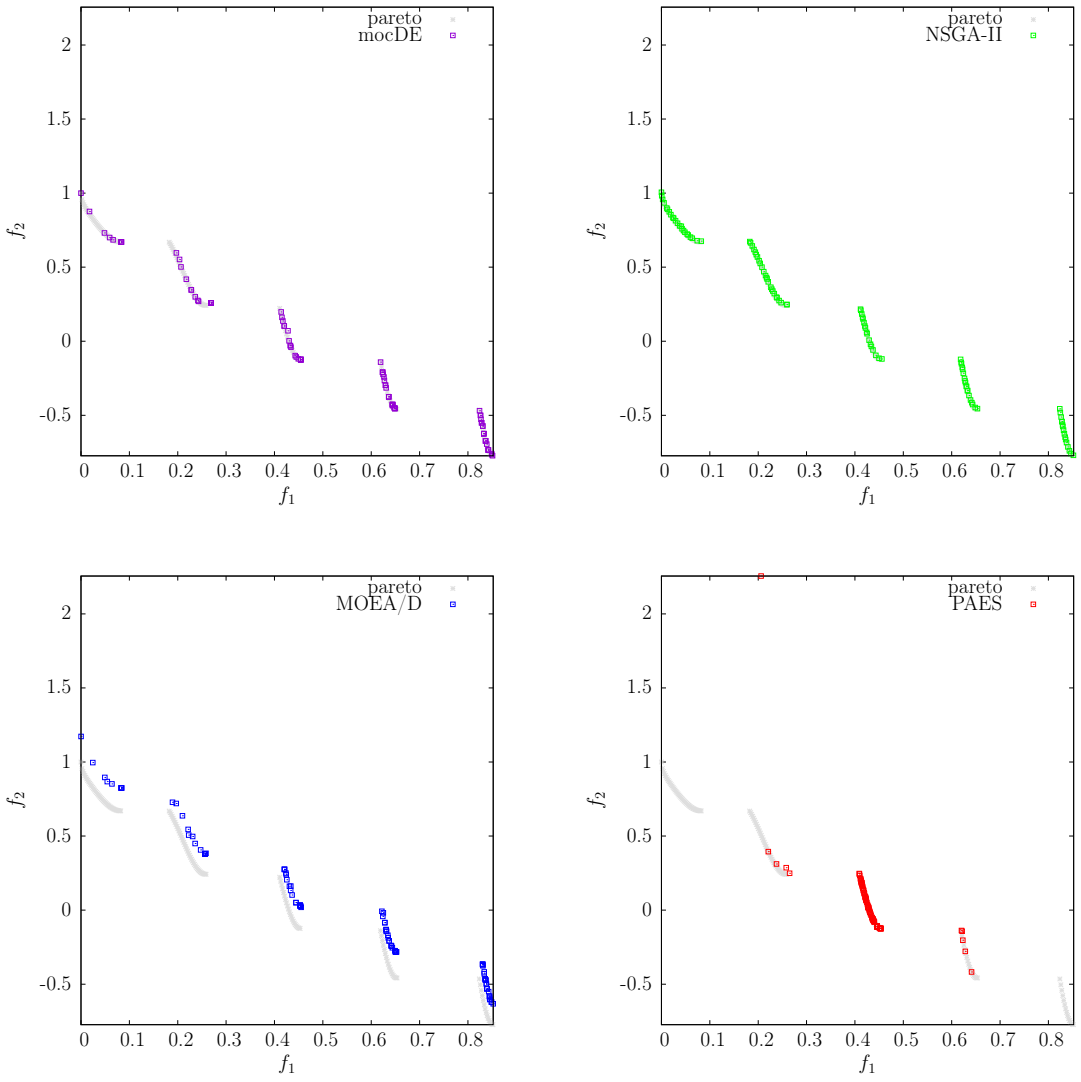


Figura 5.10: Ejecución de todos los algoritmos al resolver el problema ZDT3, donde mocDE muestra un resultado inferior que NSGA-II pero mejor que los demás algoritmos.

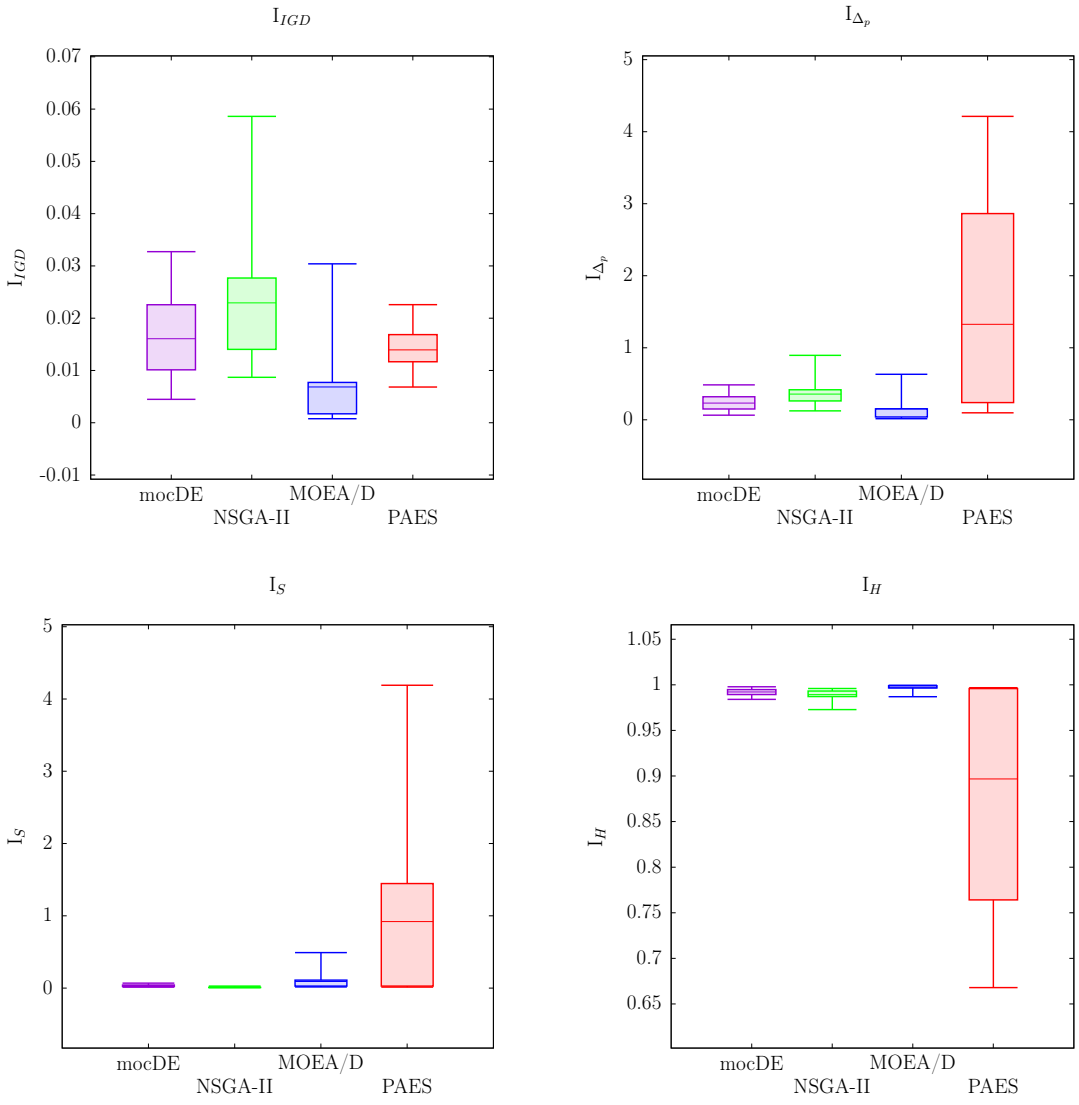


Figura 5.11: Indicadores en el problema ZDT4, donde mocDE muestra ser inferior que MOEA/D pero mejor que NSGA-II y muy competitivo con PAES (el cual muestra más inestabilidad).

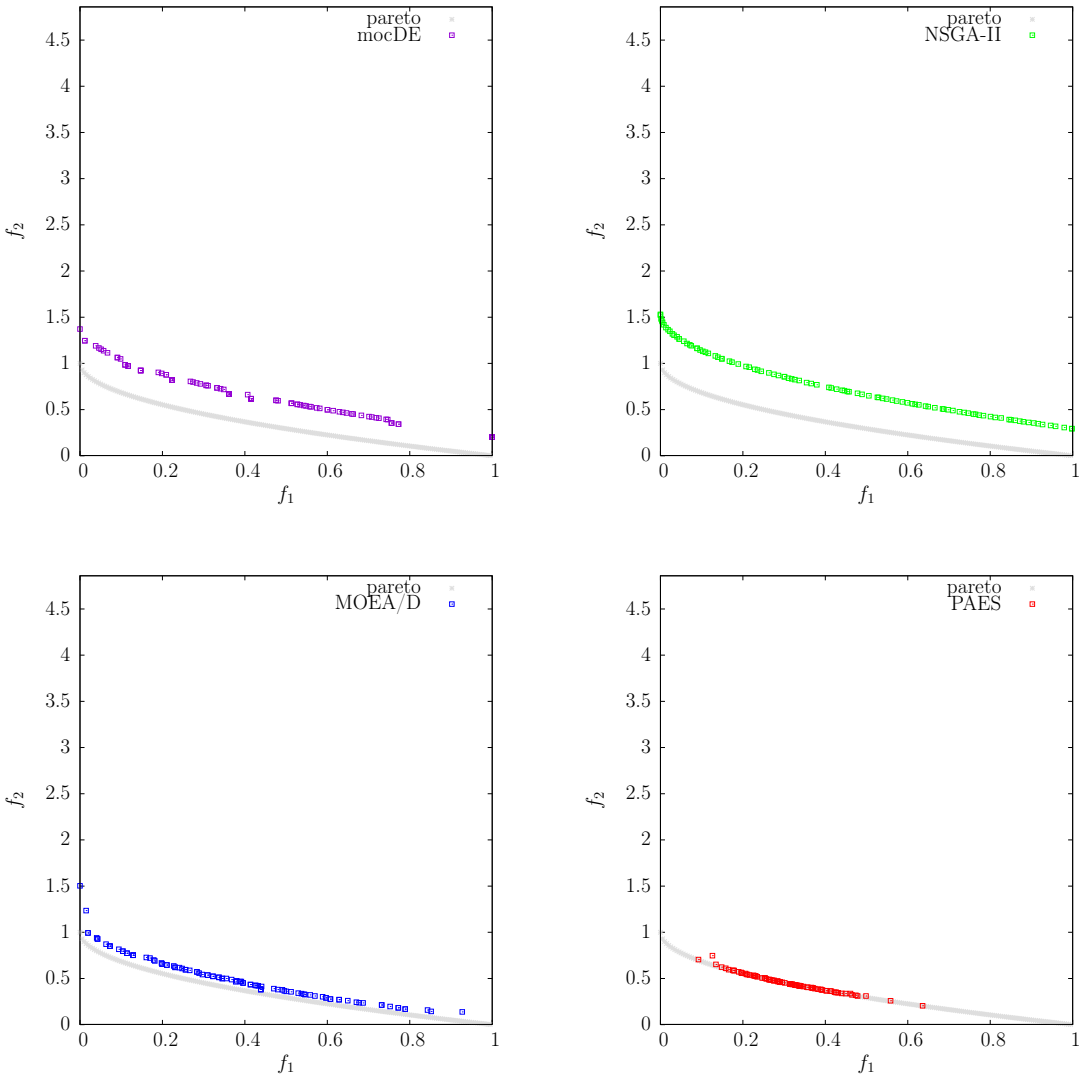


Figura 5.12: Ejecución de todos los algoritmos al resolver el problema ZDT4, donde mocDE muestra una convergencia inferior que MOEA/D y PAES pero mejor distribución que este último.

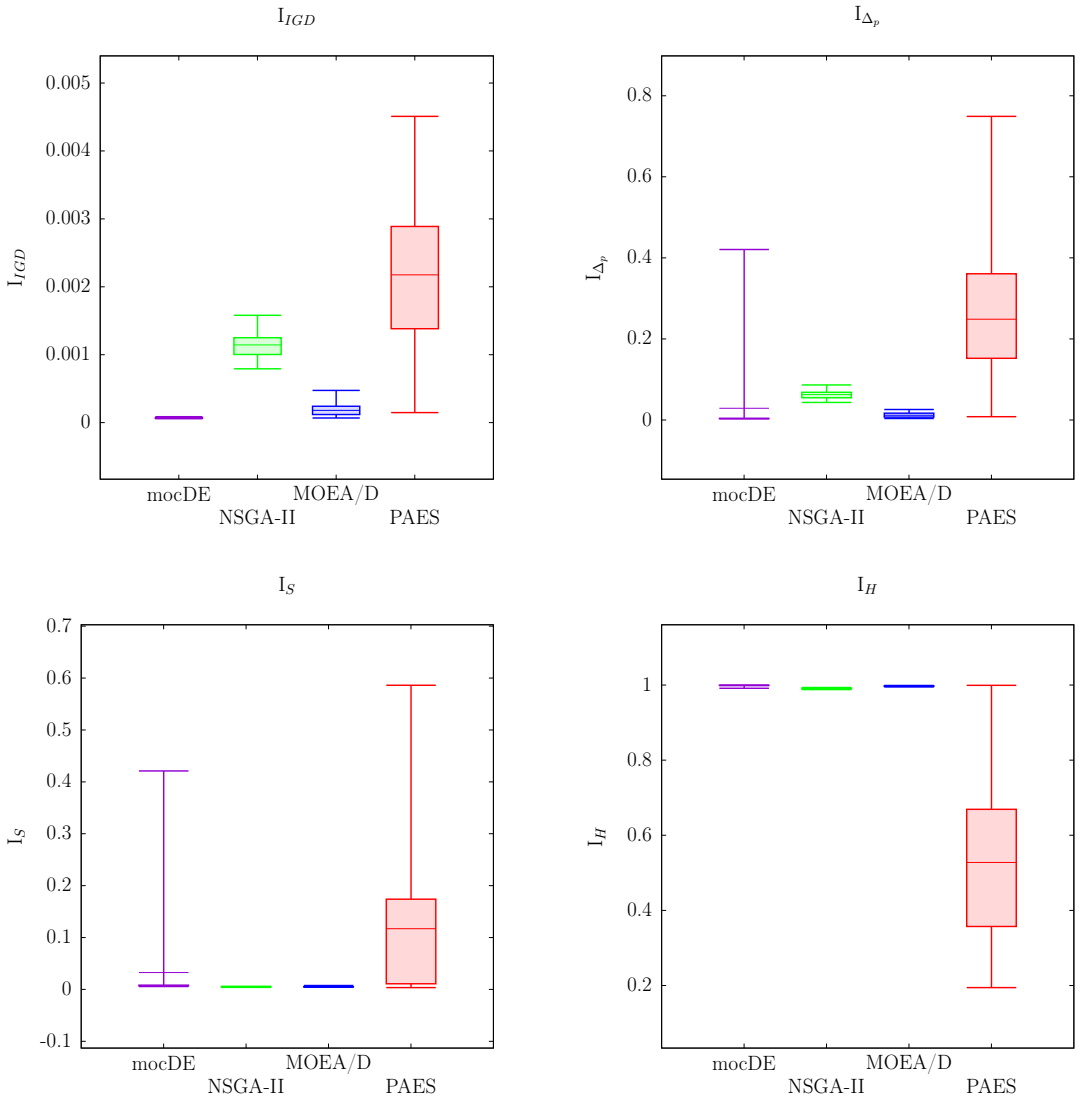


Figura 5.13: Indicadores en el problema ZDT6, donde mocDE muestra tener mejores resultados que los demás algoritmos pero de manera inestable.

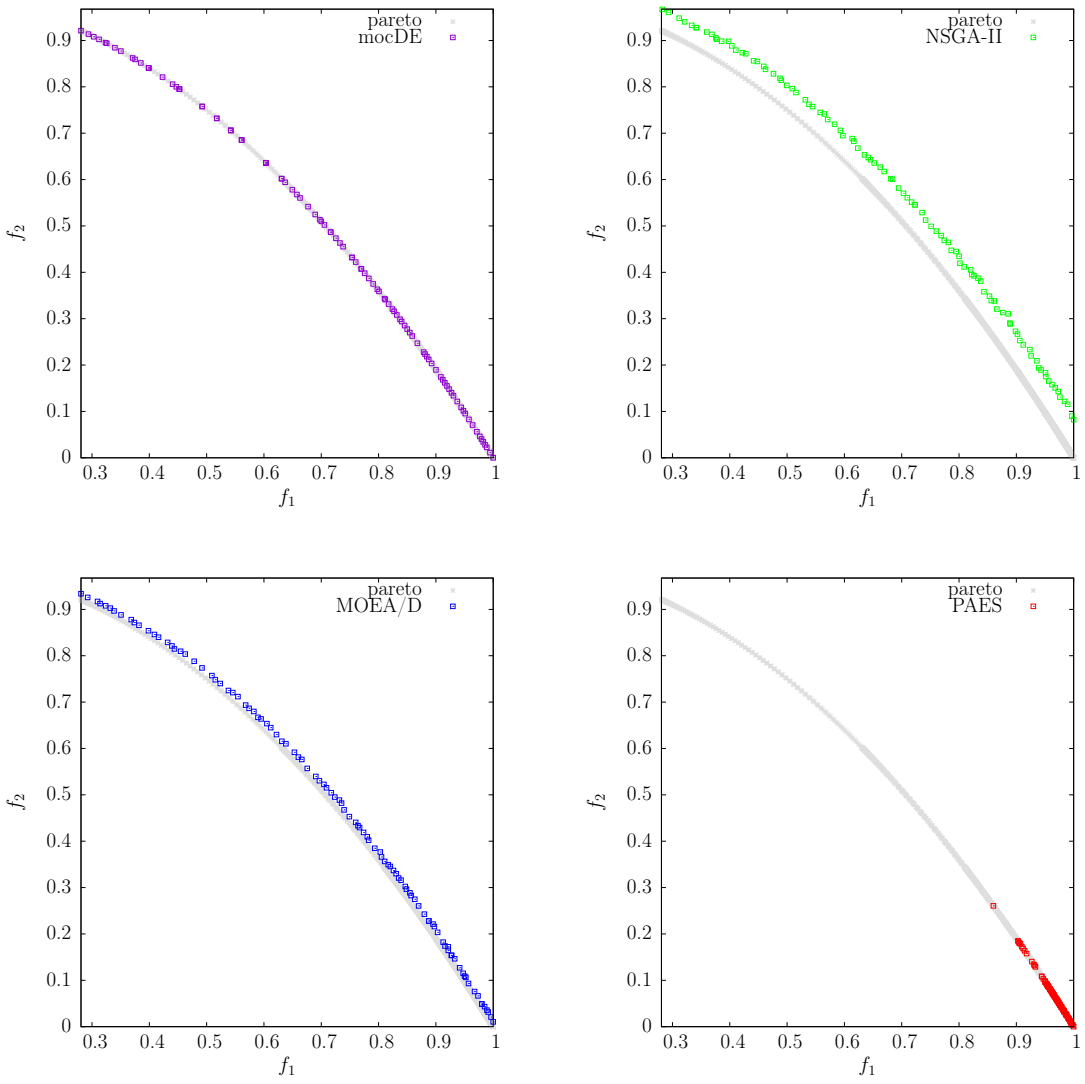


Figura 5.14: Ejecución de todos los algoritmos al resolver el problema ZDT6, donde mocDE muestra un mejor resultado que los demás algoritmos.

5.2.2. Problemas de prueba DTLZ

En los problemas DTLZ1 y DTLZ3, mocDE y PAES obtuvieron los mejores resultados (véase figuras 5.15 y 5.19). Sin embargo, PAES muestra resultados gráficos ligeramente mejores que mocDE (véase figuras 5.16 y 5.20).

En DTLZ2 tanto mocDE, como MOEA/D y NSGA-II muestran resultados muy competitivos de acuerdo a los indicadores unarios (véase figuras 5.17 y 5.18), pero mocDE los vence a todos en los indicadores binarios.

En los problemas DTLZ4 y DTLZ7, NSGA-II es el vencedor según los indicadores unarios (véase figuras 5.21, 5.22, 5.27 y 5.28), pero los indicadores binarios declaran que mocDE es mejor.

En DTLZ5, NSGA-II vuelve a resultar vencedor, pero seguido por mocDE, que demuestra ser mejor que MOEA/D (véase figuras 5.23 y 5.24).

En el problema DTLZ6, mocDE obtiene los mejores resultados de acuerdo a todos los indicadores (véase figuras 5.25 y 5.26).

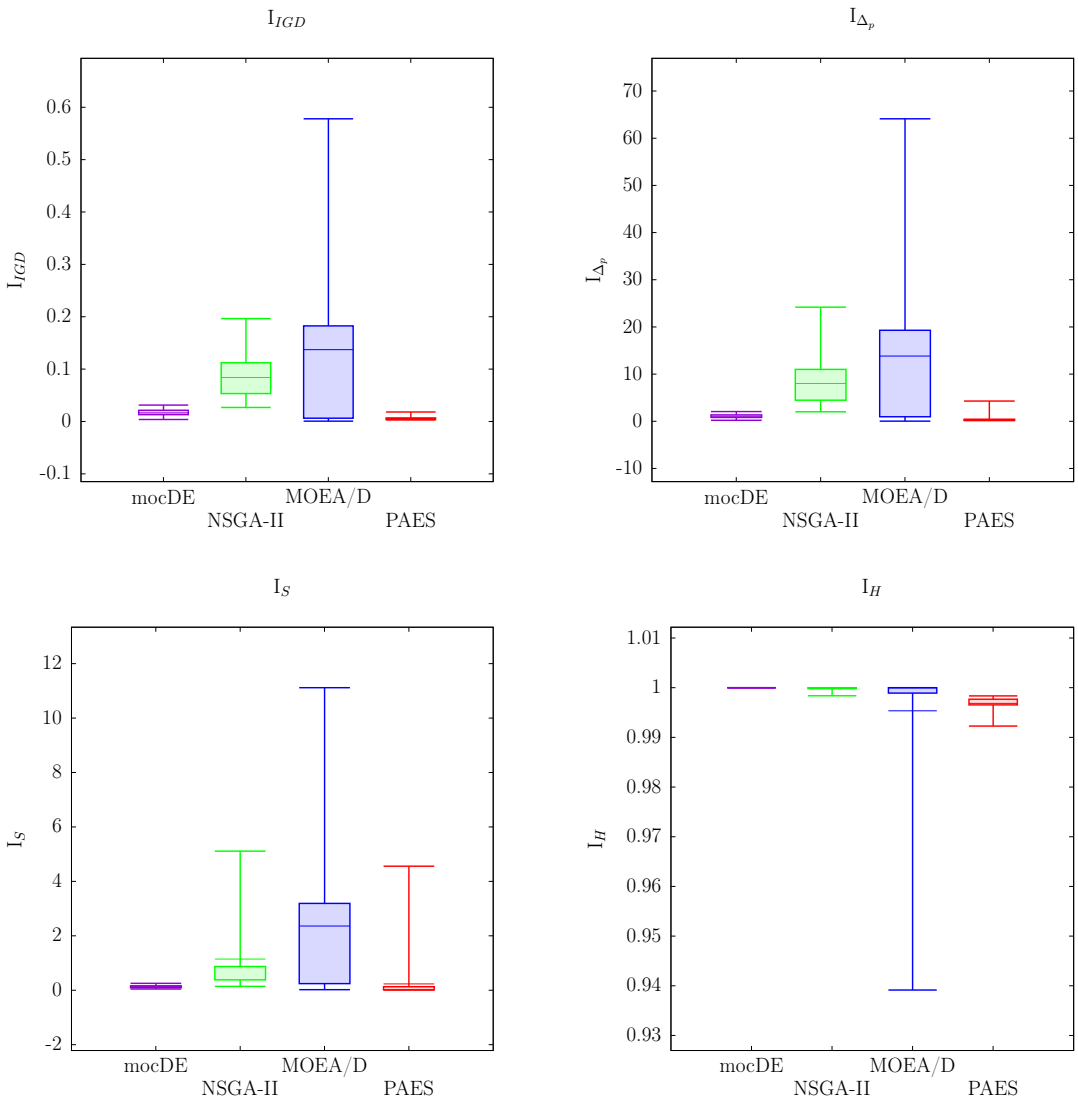


Figura 5.15: Indicadores en el problema DTLZ1, donde mocDE y PAES muestran los mejores resultados.

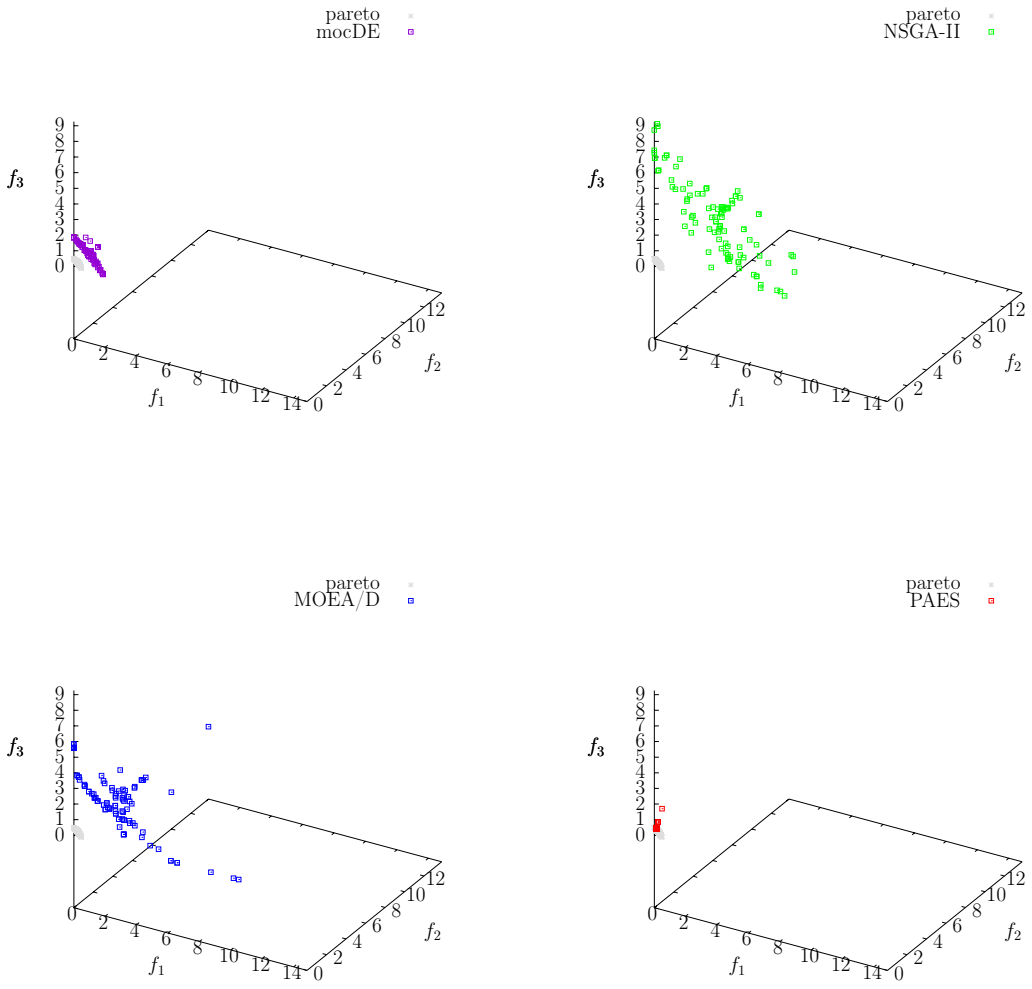


Figura 5.16: Ejecución de todos los algoritmos al resolver el problema DTLZ1, donde mocDE muestra un resultado inferior que PAES.

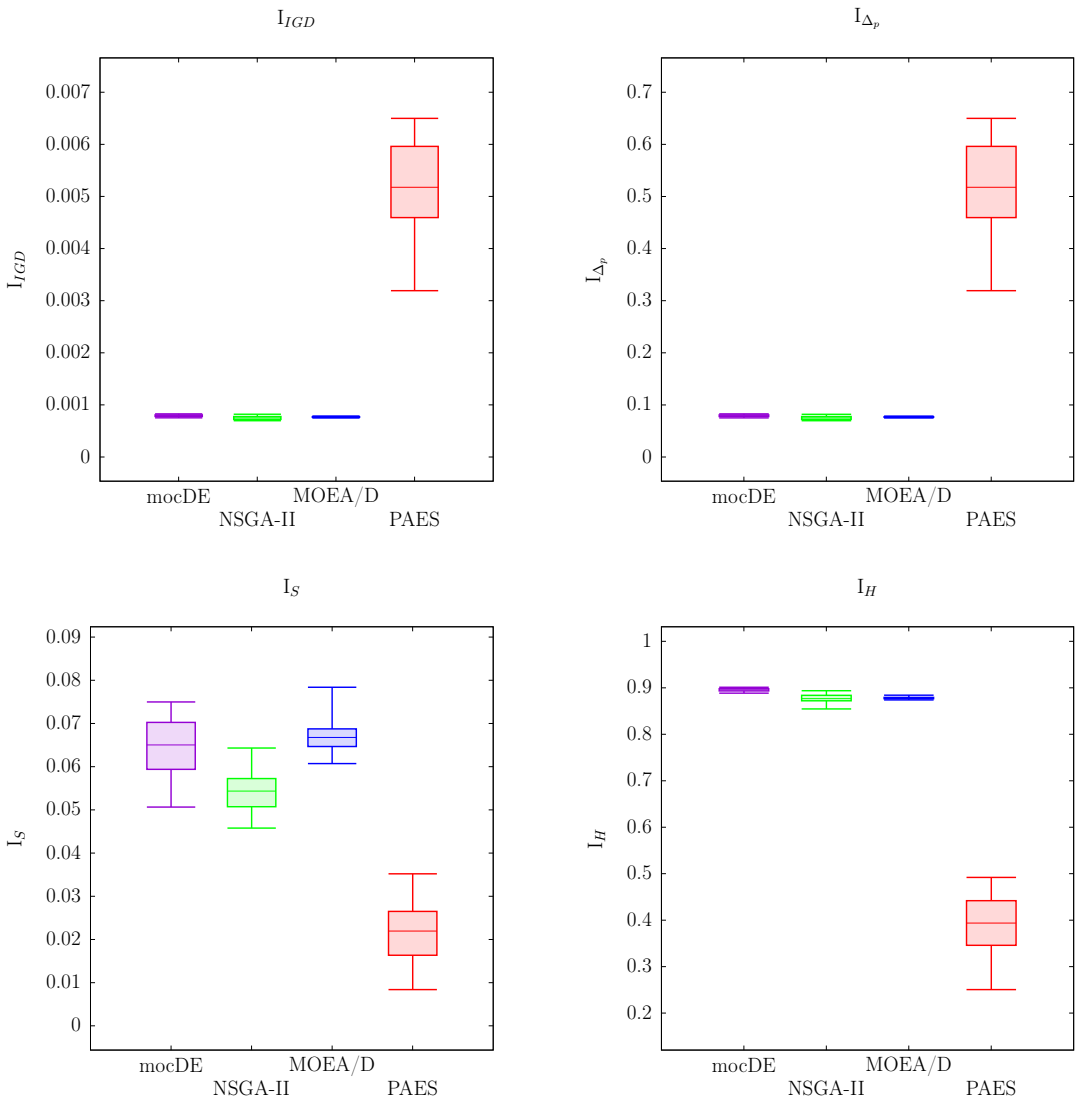


Figura 5.17: Indicadores en el problema DTLZ2, donde mocDE muestra resultados muy competitivos con NSGA-II y MOEA/D.

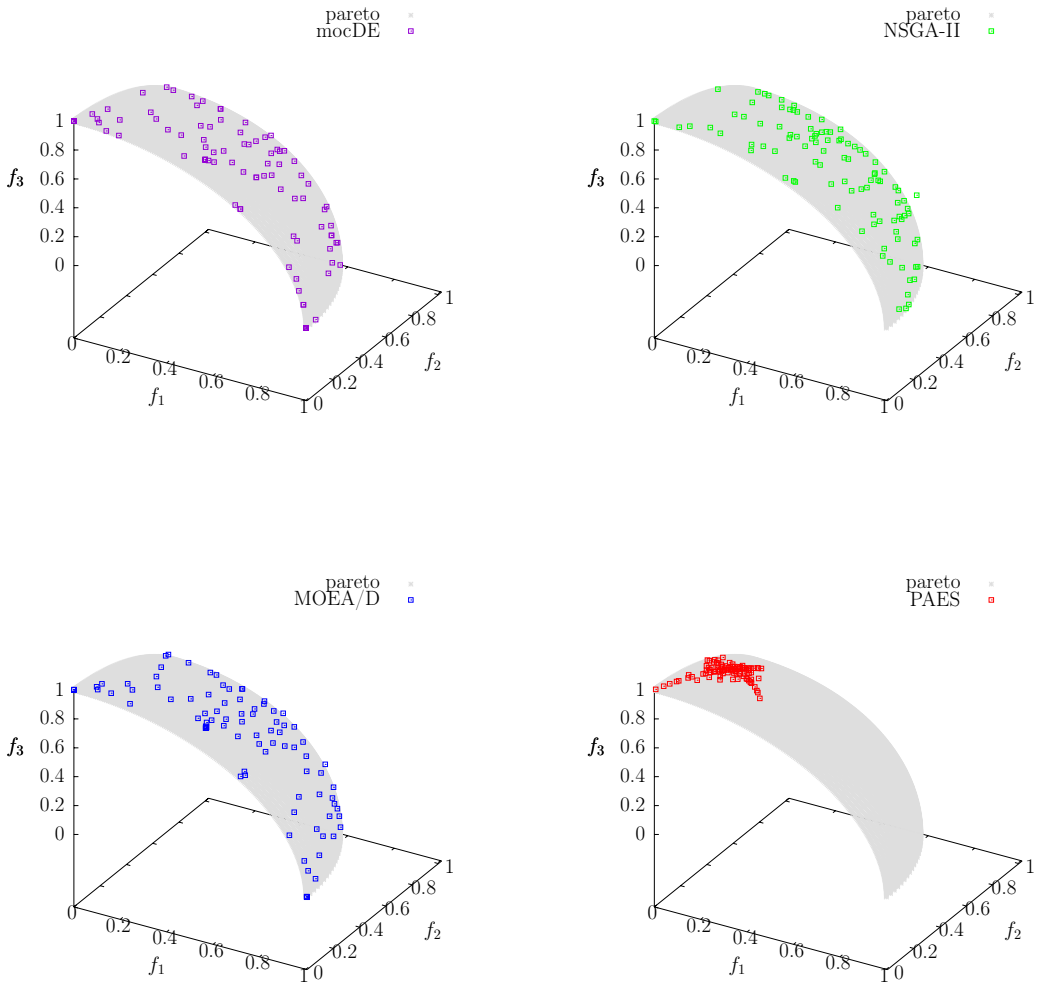


Figura 5.18: Ejecución de todos los algoritmos al resolver el problema DTLZ2, donde mocDE muestra un resultado muy competitivo con NSGA-II y MOEA/D.

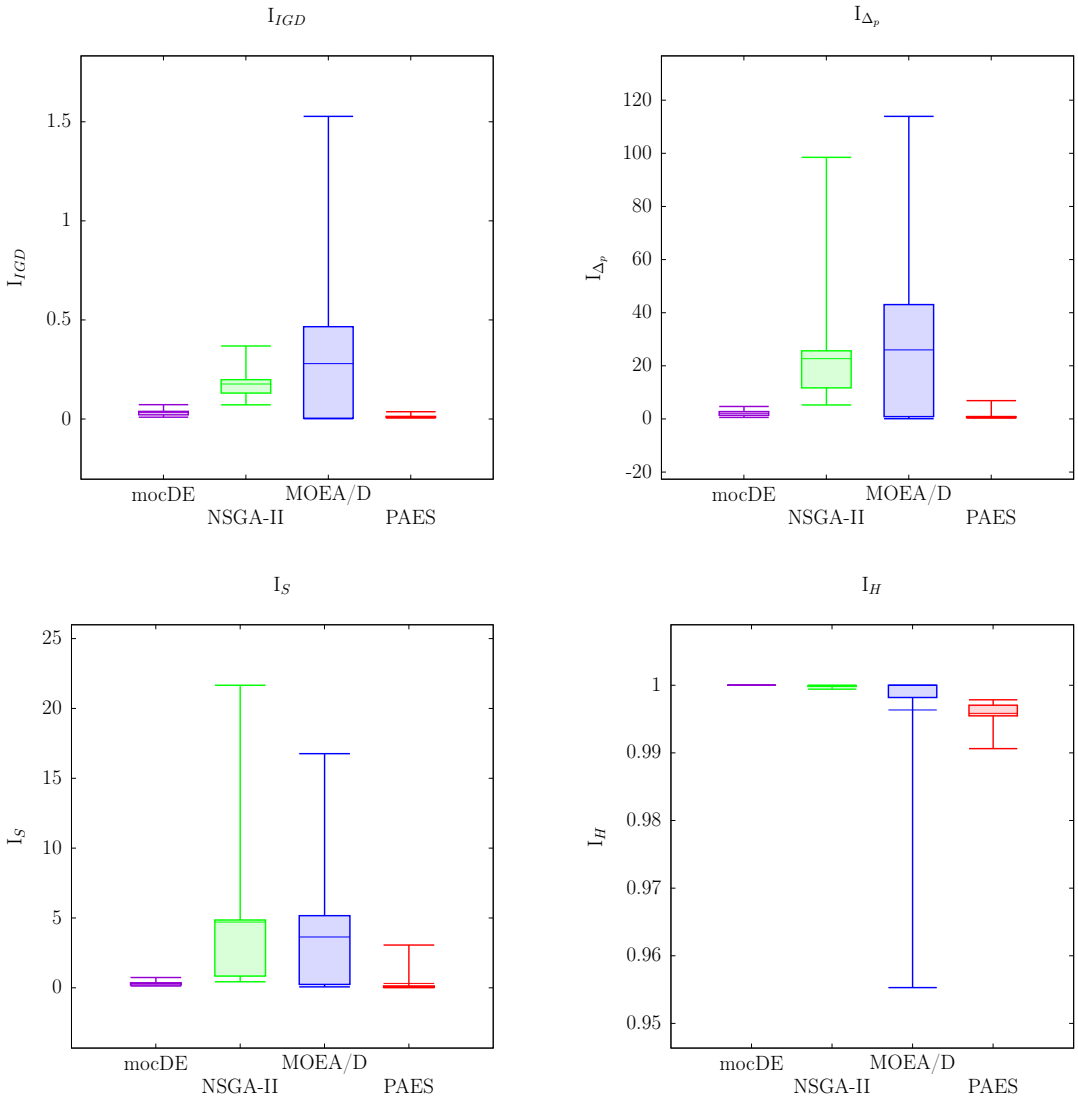


Figura 5.19: Indicadores en el problema DTLZ3, donde mocDE y PAES muestran los mejores resultados, siendo este último un poco inestable.

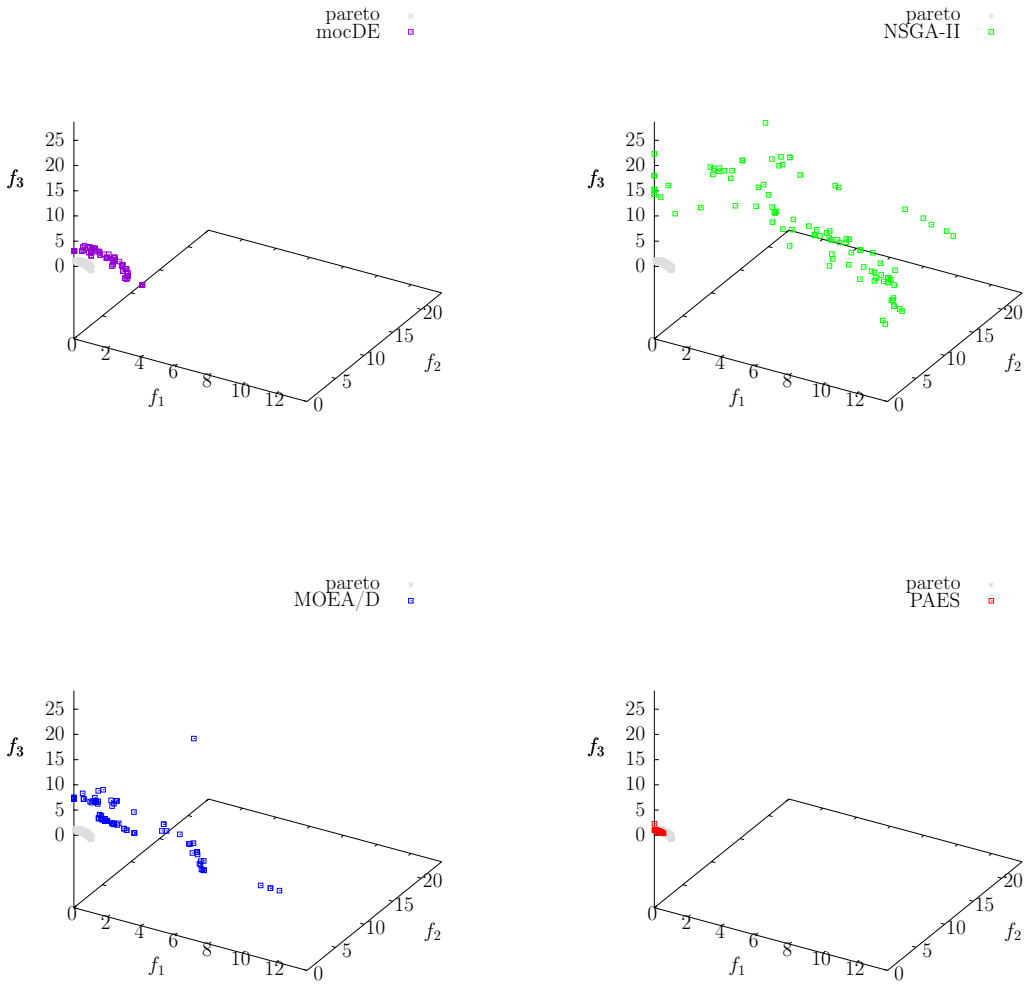


Figura 5.20: Ejecución de todos los algoritmos al resolver el problema DTLZ3, donde mocDE muestra un resultado inferior que PAES.

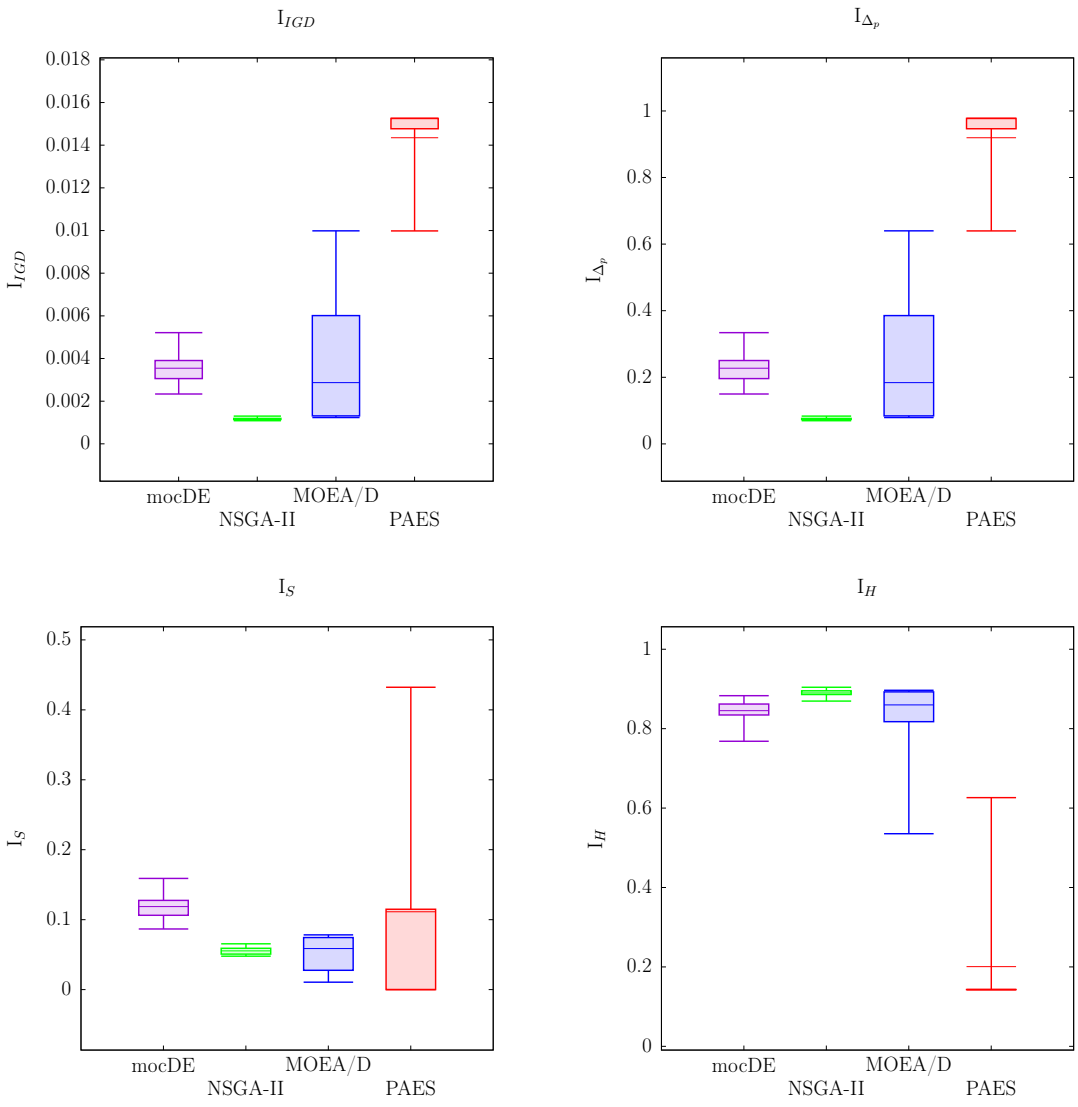


Figura 5.21: Indicadores en el problema DTLZ4, donde mocDE muestra resultados inferiores que NSGA-II y MOEA/D pero superiores que PAES.

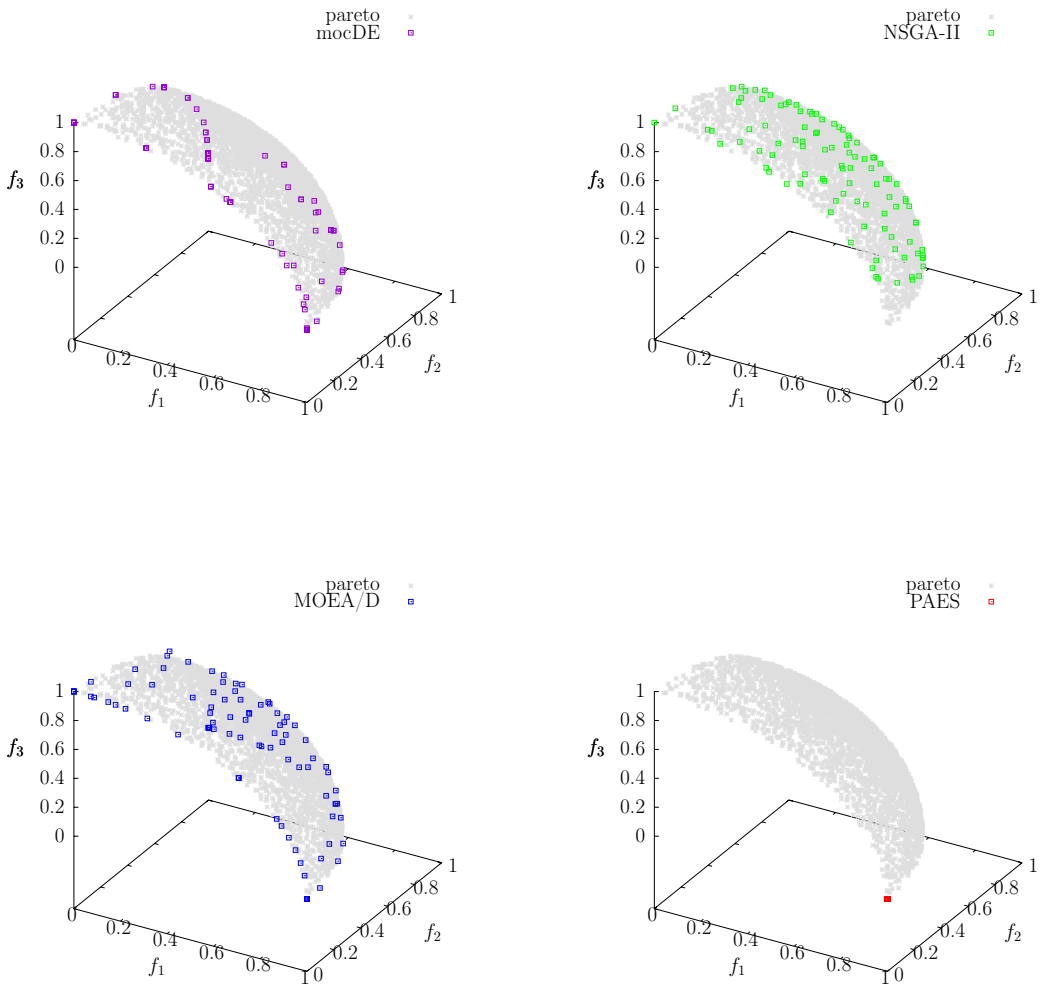


Figura 5.22: Ejecución de todos los algoritmos al resolver el problema DTLZ4, donde mocDE muestra un resultado inferior que NSGA-II y MOEA/D pero mejor que PAES, el cual se degradó y obtuvo sólo un vector solución.

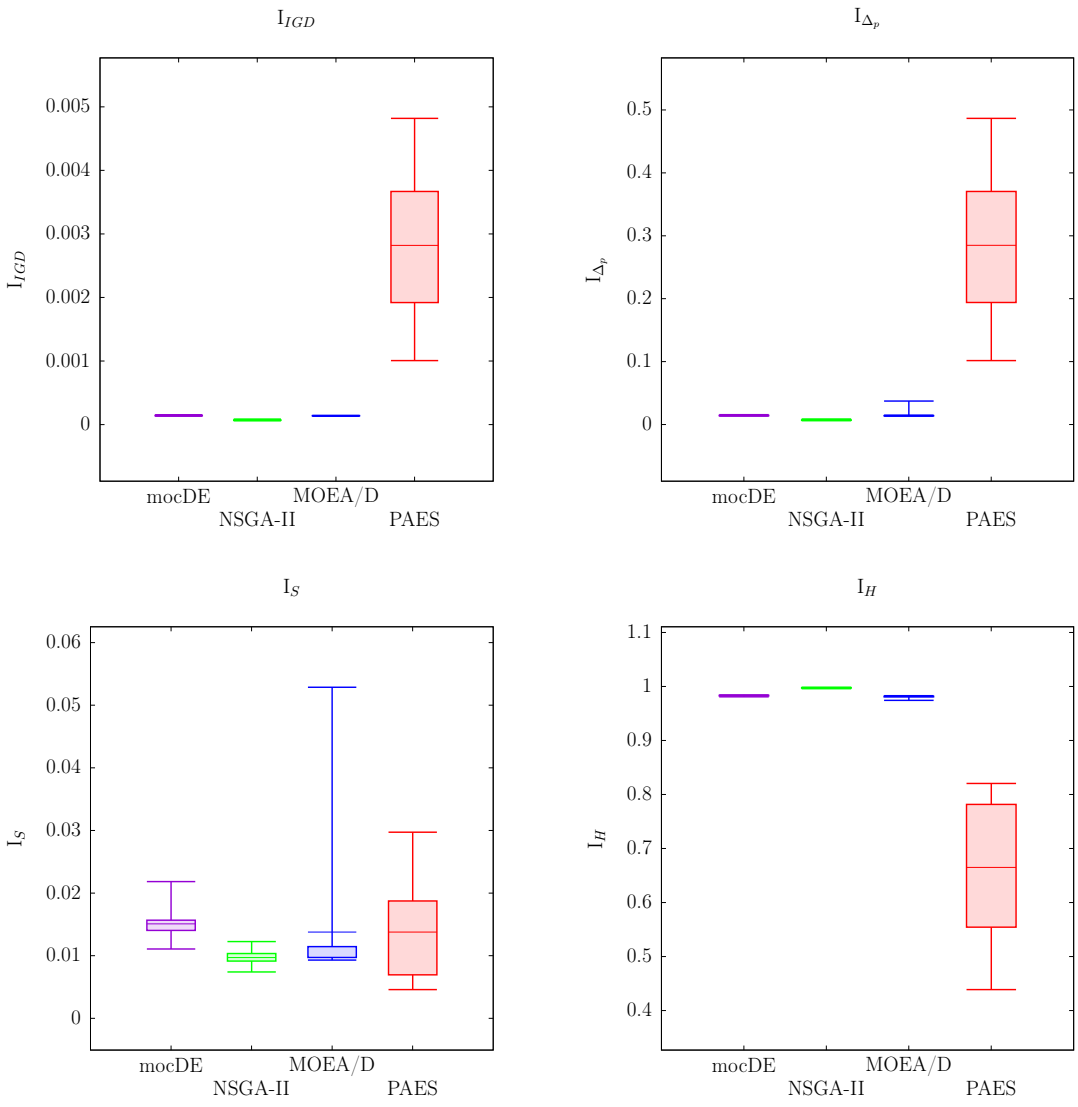


Figura 5.23: Indicadores en el problema DTLZ5, donde mocDE muestra resultados inferiores que NSGA-II pero mejores que PAES y muy competitivos con MOEA/D.

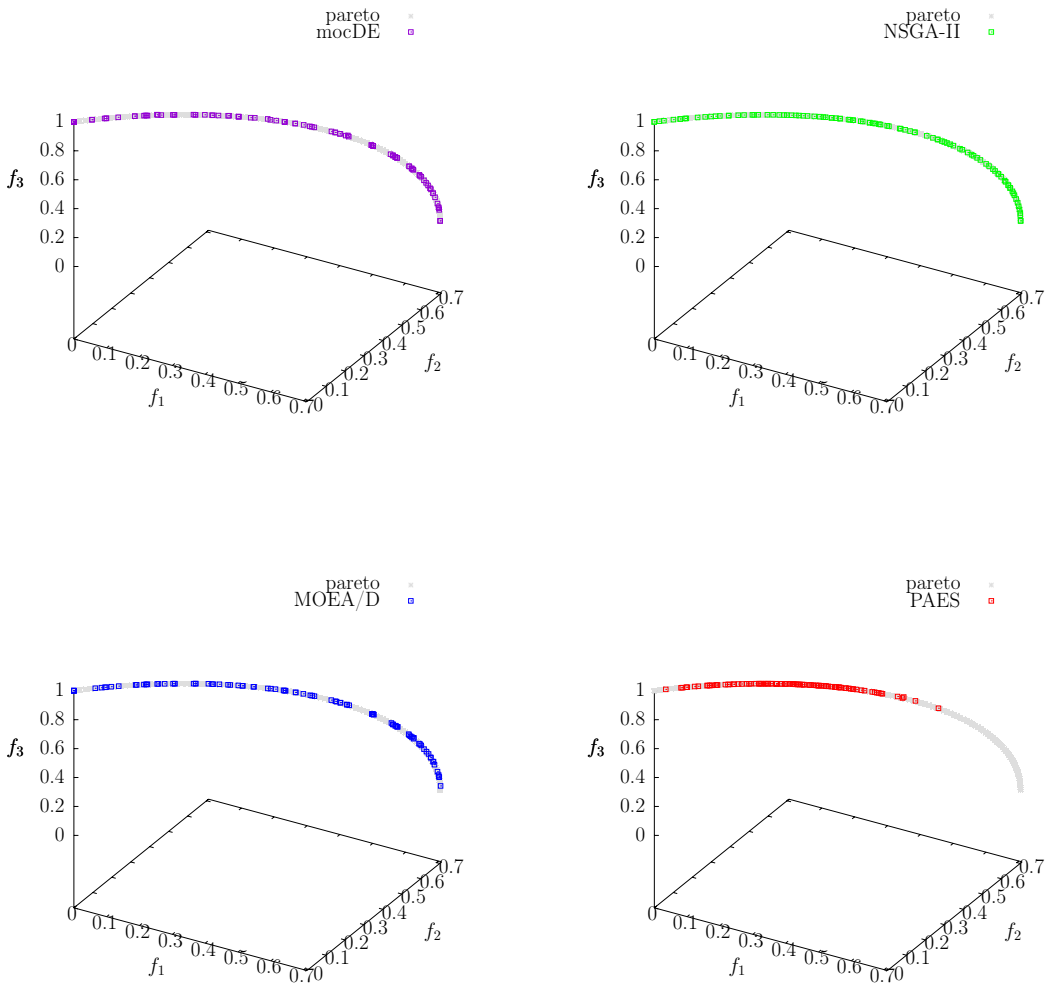


Figura 5.24: Ejecución de todos los algoritmos al resolver el problema DTLZ5, donde mocDE muestra un resultado inferior que NSGA-II pero mejor que PAES y competitivo con MOEA/D.

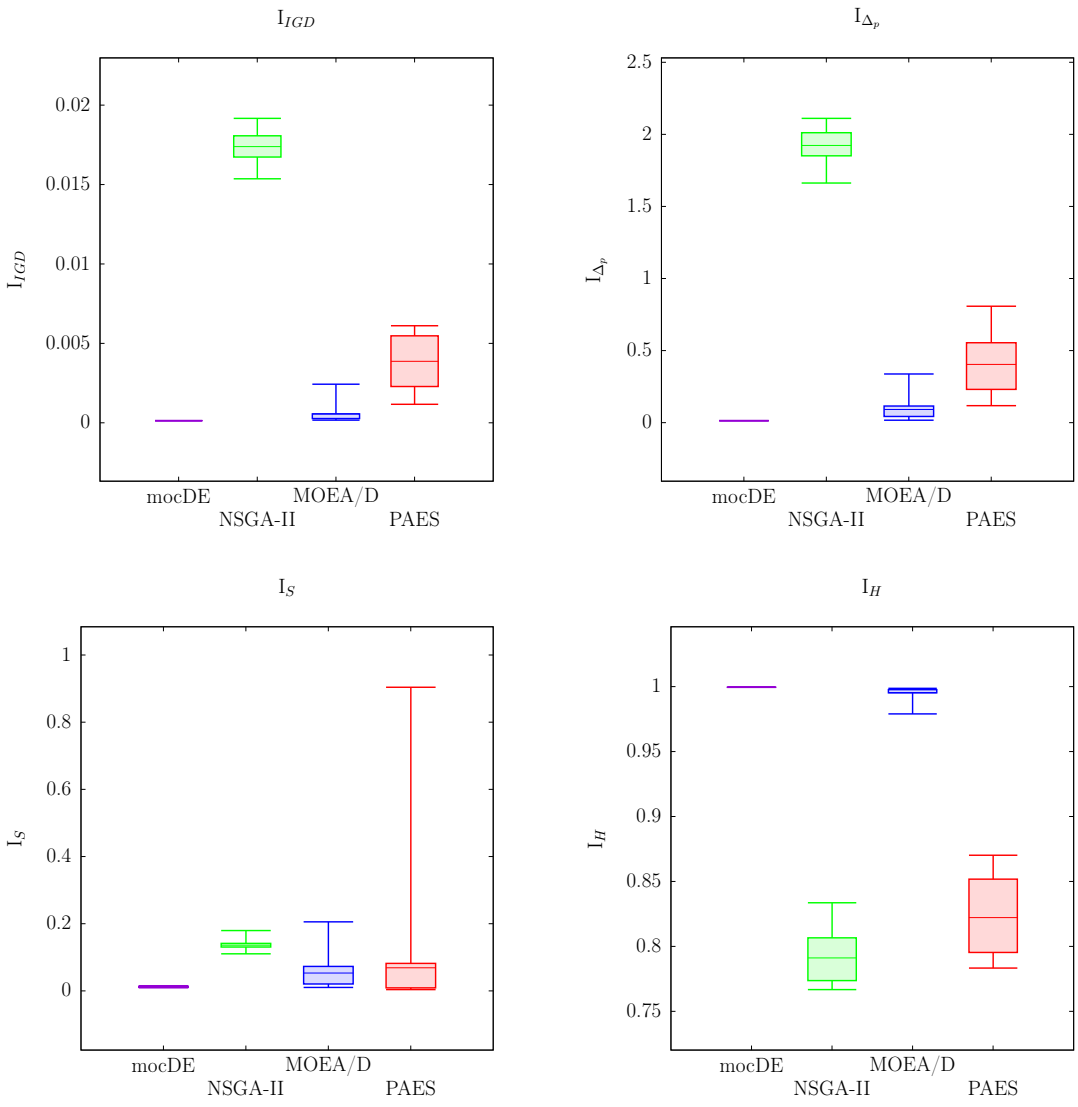


Figura 5.25: Indicadores en el problema DTLZ6, donde mocDE muestra mejores resultados que el resto de los algoritmos.

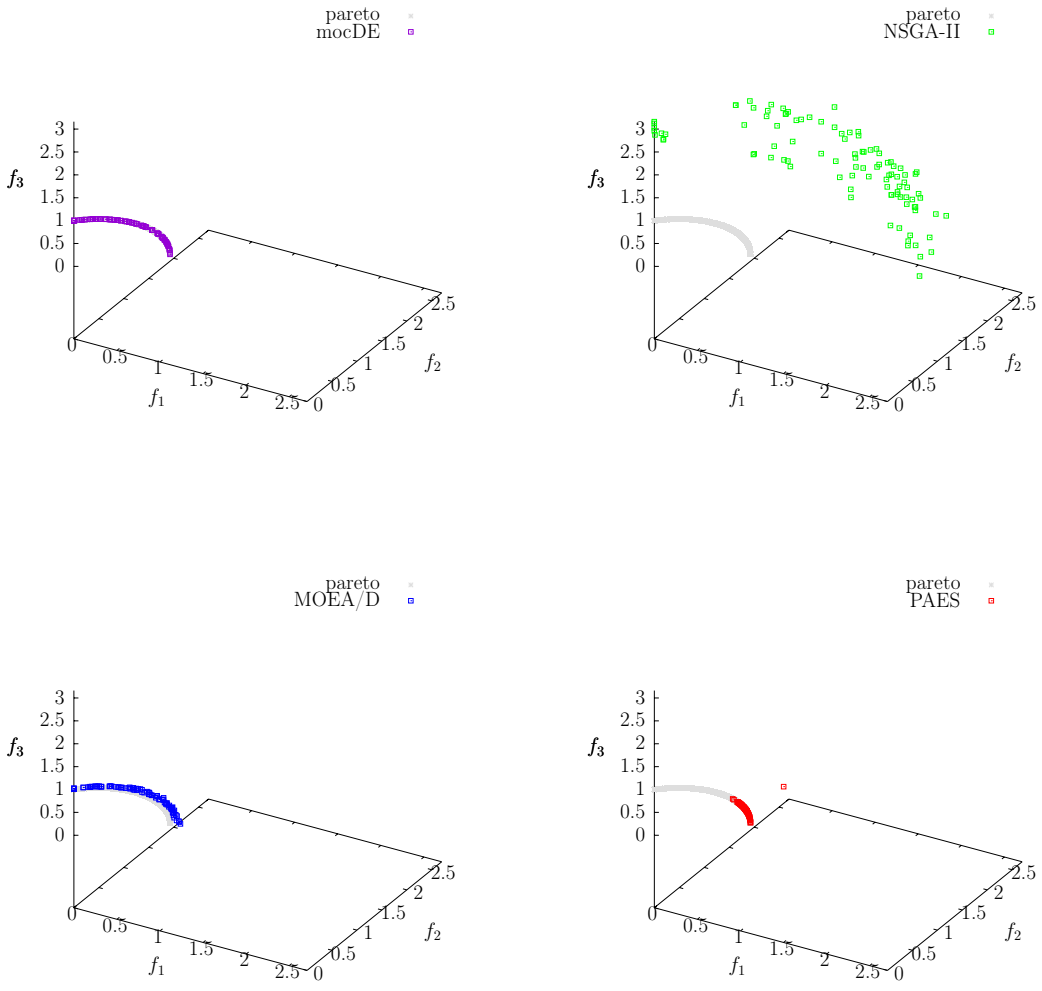


Figura 5.26: Ejecución de todos los algoritmos al resolver el problema DTLZ6, donde mocDE muestra un mejor resultado que los demás algoritmos.

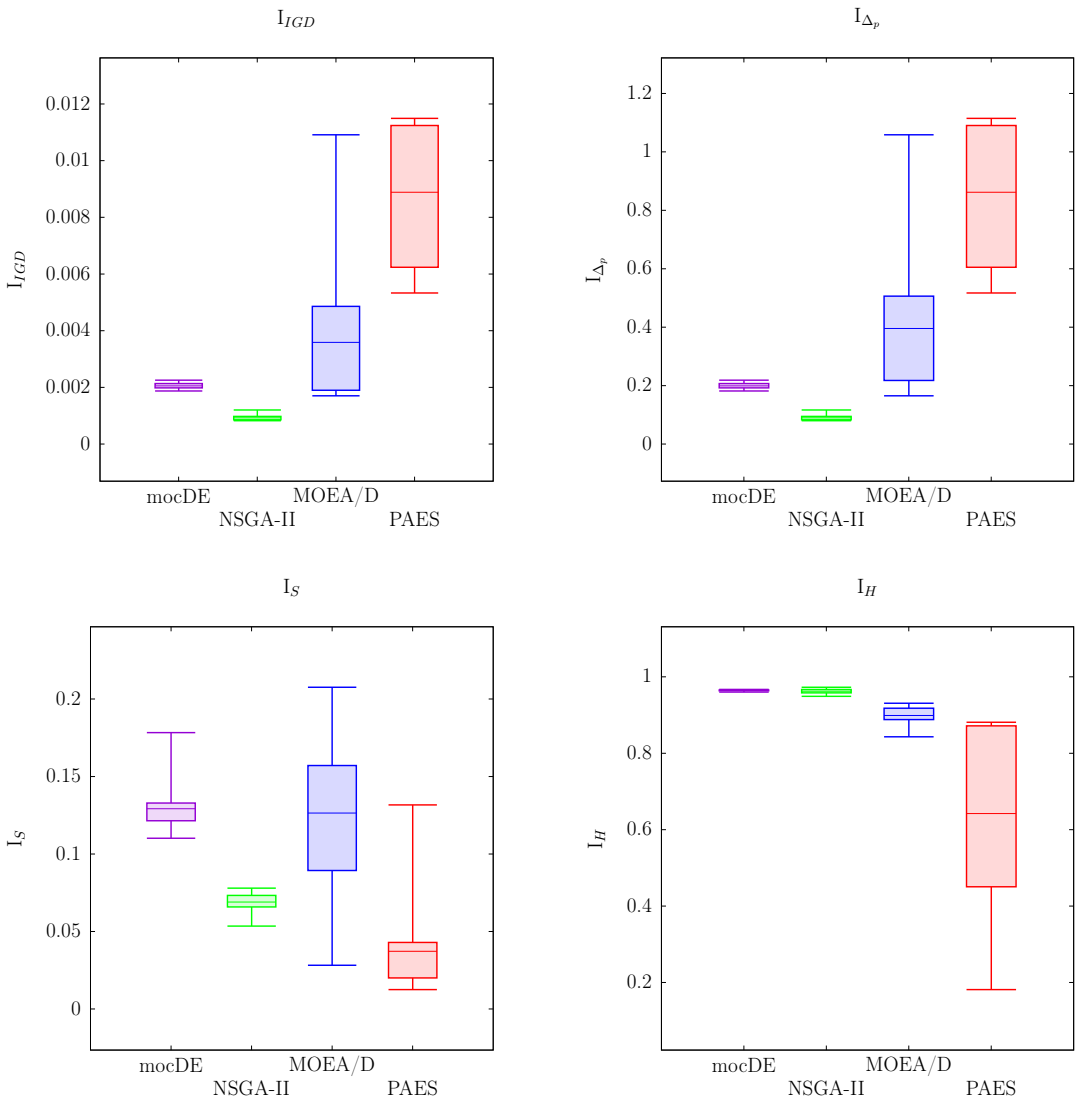


Figura 5.27: Indicadores en el problema DTLZ7, donde mocDE muestra resultados inferiores que NSGA-II pero mejores que los demás algoritmos.

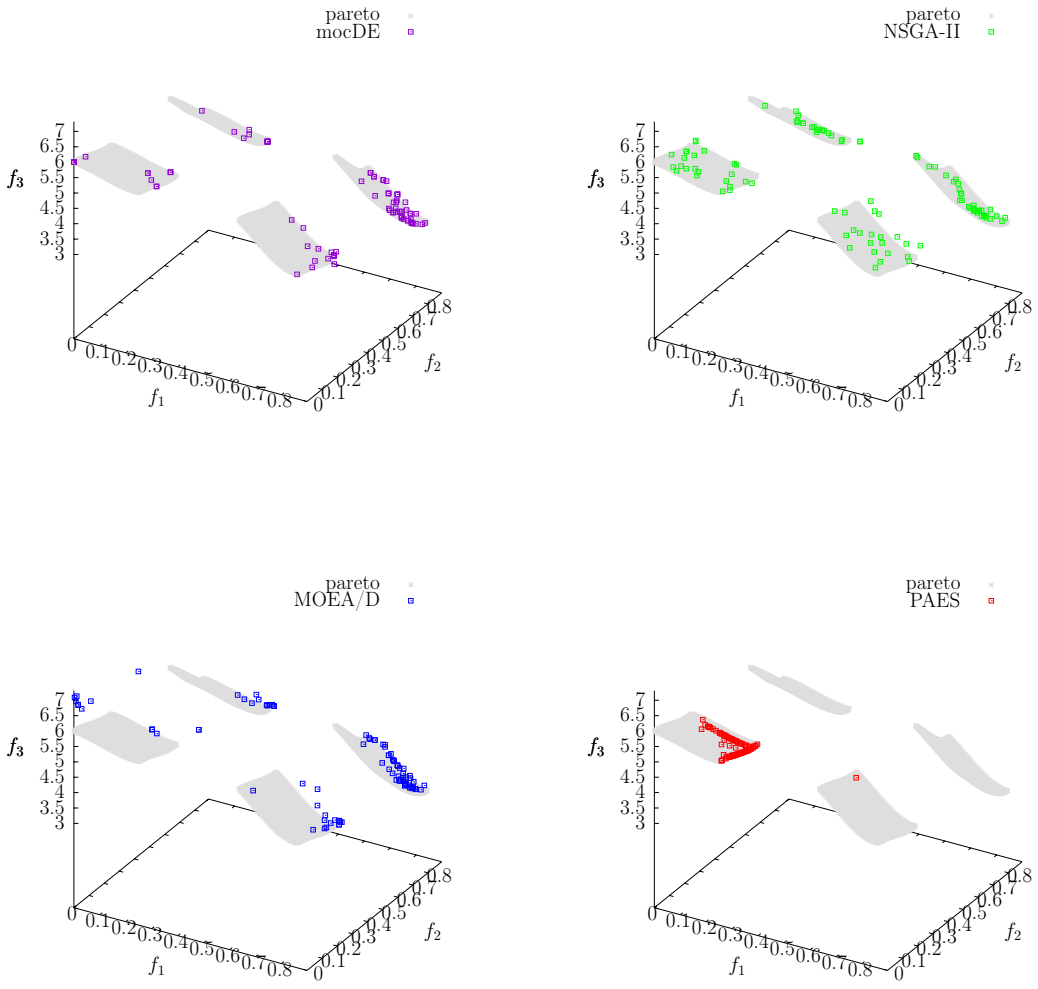


Figura 5.28: Ejecución de todos los algoritmos al resolver el problema DTLZ7, donde mocDE muestra un resultado inferior que NSGA-II pero superiores quel resto de los algoritmos.

Capítulo 6

Conclusiones

El objetivo principal de esta tesis fue diseñar un algoritmo multi-objetivo compacto que permita realizar la optimización de problemas obteniendo resultados altamente competitivos contra algoritmos del estado del arte, utilizando pocos recursos de memoria.

El diseño final presentado es mocDE, un algoritmo basado en los principios de *Compact Differential Evolution (cDE)* [25], el cual ya posee algunos atributos deseados pero sólo optimiza problemas de un objetivo. Nuestro algoritmo demuestra ser capaz de vencer o al menos obtener resultados muy competitivos contra los algoritmos del estado del arte, utilizando problemas de prueba usados en la literatura relacionada.

De forma similar a una estrategia evolutiva como PAES, mocDE utiliza sólo una solución nueva en cada iteración, lo que le permite aplicar los conceptos de cDE como si tratara de resolver un problema mono-objetivo. Sin embargo, mocDE cuenta con un archivo de soluciones no dominadas, cuya función es mantener el mejor conjunto de soluciones encontradas, y un criterio de reemplazo de dichas soluciones que coadyuva en la aproximación del verdadero frente de Pareto.

Después de validar experimentalmente mocDE y sus variantes en el capítulo 5, concluimos lo siguiente:

- Al utilizar una representación estadística de la población y no la población como tal, mocDE obtiene un ahorro inmediato en recursos de memoria, ya que sólo necesita mantener 5 vectores de longitud n, en lugar de los p vectores que mantienen los algoritmos poblacionales.

- PAES también necesita de poca memoria para realizar la optimización (dos vectores de longitud n), pero sus resultados son fácilmente superados por los de mocDE, el cual demuestra que es posible obtener buenos resultados sin necesidad del uso de memoria de un algoritmo poblacional.

- De todas las variantes de mocDE comparadas (pe-mocDE, ne-mocDE$_{50}$, ne-mocDE$_{20}$ y ne-mocDE$_{10}$), pe-mocDE fue superior en todos los indicadores de calidad en 10 de los 12 problemas probados. En los otros dos problemas, pe-mocDE no resultó ser superior, pero sí mostró ser altamente competitivo.

- Ya que el objetivo de la variante ne-mocDE es disminuir la presión elitista para evitar quedar atrapada en óptimos locales, la definitiva superioridad de pe-mocDE indica que, en problemas multi-objetivo, es vital apoyarse fuertemente en los individuos (élite) que prometen buenas regiones de búsqueda y que los óptimos locales son evitados debido a la inherente aleatoreidad obtenida por el proceso de muestreo de individuos.
- Contra el estado del arte, mocDE se mostró vencedor o altamente competitivo en 10 problemas, mientras que en los otros dos problemas quedó muy cerca del mejor algoritmo.
- A excepción de DTLZ4, mocDE obtuvo gran estabilidad en todos los problemas, demostrando su confiabilidad y robustez.

6.1. Trabajo futuro

Al compararse con los algoritmos del estado del arte, mocDE mostró tener buena distribución. Sin embargo, NSGA-II fue superior en este aspecto en casi todos los problemas de prueba. Por lo tanto, se requiere la investigación de otro método de archivado que permita mejorar la distribución, sin afectar la convergencia ya alcanzada.

Finalmente, es necesario investigar a fondo el uso del procesador por parte de mocDE y determinar cómo puede utilizarse en aplicaciones que funcionan en tiempo real.

Apéndice A

Indicadores de calidad para optimización multi-objetivo

Dado que los algoritmos evolutivos no pueden garantizar la optimalidad de sus soluciones, se han diseñado varios indicadores que permitan conocer su calidad. En este apéndice, se presentan los indicadores de calidad que serán utilizados para comparar los resultados obtenidos por los algoritmos.

A.1. Indicadores de calidad unarios

Definición 9 (Indicador de calidad unario)**.** Un *indicador de calidad unario* es una función $I_1 : \Omega \rightarrow \mathbb{R}$, donde Ω es el conjunto de todas las aproximaciones posibles al frente de Pareto para un problema dado.

El indicador I_1 debe imponer un orden total entre las aproximaciones al frente de Pareto, a fin de poder comparar la calidad de los resultados de dos algoritmos de optimización multi-objetivo simplemente al comparar los valores obtenidos por el indicador.

A continuación, se definen los cuatro indicadores unarios que serán utilizados para realizar las comparaciones entre los resultados de los algoritmos considerados en esta tesis.

Definición 10 (Indicador de hipervolumen (I_H))**.** El *indicador de hipervolumen* [28] se define como el tamaño del espacio cubierto por un conjunto de soluciones no dominadas. I_H calcula la medida *Lebesgue* de la unión de k-hiper-rectángulos definidos por cada $a_i \in A$ y un punto de referencia r, donde $A = (a_1, ..., a_l)$ es un conjunto de vectores solución y $k = |a_i|$.

Las propiedades de I_H son:

- Es el único indicador unario conocido que garantiza estricta monotonicidad de acuerdo a la dominancia de Pareto.

- Puede ser maximizado si se utiliza un punto de referencia mayor o igual al vector nadir, o minimizado si el punto de referencia es menor o igual al vector ideal.
- No es invariante a la escala, ya que es susceptible al punto de referencia.
- Su tiempo de ejecución es exponencial en el número de objetivos del problema.

Definición 11 (Indicador de la distancia generacional invertida ($\mathrm{I}_{\mathrm{IGD}}$))**.** El *indicador de la distancia generacional invertida* [29] indica qué tan lejos se encuentra un conjunto de soluciones A del frente de Pareto $\mathcal{P}_F$. Se define como:

$$I_{\mathrm{IGD}} = \frac{\left(\sum_{i=1}^{|\mathcal{P}_F|} ||\mathcal{P}_{F_i} - a_{p_i}||_c\right)^{\frac{1}{c}}}{|\mathcal{P}_F|} \tag{A.1}$$

donde $a_{p_i} \in A$ es el vector más cercano a $\mathcal{P}_{F_i}$ y c es una constante positiva previamente definida.

Propiedades de $\mathrm{I}_{\mathrm{IGD}}$:

- Un menor valor representa un mejor conjunto de soluciones.
- Es dependiente de lo fina que sea la discretización del frente de Pareto.

Definición 12 (Indicador Δ_p (I_{Δ_p}))**.** El *indicador* Δ_p [30] representa la distancia de *Hausdorff* promedio entre un conjunto de soluciones A y el frente de Pareto $\mathcal{P}_F$. Se define como:

$$I_{\Delta_p} = \max(I_{\mathrm{GD}_p}, I_{\mathrm{IGD}_p}) \tag{A.2}$$

$$I_{GD_p} = \left(\frac{1}{|A|}\sum_{i=1}^{|A|} ||a_i - \mathcal{P}_{F_{a_i}}||_c\right)^c \tag{A.3}$$

$$I_{\mathrm{IGD}_p} = \left(\frac{1}{|\mathcal{P}_F|}\sum_{i=1}^{|\mathcal{P}_F|} ||\mathcal{P}_{F_i} - a_{\mathcal{P}_{F_i}}||_c\right)^c \tag{A.4}$$

donde $\mathcal{P}_{F_{a_i}} \in \mathcal{P}_F$ es el vector más cercano a a_i y $a_{\mathcal{P}_{F_i}} \in A$ el vector más cercano a $\mathcal{P}_{F_i}$.

Propiedades de I_{Δ_p}:

- Es positiva y simétrica.
- Si las magnitudes de los conjuntos de soluciones tiene un límite, se cumple la inecualidad relajada del triángulo y es una pseudo-métrica.

- Si $c = \infty$ en las ecuaciones A.3 y A.4, entonces I_{Δ_p} es igual a la distancia de *Hausdorff*.

Definición 13 (Indicador de espaciado (I_S))**.** El *indicador de espaciado* [31] mide la variación de los vectores vecinos en un conjunto de soluciones A. Se define como:

$$I_S = \sqrt{\frac{1}{|A|-1}\sum_{i=1}^{|A|}(\bar{d} - d_i)^2} \tag{A.5}$$

donde $d_i = \min_{j\in 1,\ldots,|A|} \sum_{i=1}^{k} |f_i(a_i) - f_i(a_j)|$ y $\bar{d} = \dfrac{\sum_{i=1}^{|A|} d_i}{|A|}$

Propiedades de I_S:

- No es monotónico de acuerdo a la dominancia de Pareto.
- No es invariante a la escala.
- Su tiempo de ejecución es $O(|A|^2)$.

A.2. Indicadores de calidad binarios

Definición 14 (Indicador de calidad binario)**.** Un *indicador de calidad binario* es una función $\mathrm{I}_2 : \Omega \times \Omega \to \mathbb{R}$, donde Ω es el conjunto de todas las aproximaciones posibles al frente de Pareto para un problema dado.

Un indicador de calidad binario permite comparar de manera directa dos conjuntos de soluciones A y B, otorgando una percepción más clara de la dominancia entre ellos.

A continuación, se definen los tres indicadores binarios que serán utilizados para realizar las comparaciones entre los resultados de los algoritmos.

Definición 15. Indicador ϵ binario multiplicativo (I_{ϵ^*}) El *indicador ϵ binario multiplicativo* [32] representa la constante escalar mínima ϵ por la que deben multiplicarse las soluciones de un conjunto A, para que domine en su totalidad a un conjunto B. Se define como:

$$I_{\epsilon^*}(A, B) = \max_{b\in B} \min_{a\in A} \max_{1\le i\le k} \frac{f_i(a)}{f_i(b)} \tag{A.6}$$

Definición 16. Indicador ϵ binario aditivo (I_{ϵ^+}) El *indicador ϵ binario aditivo* [32] representa la constante escalar mínima ϵ que debe sumarse a las soluciones de un conjunto A, para que domine en su totalidad a un conjunto B. Se define como:

$$I_{\epsilon^+}(A, B) = \max_{b\in B} \min_{a\in A} \max_{1\le i\le k} f_i(a) - f_i(b) \tag{A.7}$$

Propiedades de I_{ϵ^*} y I_{ϵ^+}:

- Son monotónicos, pero no estrictamente monotónicos, de acuerdo a la dominancia de Pareto.
- Su tiempo de ejecución es $O(k|A||B|)$.
- Un menor valor representa un mejor conjunto de soluciones, es decir, A es mejor que B si $I_\epsilon(A,B) < I_\epsilon(B,A)$.

Definición 17 (Indicador de cobertura (I_C))**.** El *indicador de cobertura* [28] representa el porcentaje de soluciones de un conjunto B que son dominadas por al menos un elemento del conjunto A. Este indicador se define como:

$$I_C(A,B) = \frac{|b \in B | \exists a \in A : a \preceq b|}{|B|} \tag{A.8}$$

Las propiedades de I_C son:

- No es simétrico, por lo que debe considerarse tanto I_C(A,B) como I_C(B,A) para determinar el mejor conjunto.
- Es estrictamente monotónico, de acuerdo a la dominancia de Pareto.
- Su tiempo de ejecución es de $O(|A||B|)$.
- Es invariante a la escala.

Apéndice B

Problemas de prueba

B.1. Problemas Zitzler-Deb-Thiele (ZDT)

La definición los problemas ZDT se puede apreciar en la tabla B.1, a excepción de ZDT5, el cual se omitió por ser un problema binario. Los frentes de Pareto verdaderos de dichos problemas aparecen en las figuras B.1, B.3, B.2, B.4 y B.5.

B.2. Problemas Deb-Thiele-Laumanns-Zitzler (DTLZ)

La definición los problemas DTLZ se puede apreciar en las tablas B.2 y B.3. Los frentes de Pareto verdaderos de dichos problemas aparecen en las figuras B.6, B.8, B.7, B.9, B.10, B.12 y B.11.

Problema	Definición	Restricciones
ZDT1	$F = (f_1(x), f_2(x))$ $f_1(x) = x_1$ $f_2(x) = g(x)\left(1 - \sqrt{\frac{f_1}{g(x)}}\right)$ $g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	$n = 30$ $0 \leq x_i \leq 1$ $i = 1, 2, ..., 30$
ZDT2	$F = (f_1(x), f_2(x))$ $f_1(x) = x_1$ $f_2(x) = g(x)\left(1 - \left(\frac{f_1}{g(x)}\right)^2\right)$ $g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	$n = 30$ $0 \leq x_i \leq 1$ $i = 1, 2, ..., 30$
ZDT3	$F = (f_1(x), f_2(x))$ $f_1(x) = x_1$ $f_2(x) = g(x)\left(1 - \sqrt{\frac{f_1}{g(x)}} - \frac{f_1}{g(x)}\sin(10\pi f_1)\right)$ $g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	$n = 30$ $0 \leq x_i \leq 1$ $i = 1, 2, ..., 30$
ZDT4	$F = (f_1(x), f_2(x))$ $f_1(x) = x_1$ $f_2(x) = g(x)\left(1 - \sqrt{\frac{f_1}{g(x)}}\right)$ $g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i))$	$n = 10$ $0 \leq x_1 \leq 1$ $-5 \leq x_i \leq 5$ $i = 2, 3, ..., 10$
ZDT6	$F = (f_1(x), f_2(x))$ $f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(x) = g(x)\left(1 - \left(\frac{f_1}{g(x)}\right)^2\right)$ $g(x) = 1 + 9\sqrt[4]{\frac{\sum_{i=2}^{n} x_i}{9}}$	$n = 10$ $0 \leq x_i \leq 1$ $i = 1, 2, ..., 10$

Tabla B.1: Definición de la familia de problemas ZDT.

Problema	Definición	Restricciones
DTLZ1	$F = (f_1(x), f_2(x), f_3(x))$ $f_1 = \frac{1}{2}x_1x_2(1+g(x))$ $f_2 = \frac{1}{2}x_1(1-x_2)(1+g(x))$ $f_3 = \frac{1}{2}(1-x_1)(1+g(x))$ $g(x) = 100\left(10 + \sum_{i=3}^{n}(x_i - 0{,}5)^2 - \cos(20\pi(x_i - 0{,}5))\right)$	$n = 12$ $0 \leq x_i \leq 1$ $i = 1, ..., 12$
DTLZ2	$F = (f_1(x), f_2(x), f_3(x))$ $f_1 = \cos(\frac{\pi}{2}x_1)\cos(\frac{\pi}{2}x_2)(1+g(x))$ $f_2 = \cos(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2)(1+g(x))$ $f_3 = \sin(\frac{\pi}{2}x_1)(1+g(x))$ $g(x) = \sum_{i=3}^{n}(x_i - 0{,}5)^2$	$n = 12$ $0 \leq x_i \leq 1$ $i = 1, ..., 12$
DTLZ3	$F = (f_1(x), f_2(x), f_3(x))$ $f_1 = \cos(\frac{\pi}{2}x_1)\cos(\frac{\pi}{2}x_2)(1+g(x))$ $f_2 = \cos(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2)(1+g(x))$ $f_3 = \sin(\frac{\pi}{2}x_1)(1+g(x))$ $g(x) = 100\left(10 + \sum_{i=3}^{n}(x_i - 0{,}5)^2 - \cos(20\pi(x_i - 0{,}5))\right)$	$n = 12$ $0 \leq x_i \leq 1$ $i = 1, ..., 12$
DTLZ4	$F = (f_1(x), f_2(x), f_3(x))$ $f_1 = \cos(\frac{\pi}{2}x_1^\alpha)\cos(\frac{\pi}{2}x_2^\alpha)(1+g(x))$ $f_2 = \cos(\frac{\pi}{2}x_1^\alpha)\sin(\frac{\pi}{2}x_2^\alpha)(1+g(x))$ $f_3 = \sin(\frac{\pi}{2}x_1^\alpha)(1+g(x))$ $g(x) = \sum_{i=3}^{n}(x_i - 0{,}5)^2$	$n = 12$ $\alpha = 100$ $0 \leq x_i \leq 1$ $i = 1, ..., 12$

Tabla B.2: Definición de la familia de problemas DTLZ (I).

Problema	Definición	Restricciones
DTLZ5	$F = (f_1(x), f_2(x), f_3(x))$ $f_1 = \cos(\frac{\pi}{2}\theta_1)\cos(\frac{\pi}{2}\theta_2)(1+g(x))$ $f_2 = \cos(\frac{\pi}{2}\theta_1)\sin(\frac{\pi}{2}\theta_2)(1+g(x))$ $f_3 = \sin(\frac{\pi}{2}\theta_1)(1+g(x))$ $\theta_1 = x_1\frac{\pi}{2}$ $\theta_2 = \frac{\pi}{4(1+g(x))}(1+2x_2g(x))$ $g(x) = \sum_{i=3}^{n}(x_i - 0{,}5)^2$	$n = 12$ $0 \leq x_i \leq 1$ $i = 1, ..., 12$
DTLZ6	$F = (f_1(x), f_2(x), f_3(x))$ $f_1 = \cos(\frac{\pi}{2}\theta_1)\cos(\frac{\pi}{2}\theta_2)(1+g(x))$ $f_2 = \cos(\frac{\pi}{2}\theta_1)\sin(\frac{\pi}{2}\theta_2)(1+g(x))$ $f_3 = \sin(\frac{\pi}{2}\theta_1)(1+g(x))$ $\theta_1 = x_1\frac{\pi}{2}$ $\theta_2 = \frac{\pi}{4(1+g(x))}(1+2x_2g(x))$ $g(x) = \sum_{i=3}^{n}\sqrt[10]{x_i}$	$n = 12$ $0 \leq x_i \leq 1$ $i = 1, ..., 12$
DTLZ7	$F = (f_1(x), f_2(x), f_3(x))$ $f_1(x) = x_1$ $f_2(x) = x_2$ $f_3(x) = h(1+g(x))$ $g(x) = 1 + \frac{9}{22}\sum_{i=3}^{n} x_i$ $h = 3 - \sum_{i=1}^{2}\left(\frac{f_i}{1+g}(1+\sin(3\pi f_i))\right)$	$n = 22$ $0 \leq x_i \leq 1$ $i = 1, ..., 22$

Tabla B.3: Definición de la familia de problemas DTLZ (II).

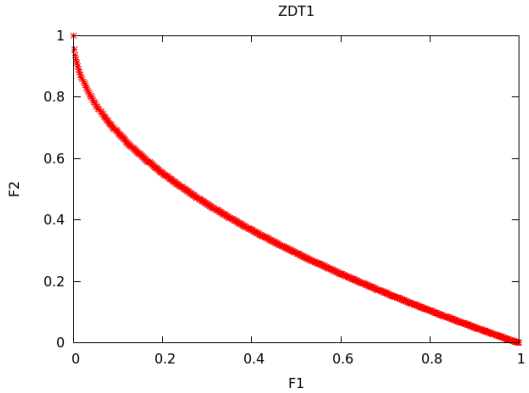


Figura B.1: Frente de Pareto verdadero del problema ZDT1.

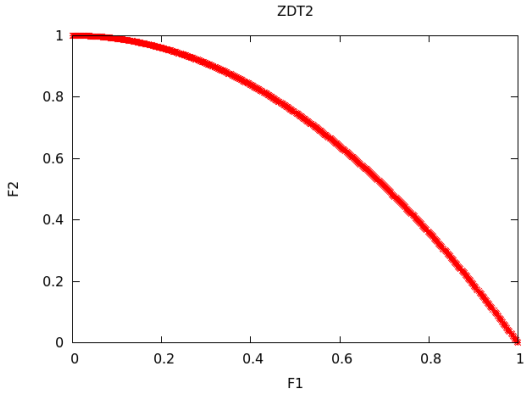


Figura B.3: Frente de Pareto verdadero del problema ZDT2.

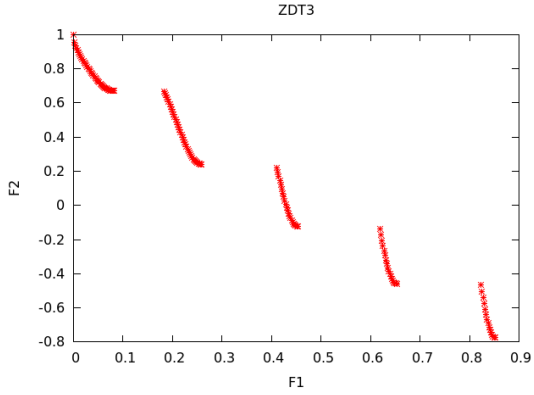


Figura B.2: Frente de Pareto verdadero del problema ZDT3.

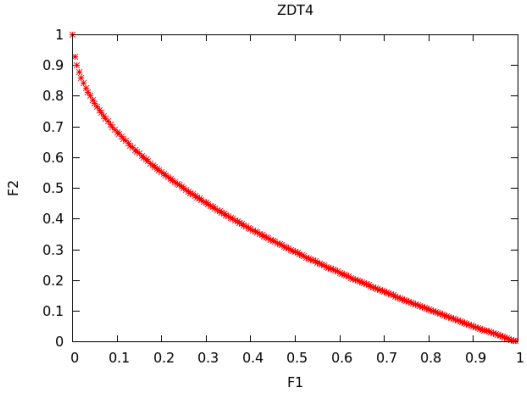


Figura B.4: Frente de Pareto verdadero del problema ZDT4.

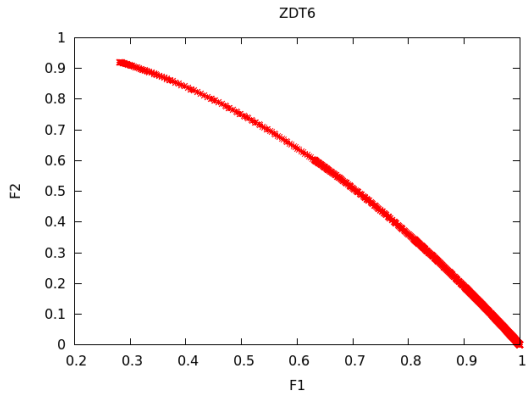


Figura B.5: Frente de Pareto verdadero del problema ZDT6.

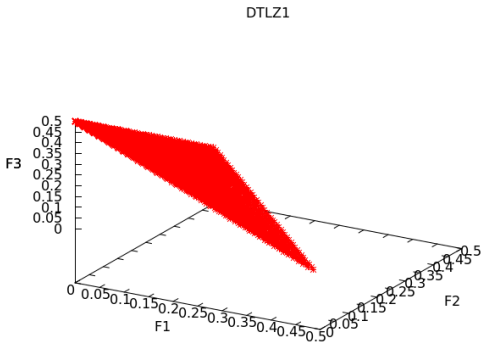


Figura B.6: Frente de Pareto verdadero del problema DTLZ1.

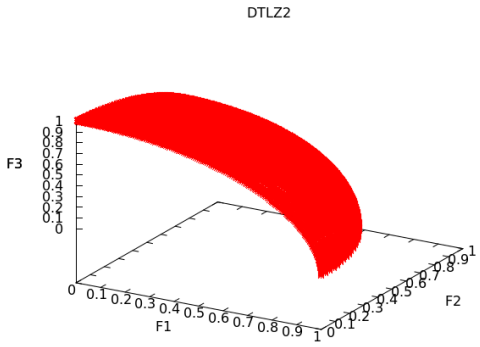


Figura B.8: Frente de Pareto verdadero del problema DTLZ2.

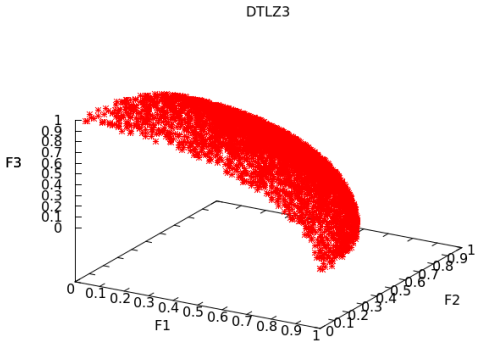


Figura B.7: Frente de Pareto verdadero del problema DTLZ3.

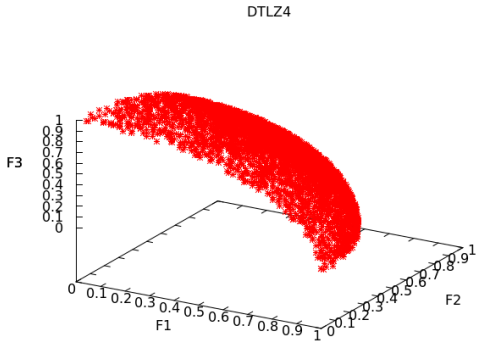


Figura B.9: Frente de Pareto verdadero del problema DTLZ4.

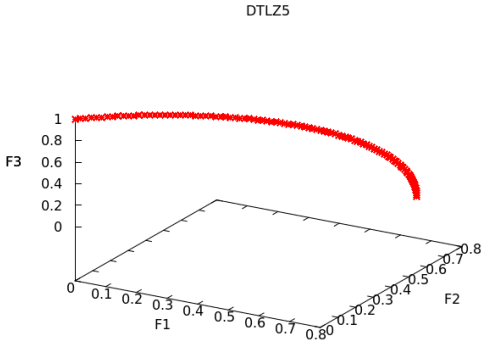


Figura B.10: Frente de Pareto verdadero del problema DTLZ5.

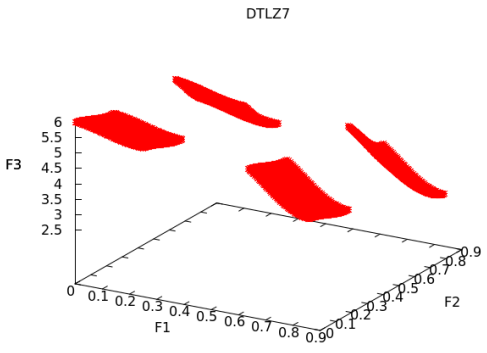


Figura B.11: Frente de Pareto verdadero del problema DTLZ7.

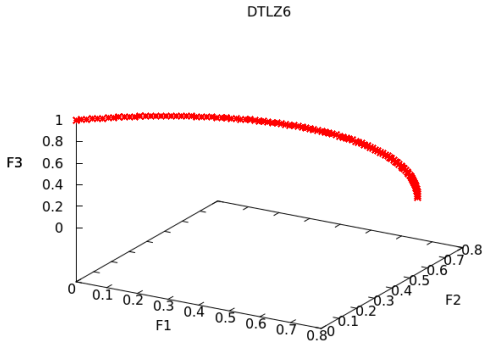


Figura B.12: Frente de Pareto verdadero del problema DTLZ6.

Apéndice C

Resultados completos

El presente apéndice detalla los resultados completos obtenidos en el capítulo 5. En la sección C.1 se detallan los resultados de la comparación entre variantes de mocDE, y en la sección C.2 se detallan los resultados de mocDE y el estado del arte.

En las tablas donde se reportan los resultados de los indicadores unarios se muestra la media (μ), la desviación estándar (σ) y los valores mínimo y máximo de cada indicador para todas las ejecuciones realizadas. En el caso de los indicadores binarios, sólo se muestra la media. En todos los casos, se resalta el mejor resultado. El valor mostrado para I_H está normalizado con base en el máximo valor posible, es decir, el hipervolumen obtenido por el frente de Pareto real.

C.1. Comparación entre las variantes de mocDE

Las tablas C.2 y C.3 muestran los resultados obtenidos por las distintas variantes de mocDE en la familia de problemas ZDT. Así mismo, en las tablas C.4, C.5 y C.6 se reportan los resultados de las mismas variantes pero en la familia de problemas DTLZ.

Los puntos de referencia utilizados para calcular I_H se observan en la tabla C.1.

C.2. Comparación entre mocDE y el estado del arte

Las tablas C.8 y C.9 muestran los resultados obtenidos por mocDE y los algoritmos del estado del arte en la familia de problemas ZDT. Así mismo, en las tablas C.10, C.11 y C.12 se reportan los resultados de los mismos algoritmos pero en la familia de problemas DTLZ.

Los puntos de referencia utilizados para calcular I_H se observan en la tabla C.7.

Problema	Punto de Referencia
ZDT1	(1.1, 1.2)
ZDT2	(1.1, 1.1)
ZDT3	(0.9, 1.2)
ZDT4	(1.1, 32.8)
ZDT5	(1.1, 6.6)
DTLZ1	(72.7, 74.9, 68.2)
DTLZ2	(1.1, 1.1, 1.1)
DTLZ3	(146.6, 133.3, 125.1)
DTLZ4	(1.1, 1.1, 1.1)
DTLZ5	(0.9, 0.9, 1.1)
DTLZ6	(0.8, 0.8, 1.1)
DTLZ7	(1.1, 1.0, 6.2)

Tabla C.1: Puntos de referencia utilizados para calcular el I_H de las variantes de mocDE.

Problema	Indicador	pe-mocDE				ne-mocDE$_{50}$				ne-mocDE$_{20}$				ne-mocDE$_{10}$			
		μ	σ	min	max	μ	σ	min	max	μ	σ	min	max	μ	σ	min	max
ZDT1	I_{IGD}	**0.0003**	**0.0000**	**0.0003**	**0.0003**	0,0004	**0.0000**	0,0004	0,0005	0,0009	0,0001	0,0007	0,0012	0,0024	0,0003	0,0018	0,0031
	I_{Δ_p}	**0.0070**	**0.0003**	**0.0066**	**0.0076**	0,0098	0,0010	0,0082	0,0116	0,0209	0,0029	0,0158	0,0277	0,0539	0,0076	0,0393	0,0706
	I_S	**0.0142**	**0.0008**	**0.0109**	**0.0157**	0,0163	0,0022	0,0131	0,0216	0,0250	0,0049	0,0141	0,0378	0,0380	0,0069	0,0266	0,0546
	I_H	**0.9940**	**0.0003**	**0.9936**	**0.9945**	0,9901	0,0013	0,9876	0,9918	0,9714	0,0048	0,9584	0,9802	0,9124	0,0123	0,8843	0,9375
ZDT2	I_{IGD}	**0.0003**	**0.0000**	**0.0003**	**0.0003**	0,0005	0,0001	0,0004	0,0006	0,0014	0,0002	0,0010	0,0019	0,0039	0,0008	0,0027	0,0056
	I_{Δ_p}	**0.0060**	**0.0003**	**0.0056**	**0.0070**	0,0114	0,0013	0,0089	0,0142	0,0320	0,0054	0,0215	0,0416	0,0876	0,0186	0,0593	0,1242
	I_S	**0.0070**	**0.0006**	**0.0059**	**0.0083**	0,0124	0,0020	0,0088	0,0172	0,0342	0,0095	0,0223	0,0596	0,0865	0,0310	0,0236	0,1523
	I_H	**0.9905**	**0.0005**	**0.9894**	**0.9913**	0,9806	0,0028	0,9755	0,9856	0,9278	0,0138	0,8985	0,9512	0,7596	0,0580	0,5789	0,8657
ZDT3	I_{IGD}	**0.0021**	**0.0002**	**0.0019**	**0.0025**	0,0022	**0.0002**	**0.0019**	0,0028	0,0028	0,0004	0,0021	0,0036	0,0051	0,0008	0,0040	0,0072
	I_{Δ_p}	**0.0242**	**0.0019**	**0.0220**	**0.0295**	0,0259	0,0025	0,0221	0,0321	0,0325	0,0046	0,0248	0,0414	0,0601	0,0092	0,0466	0,0843
	I_S	**0.0311**	**0.0018**	0,0274	**0.0351**	0,0327	0,0037	**0.0263**	0,0451	0,0369	0,0089	0,0288	0,0736	0,0502	0,0127	0,0264	0,0819
	I_H	**0.9925**	**0.0008**	**0.9892**	**0.9936**	0,9879	0,0018	0,9818	0,9905	0,9676	0,0057	0,9583	0,9800	0,9122	0,0141	0,8826	0,9510
ZDT4	I_{IGD}	**0.0161**	**0.0077**	**0.0045**	**0.0327**	0,2351	0,0735	0,0862	0,3906	0,6380	0,1640	0,3302	0,9615	1,1765	0,2465	0,7353	1,6420
	I_{Δ_p}	**0.2317**	**0.1111**	**0.0641**	**0.4846**	3,9446	1,1349	2,1146	6,6981	10,6938	2,3165	6,2144	15,0909	18,7420	2,8043	13,7032	25,8665
	I_S	**0.0339**	**0.0142**	0,0129	**0.0693**	0,4972	0,3687	**0.0000**	1,3992	0,8503	0,7690	**0.0000**	2,6845	1,2088	1,4374	**0.0000**	5,3900
	I_H	**0.9900**	**0.0047**	**0.9796**	**0.9973**	0,8661	0,0321	0,8108	0,9192	0,6713	0,0651	0,5260	0,7901	0,4199	0,0893	0,1696	0,5658
ZDT6	I_{IGD}	**0.0001**	**0.0000**	**0.0001**	**0.0001**	**0.0000**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0000**	**0.0001**	0,0002	0,0002	**0.0000**	**0.0001**	0,0002
	I_{Δ_p}	0,0290	0,0951	**0.0031**	0,4207	**0.0042**	**0.0006**	0,0034	**0.0066**	0,0053	0,0011	0,0040	0,0096	0,0982	0,2130	0,0050	0,7936
	I_S	0,0325	0,0943	**0.0057**	0,4210	**0.0088**	**0.0018**	0,0061	**0.0133**	0,0116	0,0026	0,0082	0,0188	0,1076	0,2088	0,0102	0,7924
	I_H	**0.9983**	0,0022	0,9896	**0.9993**	**0.9983**	0,0009	0,9955	0,9992	0,9979	**0.0006**	**0.9960**	0,9985	0,9958	0,0024	0,9884	0,9981

Tabla C.2: Resultados de las variantes de mocDE en la familia de problemas ZDT.

Problema	Indicador	Algoritmo A	Algoritmo B			
			$pe-mocDE$	$ne-mocDE_{50}$	$ne-mocDE_{20}$	$ne-mocDE_{10}$
ZDT1	I_C(A,B)	pe-mocDE	—	**0.1829**	**0.6338**	**0.9652**
		ne-mocDE$_{50}$	0,0331	—	**0.5458**	**0.9496**
		ne-mocDE$_{20}$	0,0183	0,0503	—	**0.8242**
		ne-mocDE$_{10}$	0,0014	0,0037	0,0341	—
	$I_{\epsilon+}$(A,B)	pe-mocDE	—	**0.0102**	**0.0081**	**0.0018**
		ne-mocDE$_{50}$	0,0172	—	**0.0128**	**0.0035**
		ne-mocDE$_{20}$	0,0387	0,0374	—	**0.0163**
		ne-mocDE$_{10}$	0,0848	0,0834	0,0768	—
	$I_{\epsilon*}$(A,B)	pe-mocDE	—	3,2752	**2.8767**	**1.1540**
		ne-mocDE$_{50}$	**1.0793**	—	**1.0465**	**1.0123**
		ne-mocDE$_{20}$	458,9800	490,6647	—	**33.7203**
		ne-mocDE$_{10}$	21362,0167	22840,0617	19039,0111	—
ZDT2	I_C(A,B)	pe-mocDE	—	**0.1437**	**0.6661**	**0.9639**
		ne-mocDE$_{50}$	0,0276	—	**0.5889**	**0.9488**
		ne-mocDE$_{20}$	0,0197	0,0458	—	**0.8278**
		ne-mocDE$_{10}$	0,0015	0,0032	0,0301	—
	$I_{\epsilon+}$(A,B)	pe-mocDE	—	**0.0094**	**0.0067**	**0.0012**
		ne-mocDE$_{50}$	0,0214	—	**0.0137**	**0.0024**
		ne-mocDE$_{20}$	0,0558	0,0533	—	**0.0160**
		ne-mocDE$_{10}$	0,1303	0,1279	0,1185	—
	$I_{\epsilon*}$(A,B)	pe-mocDE	—	**1.0150**	**1.0108**	**1.0016**
		ne-mocDE$_{50}$	1,0330	—	**1.0214**	**1.0036**
		ne-mocDE$_{20}$	3357,6992	3357,6967	—	**1.0340**
		ne-mocDE$_{10}$	92098,7609	92098,7609	73686,3543	—
ZDT3	I_C(A,B)	pe-mocDE	—	**0.2982**	**0.6185**	**0.9246**
		ne-mocDE$_{50}$	0,0818	—	**0.5001**	**0.8703**
		ne-mocDE$_{20}$	0,0337	0,0827	—	**0.7063**
		ne-mocDE$_{10}$	0,0039	0,0086	0,0567	—
	$I_{\epsilon+}$(A,B)	pe-mocDE	—	**0.0120**	**0.0098**	**0.0044**
		ne-mocDE$_{50}$	0,0183	—	**0.0127**	**0.0062**
		ne-mocDE$_{20}$	0,0499	0,0472	—	**0.0160**
		ne-mocDE$_{10}$	0,1250	0,1216	0,1062	—
	$I_{\epsilon*}$(A,B)	pe-mocDE	—	**443.2974**	**44.8721**	**5.5832**
		ne-mocDE$_{50}$	6751,9740	—	**1224.1240**	**349.0649**
		ne-mocDE$_{20}$	24342,5556	19213,1342	—	**2532.5583**
		ne-mocDE$_{10}$	77415,6072	71450,4121	55644,8946	—
ZDT4	I_C(A,B)	pe-mocDE	—	**0.9895**	**0.9796**	**0.9718**
		ne-mocDE$_{50}$	0,0000	—	**0.9918**	**1.0000**
		ne-mocDE$_{20}$	0,0000	0,0009	—	**0.9532**
		ne-mocDE$_{10}$	0,0000	0,0000	0,0131	—
	$I_{\epsilon+}$(A,B)	pe-mocDE	—	**0.0000**	**0.0000**	**0.0000**
		ne-mocDE$_{50}$	3,3611	—	**0.0042**	**0.0000**
		ne-mocDE$_{20}$	9,0984	5,7458	—	**0.0378**
		ne-mocDE$_{10}$	16,7232	13,3620	7,6727	—
	$I_{\epsilon*}$(A,B)	pe-mocDE	—	**2.5826**	**2.5826**	**2.5826**
		ne-mocDE$_{50}$	19,9100	—	**1.0017**	**1.0000**
		ne-mocDE$_{20}$	51,6097	3,0368	—	**1.0083**
		ne-mocDE$_{10}$	93,7397	5,3372	2,1420	—
ZDT6	I_C(A,B)	pe-mocDE	—	0,0000	0,0000	**0.0014**
		ne-mocDE$_{50}$	**0.0006**	—	0,0000	**0.0012**
		ne-mocDE$_{20}$	**0.0006**	0,0000	—	**0.0012**
		ne-mocDE$_{10}$	0,0006	0,0000	0,0000	—
	$I_{\epsilon+}$(A,B)	pe-mocDE	—	**0.0170**	**0.0155**	**0.0136**
		ne-mocDE$_{50}$	0,0241	—	**0.0211**	**0.0179**
		ne-mocDE$_{20}$	0,0314	0,0300	—	**0.0251**
		ne-mocDE$_{10}$	0,0466	0,0451	0,0431	—
	$I_{\epsilon*}$(A,B)	pe-mocDE	—	1,0599	1,0905	1,1505
		ne-mocDE$_{50}$	**1.0554**	—	1,0920	1,1441
		ne-mocDE$_{20}$	**1.0631**	**1.0696**	—	1,1429
		ne-mocDE$_{10}$	**1.0800**	**1.0812**	**1.1009**	—

Tabla C.3: Resultados de las variantes de mocDE en la familia de problemas ZDT.

Problema	Indicador	pe-mocDE				ne-mocDE$_{50}$				ne-mocDE$_{20}$				ne-mocDE$_{10}$			
		μ	σ	min	max	μ	σ	min	max	μ	σ	min	max	μ	σ	min	max
DTLZ1	I_{IGD}	**0.0164**	**0.0066**	**0.0037**	**0.0313**	0,1616	0,0324	0,0739	0,2675	0,3829	0,0894	0,1873	0,5163	0,5824	0,0767	0,4082	0,6963
	I_{Δ_p}	**1.0627**	**0.4465**	**0.2133**	**2.0670**	13,0044	2,6496	8,1538	18,6585	29,2375	4,3853	18,8609	38,5384	43,8121	3,1813	38,5063	50,6064
	I_S	**0.1365**	**0.0577**	**0.0447**	**0.2545**	2,2535	0,5542	1,2300	3,3927	5,0151	1,6959	2,2786	9,9877	5,9482	1,5227	4,1706	11,2676
	I_H	**1.0000**	**0.0000**	**1.0000**	**1.0000**	0,9965	0,0018	0,9917	0,9994	0,9591	0,0150	0,9309	0,9865	0,8703	0,0254	0,8115	0,9102
DTLZ2	I_{IGD}	**0.0008**	**0.0000**	**0.0008**	**0.0008**	**0.0008**	**0.0000**	**0.0008**	0,0009	0,0010	**0.0000**	0,0009	0,0011	0,0012	0,0001	0,0011	0,0013
	I_{Δ_p}	**0.0793**	**0.0018**	**0.0751**	**0.0828**	0,0841	0,0031	0,0800	0,0930	0,0977	0,0042	0,0883	0,1094	0,1217	0,0063	0,1056	0,1347
	I_S	**0.0650**	0,0064	**0.0506**	**0.0750**	0,0664	**0.0057**	0,0561	0,0793	0,0744	0,0087	0,0589	0,0929	0,0897	0,0109	0,0708	0,1090
	I_H	**0.8956**	**0.0032**	**0.8885**	**0.9012**	0,8824	0,0050	0,8733	0,8908	0,8511	0,0091	0,8309	0,8659	0,7803	0,0156	0,7393	0,8041
DTLZ3	I_{IGD}	**0.0321**	**0.0152**	**0.0091**	**0.0724**	0,3664	0,0845	0,2083	0,5418	0,8280	0,1797	0,3212	1,1221	1,3405	0,2075	0,8390	1,6241
	I_{Δ_p}	**2.1341**	**1.0264**	**0.5873**	**4.6864**	28,8088	6,4745	15,2155	40,0464	66,9989	10,3178	37,1976	83,2247	107,0371	10,0556	77,9817	124,4320
	I_S	**0.3069**	**0.1516**	**0.1261**	**0.7348**	5,4593	2,3097	1,4025	10,0610	15,6601	7,5643	6,6439	37,6012	24,8385	10,2733	11,8240	57,8598
	I_H	**1.0000**	**0.0000**	**1.0000**	**1.0000**	0,9921	0,0048	0,9821	0,9988	0,9098	0,0333	0,8390	0,9792	0,6673	0,0851	0,5133	0,8688
DTLZ4	I_{IGD}	**0.0035**	**0.0007**	**0.0023**	**0.0052**	0,0040	**0.0007**	0,0026	0,0055	0,0052	0,0009	0,0036	0,0071	0,0064	0,0009	0,0045	0,0079
	I_{Δ_p}	**0.2271**	**0.0452**	**0.1498**	**0.3340**	0,2562	0,0475	0,1676	0,3531	0,3363	0,0563	0,2323	0,4574	0,4085	0,0564	0,2882	0,5075
	I_S	**0.1187**	**0.0180**	0,0867	**0.1589**	0,1700	0,0536	0,0798	0,3249	0,2399	0,0857	0,1252	0,4396	0,2751	0,0763	**0.0632**	0,4187
	I_H	**0.8214**	**0.0293**	**0.7328**	**0.8646**	0,7671	0,0414	0,6281	0,8305	0,6645	0,0527	0,5494	0,7436	0,5234	0,0653	0,3838	0,6305
DTLZ5	I_{IGD}	**0.0001**	**0.0000**	**0.0001**	**0.0002**	0,0002	**0.0000**	**0.0001**	**0.0002**	0,0003	**0.0000**	0,0003	0,0004	0,0005	0,0001	0,0004	0,0007
	I_{Δ_p}	**0.0144**	**0.0005**	**0.0134**	**0.0157**	0,0182	0,0021	**0.0134**	0,0231	0,0320	0,0038	0,0267	0,0423	0,0604	0,0092	0,0467	0,0791
	I_S	**0.0151**	**0.0021**	**0.0111**	**0.0218**	0,0216	0,0048	0,0149	0,0330	0,0365	0,0089	0,0221	0,0570	0,0513	0,0129	0,0292	0,0834
	I_H	**0.9814**	**0.0009**	**0.9790**	**0.9828**	0,9632	0,0037	0,9539	0,9712	0,9163	0,0075	0,9022	0,9307	0,8304	0,0168	0,7964	0,8625
DTLZ6	I_{IGD}	**0.0001**	**0.0000**	**0.0001**	**0.0001**	**0.0000**	**0.0001**	**0.0001**	**0.0001**	**0.0001**	**0.0000**	**0.0001**	0,0002	0,0002	**0.0000**	**0.0001**	0,0002
	I_{Δ_p}	**0.0138**	**0.0002**	0,0134	**0.0143**	0,0140	0,0003	0,0133	0,0150	0,0143	0,0005	0,0131	0,0154	0,0159	0,0013	**0.0127**	0,0182
	I_S	**0.0121**	**0.0016**	**0.0094**	**0.0155**	0,0125	0,0023	0,0096	0,0215	0,0157	0,0028	0,0119	0,0217	0,0214	0,0042	0,0141	0,0291
	I_H	**0.9825**	**0.0005**	**0.9815**	0,9834	0,9822	0,0007	0,9806	**0.9839**	0,9814	0,0011	0,9790	0,9837	0,9769	0,0027	0,9729	0,9836
DTLZ7	I_{IGD}	0,0021	**0.0001**	0,0019	0,0023	**0.0020**	**0.0001**	0,0018	**0.0022**	**0.0020**	**0.0001**	**0.0017**	**0.0022**	**0.0020**	0,0002	0,0018	0,0024
	I_{Δ_p}	0,1992	0,0094	0,1817	0,2187	0,1968	**0.0089**	0,1734	0,2174	**0.1960**	0,0119	**0.1635**	**0.2162**	0,1984	0,0165	0,1747	0,2347
	I_S	**0.1292**	0,0141	0,1102	0,1783	0,1308	**0.0099**	0,1140	**0.1522**	0,1412	0,0188	0,1104	0,1845	0,1604	0,0256	**0.0950**	0,1967
	I_H	**0.8714**	**0.0054**	0,8556	0,8808	0,8705	0,0057	**0.8585**	**0.8825**	0,8508	0,0099	0,8277	0,8665	0,7884	0,0210	0,7435	0,8170

Tabla C.4: Resultados de las variantes de mocDE en la familia de problemas DTLZ.

Problema	Indicador	Algoritmo A	Algoritmo B			
			$pe-mocDE$	$ne-mocDE_{50}$	$ne-mocDE_{20}$	$ne-mocDE_{10}$
DTLZ1	I_C(A,B)	pe-mocDE	—	**1.0000**	**1.0000**	**1.0000**
		ne-mocDE$_{50}$	0,0000	—	**0.9528**	**0.9998**
		ne-mocDE$_{20}$	0,0000	0,0112	—	**0.8862**
		ne-mocDE$_{10}$	0,0000	0,0000	0,0233	—
	$I_{\epsilon+}$(A,B)	pe-mocDE	—	**0.0000**	**0.0000**	**0.0000**
		ne-mocDE$_{50}$	6,0453	—	**0.4018**	**0.0005**
		ne-mocDE$_{20}$	14,1654	11,9931	—	**2.3262**
		ne-mocDE$_{10}$	21,9097	19,8112	14,7473	—
	$I_{\epsilon*}$(A,B)	pe-mocDE	—	**1.0000**	**1.0000**	**1.0000**
		ne-mocDE$_{50}$	16,1734	—	**1.0736**	**1.0000**
		ne-mocDE$_{20}$	34,6028	4,0971	—	**1.1797**
		ne-mocDE$_{10}$	49,1075	5,8502	2,7221	—
DTLZ2	I_C(A,B)	pe-mocDE	—	**0.1930**	**0.3803**	**0.6351**
		ne-mocDE$_{50}$	0,0050	—	**0.2912**	**0.5801**
		ne-mocDE$_{20}$	0,0005	0,0116	—	**0.4310**
		ne-mocDE$_{10}$	0,0001	0,0005	0,0145	—
	$I_{\epsilon+}$(A,B)	pe-mocDE	—	0,0798	**0.0802**	**0.0736**
		ne-mocDE$_{50}$	**0.0782**	—	**0.0791**	**0.0725**
		ne-mocDE$_{20}$	0,0862	0,0859	—	**0.0769**
		ne-mocDE$_{10}$	0,1155	0,1134	0,1092	—
	$I_{\epsilon*}$(A,B)	pe-mocDE	—	1,1225	**1.1202**	**1.1108**
		ne-mocDE$_{50}$	**1.1215**	—	**1.1184**	**1.1084**
		ne-mocDE$_{20}$	5,1605	5,1577	—	5,1411
		ne-mocDE$_{10}$	1,1815	1,1760	**1.1687**	—
DTLZ3	I_C(A,B)	pe-mocDE	—	**1.0000**	**1.0000**	**1.0000**
		ne-mocDE$_{50}$	0,0000	—	**0.9608**	**0.9997**
		ne-mocDE$_{20}$	0,0000	0,0173	—	**0.9285**
		ne-mocDE$_{10}$	0,0000	0,0001	0,0342	—
	$I_{\epsilon+}$(A,B)	pe-mocDE	—	**0.0000**	**0.0000**	**0.0000**
		ne-mocDE$_{50}$	18,3338	—	**0.5406**	**0.0003**
		ne-mocDE$_{20}$	41,9089	36,1368	—	**2.6854**
		ne-mocDE$_{10}$	69,1623	64,7494	46,8477	—
	$I_{\epsilon*}$(A,B)	pe-mocDE	—	**1.0000**	**1.0000**	**1.0000**
		ne-mocDE$_{50}$	15,3427	—	**1.0366**	**1.0002**
		ne-mocDE$_{20}$	34,4129	3,7160	—	**1.0605**
		ne-mocDE$_{10}$	349,1737	305,2106	302,1788	—
DTLZ4	I_C(A,B)	pe-mocDE	—	**0.2419**	**0.4456**	**0.6601**
		ne-mocDE$_{50}$	0,0041	—	**0.3937**	**0.6547**
		ne-mocDE$_{20}$	0,0010	0,0104	—	**0.5524**
		ne-mocDE$_{10}$	0,0003	0,0008	0,0149	—
	$I_{\epsilon+}$(A,B)	pe-mocDE	—	**0.1004**	**0.0715**	**0.0391**
		ne-mocDE$_{50}$	0,1403	—	**0.0965**	**0.0559**
		ne-mocDE$_{20}$	0,1898	0,1729	—	**0.0928**
		ne-mocDE$_{10}$	0,2403	0,2221	0,1962	—
	$I_{\epsilon*}$(A,B)	pe-mocDE	—	30,3177	31,6908	33,0010
		ne-mocDE$_{50}$	**16.7175**	—	18,5221	19,4577
		ne-mocDE$_{20}$	**7.2712**	**7.4802**	—	8,0788
		ne-mocDE$_{10}$	**2.3197**	**2.4942**	**2.4138**	—
DTLZ5	I_C(A,B)	pe-mocDE	—	**0.5173**	**0.8284**	**0.9373**
		ne-mocDE$_{50}$	0,0035	—	**0.6371**	**0.8817**
		ne-mocDE$_{20}$	0,0000	0,0077	—	**0.6591**
		ne-mocDE$_{10}$	0,0000	0,0003	0,0170	—
	$I_{\epsilon+}$(A,B)	pe-mocDE	—	**0.0082**	**0.0091**	**0.0075**
		ne-mocDE$_{50}$	0,0149	—	**0.0116**	**0.0100**
		ne-mocDE$_{20}$	0,0328	0,0301	—	**0.0195**
		ne-mocDE$_{10}$	0,0613	0,0587	0,0514	—
	$I_{\epsilon*}$(A,B)	pe-mocDE	—	**1.0143**	**1.0162**	**1.0126**
		ne-mocDE$_{50}$	1,0240	—	**1.0203**	**1.0169**
		ne-mocDE$_{20}$	1,0524	1,0486	—	**1.0325**
		ne-mocDE$_{10}$	1,0992	1,0946	1,0826	—

Tabla C.5: Resultados de las variantes de mocDE en la familia de problemas DTLZ (I).

Problema	Indicador	Algoritmo A	Algoritmo B			
			$pe - mocDE$	$ne - mocDE_{50}$	$ne - mocDE_{20}$	$ne - mocDE_{10}$
DTLZ6	I_C(A,B)	pe-mocDE	—	**0.0249**	**0.0287**	**0.0349**
		ne-mocDE$_{50}$	0,0242	—	**0.0297**	**0.0362**
		ne-mocDE$_{20}$	0,0246	0,0262	—	**0.0368**
		ne-mocDE$_{10}$	0,0238	0,0253	0,0292	—
	$I_{\epsilon+}$(A,B)	pe-mocDE	—	0,0046	0,0067	**0.0108**
		ne-mocDE$_{50}$	**0.0045**	—	0,0070	**0.0110**
		ne-mocDE$_{20}$	**0.0061**	**0.0062**	—	**0.0111**
		ne-mocDE$_{10}$	0,0116	0,0116	0,0116	—
	$I_{\epsilon*}$(A,B)	pe-mocDE	—	5,0660	5,2076	5,2137
		ne-mocDE$_{50}$	**2.4646**	—	2,5837	**2.5900**
		ne-mocDE$_{20}$	**1.0107**	**1.0113**	—	**1.0195**
		ne-mocDE$_{10}$	**3.9646**	3,9760	3,9769	—
DTLZ7	I_C(A,B)	pe-mocDE	—	**0.1099**	**0.2151**	**0.4061**
		ne-mocDE$_{50}$	0,0384	—	**0.1866**	**0.3954**
		ne-mocDE$_{20}$	0,0224	0,0481	—	**0.3253**
		ne-mocDE$_{10}$	0,0102	0,0172	0,0373	—
	$I_{\epsilon+}$(A,B)	pe-mocDE	—	0,0806	**0.0826**	**0.1207**
		ne-mocDE$_{50}$	**0.0775**	—	**0.0785**	**0.1109**
		ne-mocDE$_{20}$	0,1118	0,1086	—	**0.1116**
		ne-mocDE$_{10}$	0,3119	0,3008	0,2772	—
	$I_{\epsilon*}$(A,B)	pe-mocDE	—	62,4914	64,5745	64,5796
		ne-mocDE$_{50}$	**11.0501**	—	13,0043	13,0107
		ne-mocDE$_{20}$	**1.1130**	**1.1173**	—	**1.1213**
		ne-mocDE$_{10}$	**1.1497**	**1.1479**	1,1424	—

Tabla C.6: Resultados de las variantes de mocDE en la familia de problemas DTLZ (II).

Problema	Punto de Referencia
ZDT1	(1.1, 4.6)
ZDT2	(1.1, 1.7)
ZDT3	(0.9, 2.3)
ZDT4	(1.1, 41.7)
ZDT5	(1.1, 7.9)
DTLZ1	(117.0, 116.9, 134.5)
DTLZ2	(1.1, 1.1, 1.1)
DTLZ3	(207.2, 226.0, 423.2)
DTLZ4	(1.1, 1.1, 1.2)
DTLZ5	(1.1, 0.9, 1.1)
DTLZ6	(3.4, 8.6, 3.6)
DTLZ7	(1.0, 1.0, 11.4)

Tabla C.7: Puntos de referencia utilizados para calcular el I_H de mocDE y los algoritmos del estado del arte.

Problema	Indicador	mocDE				NSGA-II				MOEA/D				PAES			
		μ	σ	min	max	μ	σ	min	max	μ	σ	min	max	μ	σ	min	max
ZDT1	I_{GD}	**0.0003**	**0.0000**	**0.0003**	**0.0003**	**0.0003**	**0.0000**	**0.0003**	**0.0003**	0,0008	0,0002	0,0004	0,0013	0,0122	0,0028	0,0082	0,0180
	I_{Δ_p}	0,0070	**0.0003**	0,0066	0,0076	**0.0062**	**0.0003**	**0.0057**	**0.0070**	0,0193	0,0067	0,0081	0,0419	0,2795	0,0555	0,1980	0,4016
	I_S	0,0142	0,0008	0,0109	0,0157	**0.0071**	**0.0006**	**0.0059**	**0.0084**	0,0185	0,0071	0,0135	0,0492	0,0711	0,1079	0,0061	0,3086
	I_H	**0.9988**	**0.0001**	**0.9987**	**0.9989**	0,9983	**0.0001**	0,9981	0,9985	0,9938	0,0016	0,9898	0,9973	0,8372	0,1714	0,4919	0,9730
ZDT2	I_{GD}	**0.0003**	**0.0000**	**0.0003**	**0.0003**	**0.0003**	**0.0000**	**0.0003**	**0.0003**	0,0006	0,0002	**0.0003**	0,0012	0,0139	0,0070	0,0056	0,0260
	I_{Δ_p}	**0.0060**	**0.0003**	**0.0056**	**0.0070**	0,0065	0,0004	0,0058	0,0075	0,0136	0,0042	0,0078	0,0257	0,3102	0,1572	0,1243	0,5819
	I_S	0,0070	**0.0006**	0,0059	0,0083	0,0069	**0.0006**	0,0059	**0.0082**	**0.0065**	0,0012	0,0052	0,0098	0,0228	0,0254	**0.0039**	0,0876
	I_H	**0.9957**	**0.0002**	**0.9952**	**0.9961**	0,9926	0,0007	0,9912	0,9936	0,9787	0,0065	0,9613	0,9877	0,6757	0,0599	0,5353	0,7716
ZDT3	I_{GD}	0,0021	0,0002	0,0019	0,0025	**0.0006**	**0.0000**	**0.0005**	**0.0007**	0,0068	0,0021	0,0025	0,0103	0,0310	0,0116	0,0184	0,0482
	I_{Δ_p}	0,0242	0,0019	0,0220	0,0295	**0.0070**	**0.0004**	**0.0065**	**0.0080**	0,0801	0,0245	0,0286	0,1205	0,3609	0,1354	0,2151	0,5619
	I_S	0,0311	0,0018	0,0274	0,0351	**0.0077**	**0.0006**	**0.0067**	**0.0088**	0,0444	0,0161	0,0238	0,0933	0,0580	0,0670	0,0076	0,2096
	I_H	0,9962	**0.0004**	0,9945	0,9967	**0.9975**	**0.0004**	**0.9964**	**0.9982**	0,9345	0,0197	0,8980	0,9741	0,7606	0,1506	0,4064	0,9342
ZDT4	I_{GD}	0,0161	0,0077	0,0045	0,0327	0,0229	0,0121	0,0087	0,0586	**0.0068**	0,0080	**0.0008**	0,0304	0,0139	**0.0043**	0,0068	**0.0226**
	I_{Δ_p}	0,2317	**0.1111**	0,0641	**0.4846**	0,3566	0,1892	0,1241	0,8946	**0.1524**	0,1679	**0.0147**	0,6312	1,3250	1,4512	0,0966	4,2126
	I_S	0,0339	0,0142	0,0129	0,0693	**0.0109**	**0.0057**	**0.0070**	**0.0277**	0,0922	0,1083	0,0159	0,4908	0,9211	1,3092	0,0138	4,1900
	I_H	0,9921	**0.0037**	0,9840	0,9978	0,9893	0,0056	0,9728	0,9960	**0.9966**	0,0039	**0.9870**	**0.9995**	0,8968	0,1293	0,6679	0,9969
ZDT6	I_{GD}	**0.0001**	**0.0000**	**0.0001**	**0.0001**	0,0011	0,0002	0,0008	0,0016	0,0002	0,0001	**0.0001**	0,0005	0,0022	0,0014	**0.0001**	0,0045
	I_{Δ_p}	0,0290	0,0951	**0.0031**	0,4207	0,0626	0,0106	0,0434	0,0864	**0.0117**	**0.0060**	0,0037	**0.0260**	0,2487	0,1863	0,0081	0,7492
	I_S	0,0325	0,0943	0,0057	0,4210	**0.0051**	**0.0005**	0,0039	**0.0060**	0,0052	0,0008	0,0042	0,0076	0,1171	0,1597	**0.0035**	0,5862
	I_H	**0.9985**	0,0018	0,9913	**0.9994**	0,9906	0,0014	0,9878	0,9935	0,9973	**0.0011**	**0.9951**	0,9989	0,5277	0,2614	0,1944	0,9992

Tabla C.8: Resultados de mocDE y los algoritmos del estado del arte en la familia de problemas ZDT.

Problema	Indicador	Algoritmo A	Algoritmo B			
			mocDE	*NSGA − II*	*MOEA/D*	*PAES*
ZDT1	I_C(A,B)	mocDE	—	**0.3072**	**0.9536**	**0.0357**
		NSGA-II	0,0076	—	**0.9061**	0,0179
		MOEA/D	0,0004	0,0003	—	0,0047
		PAES	0,0127	**0.2043**	**0.4496**	—
	$I_{\epsilon+}$(A,B)	mocDE	—	**0.0121**	**0.0030**	**0.0106**
		NSGA-II	0,0127	—	**0.0036**	**0.0115**
		MOEA/D	0,0235	0,0223	—	**0.0197**
		PAES	0,3677	0,3661	0,3528	—
	$I_{\epsilon*}$(A,B)	mocDE	—	**1.0933**	**1.0155**	**1.0562**
		NSGA-II	1939,8397	—	**3.7745**	**2.5625**
		MOEA/D	19415,3127	18,0051	—	**16.8749**
		PAES	352979,4390	69357,5846	60707,0336	—
ZDT2	I_C(A,B)	mocDE	—	**0.3780**	**0.8545**	**0.0291**
		NSGA-II	0,0046	—	**0.7233**	0,0177
		MOEA/D	0,0001	0,0459	—	0,0091
		PAES	0,0069	**0.2018**	**0.3836**	—
	$I_{\epsilon+}$(A,B)	mocDE	—	**0.0091**	**0.0040**	**0.0093**
		NSGA-II	0,0125	—	**0.0059**	**0.0113**
		MOEA/D	0,0409	0,0357	—	**0.0185**
		PAES	0,5870	0,5851	0,5722	—
	$I_{\epsilon*}$(A,B)	mocDE	—	**1.0144**	**1.0054**	**1.0146**
		NSGA-II	5045,5631	—	**1.7980**	**1.1681**
		MOEA/D	39864,9492	12,8372	—	**2.5276**
		PAES	585684,8224	259213,3343	249077,7582	—
ZDT3	I_C(A,B)	mocDE	—	**0.1542**	**0.9779**	0,0439
		NSGA-II	0,0997	—	**0.9935**	0,0344
		MOEA/D	0,0004	0,0000	—	0,0059
		PAES	**0.0564**	**0.1551**	**0.4510**	—
	$I_{\epsilon+}$(A,B)	mocDE	—	0,0218	**0.0011**	**0.0114**
		NSGA-II	**0.0070**	—	**0.0000**	**0.0065**
		MOEA/D	0,1442	0,1424	—	**0.1296**
		PAES	0,5956	0,5942	0,5224	—
	$I_{\epsilon*}$(A,B)	mocDE	—	**291.8517**	**1.0010**	**141.5607**
		NSGA-II	2212,7811	—	**1.0000**	**322.0125**
		MOEA/D	110170,1276	108249,1763	—	**27228.4280**
		PAES	460754,0149	459013,5366	377671,7812	—
ZDT4	I_C(A,B)	mocDE	—	**0.6562**	0,1417	0,0146
		NSGA-II	0,3078	—	0,1058	0,0135
		MOEA/D	**0.8143**	**0.8510**	—	0,0241
		PAES	**0.7710**	**0.7489**	**0.5783**	—
	$I_{\epsilon+}$(A,B)	mocDE	—	**0.0525**	0,1750	0,1989
		NSGA-II	0,1615	—	0,2684	0,2702
		MOEA/D	**0.0454**	**0.0322**	—	**0.0906**
		PAES	**0.1719**	**0.1507**	0,2267	—
	$I_{\epsilon*}$(A,B)	mocDE	—	2,8921	**8.7886**	**2.1943**
		NSGA-II	**2.1174**	—	**12.8377**	**2.8963**
		MOEA/D	44,5183	47,3170	—	**1.8115**
		PAES	96014,7412	102119,2899	13659,7303	—
ZDT6	I_C(A,B)	mocDE	—	**0.9846**	**0.8412**	**0.0077**
		NSGA-II	0,0008	—	0,0004	0,0099
		MOEA/D	0,0008	**0.9849**	—	0,0068
		PAES	0,0001	**0.4649**	**0.4043**	—
	$I_{\epsilon+}$(A,B)	mocDE	—	**0.0021**	**0.0123**	**0.0083**
		NSGA-II	0,0826	—	0,0740	**0.0827**
		MOEA/D	0,0170	**0.0002**	—	**0.0156**
		PAES	0,4066	0,4043	0,4062	—
	$I_{\epsilon*}$(A,B)	mocDE	—	**1.0052**	**1.0209**	41,9239
		NSGA-II	1234,2914	—	135,2997	50818,6292
		MOEA/D	129,3453	**1.0004**	—	5325,9426
		PAES	**2.6926**	**2.4380**	**2.4569**	—

Tabla C.9: Resultados de mocDE y los algoritmos del estado del arte en la familia de problemas ZDT.

Problema	Indicador	mocDE				NSGA-II				MOEA/D				PAES			
		μ	σ	min	max	μ	σ	min	max	μ	σ	min	max	μ	σ	min	max
DTLZ1	I_{GD}	0,0164	0,0066	0,0037	0,0313	0,0839	0,0445	0,0268	0,1964	0,1371	0,1793	**0.0006**	0,5780	**0.0062**	**0.0035**	0,0030	**0.0181**
	I_{Δ_p}	1,0627	**0.4465**	0,2133	**2.0670**	8,0121	5,1015	2,0151	24,1942	13,8322	17,0127	**0.0303**	64,1157	**0.4499**	0,7339	0,1501	4,2975
	I_S	**0.1365**	**0.0577**	0,0447	**0.2545**	1,1452	1,3349	0,1436	5,1126	2,3596	3,0333	0,0227	11,1179	0,2328	0,8103	**0.0084**	4,5603
	I_H	**1.0000**	**0.0000**	**1.0000**	**1.0000**	0,9998	0,0004	0,9984	**1.0000**	0,9953	0,0122	0,9391	**1.0000**	0,9968	0,0015	0,9923	0,9983
DTLZ2	I_{GD}	**0.0008**	**0.0000**	0,0008	**0.0008**	**0.0008**	**0.0000**	**0.0007**	**0.0008**	**0.0008**	**0.0000**	**0.0007**	**0.0008**	0,0052	0,0008	0,0032	0,0065
	I_{Δ_p}	0,0793	0,0018	0,0751	0,0828	**0.0751**	0,0035	**0.0697**	0,0820	0,0768	**0.0009**	0,0749	**0.0786**	0,5176	0,0842	0,3192	0,6439
	I_S	0,0650	0,0064	0,0506	0,0750	0,0544	0,0045	0,0458	0,0643	0,0668	**0.0038**	0,0607	0,0784	**0.0220**	0,0070	**0.0084**	**0.0352**
	I_H	**0.8956**	0,0032	**0.8885**	**0.9012**	0,8768	0,0098	0,8546	0,8938	0,8781	**0.0025**	0,8740	0,8841	0,3937	0,0652	0,2505	0,4920
DTLZ3	I_{GD}	0,0321	0,0152	0,0091	0,0724	0,1765	0,0665	0,0714	0,3685	0,2794	0,3707	**0.0013**	1,5272	**0.0120**	**0.0061**	0,0052	**0.0364**
	I_{Δ_p}	2,1341	**1.0264**	0,5873	**4.6864**	22,7077	19,3330	5,2739	98,4840	25,9961	32,1556	**0.0850**	113,9089	**0.9931**	1,1734	0,3301	6,9059
	I_S	**0.3069**	**0.1516**	0,1261	**0.7348**	4,7033	6,5365	0,4294	21,6622	3,6404	4,5000	0,0648	16,7660	0,3114	0,6383	**0.0129**	3,0623
	I_H	**1.0000**	**0.0000**	**1.0000**	**1.0000**	0,9998	0,0002	0,9994	**1.0000**	0,9963	0,0093	0,9553	**1.0000**	0,9958	0,0017	0,9906	0,9979
DTLZ4	I_{GD}	0,0035	0,0007	0,0023	0,0052	**0.0012**	**0.0000**	**0.0011**	**0.0013**	0,0029	0,0024	0,0012	0,0100	0,0143	0,0019	0,0100	0,0153
	I_{Δ_p}	0,2271	0,0452	0,1498	0,3340	**0.0751**	**0.0031**	**0.0695**	**0.0834**	0,1840	0,1567	0,0791	0,6399	0,9194	0,1200	0,6395	0,9777
	I_S	0,1187	0,0180	0,0867	0,1589	**0.0554**	**0.0050**	0,0478	**0.0655**	0,0587	0,0226	0,0106	0,0782	0,1113	0,1854	**0.0000**	0,4323
	I_H	0,8454	0,0255	0,7683	0,8830	**0.8900**	**0.0085**	**0.8695**	**0.9043**	0,8598	0,0687	0,5354	0,8966	0,2006	0,1358	0,1428	0,6263
DTLZ5	I_{GD}	**0.0001**	**0.0000**	**0.0001**	0,0002	**0.0001**	**0.0000**	**0.0001**	**0.0001**	**0.0001**	**0.0000**	**0.0001**	**0.0001**	0,0028	0,0012	0,0010	0,0048
	I_{Δ_p}	0,0144	**0.0005**	0,0134	0,0157	**0.0071**	0,0006	**0.0061**	**0.0088**	0,0153	0,0045	0,0137	0,0374	0,2848	0,1195	0,1017	0,4866
	I_S	0,0151	0,0021	0,0111	0,0218	**0.0097**	**0.0011**	0,0074	**0.0123**	0,0138	0,0093	0,0093	0,0529	0,0138	0,0071	**0.0046**	0,0297
	I_H	0,9830	0,0008	0,9808	0,9844	**0.9974**	**0.0006**	**0.9958**	**0.9986**	0,9809	0,0019	0,9743	0,9830	0,6649	0,1216	0,4388	0,8205
DTLZ6	I_{GD}	**0.0001**	**0.0000**	**0.0001**	**0.0001**	0,0174	0,0010	0,0154	0,0192	0,0006	0,0005	0,0002	0,0024	0,0039	0,0016	0,0012	0,0061
	I_{Δ_p}	**0.0138**	**0.0002**	**0.0134**	**0.0143**	1,9231	0,1113	1,6621	2,1108	0,0914	0,0765	0,0172	0,3376	0,4039	0,1811	0,1178	0,8078
	I_S	**0.0121**	**0.0016**	0,0094	**0.0155**	0,1362	0,0151	0,1105	0,1798	0,0533	0,0481	0,0104	0,2057	0,0689	0,1598	**0.0039**	0,9040
	I_H	**0.9995**	**0.0000**	**0.9995**	**0.9995**	0,7912	0,0188	0,7667	0,8336	0,9953	0,0043	0,9790	0,9987	0,8222	0,0290	0,7833	0,8702
DTLZ7	I_{GD}	0,0021	**0.0001**	0,0019	0,0023	**0.0009**	**0.0001**	**0.0008**	**0.0012**	0,0036	0,0027	0,0017	0,0109	0,0089	0,0024	0,0053	0,0115
	I_{Δ_p}	0,1992	0,0094	0,1817	0,2187	**0.0899**	**0.0078**	**0.0799**	**0.1167**	0,3955	0,2522	0,1652	1,0586	0,8618	0,2304	0,5171	1,1147
	I_S	0,1292	0,0141	0,1102	0,1783	0,0690	**0.0059**	0,0535	**0.0780**	0,1264	0,0511	0,0282	0,2076	**0.0372**	0,0263	**0.0125**	0,1317
	I_H	**0.9646**	**0.0015**	**0.9602**	0,9672	0,9624	0,0059	0,9490	**0.9728**	0,8988	0,0229	0,8430	0,9309	0,6423	0,2512	0,1814	0,8811

Tabla C.10: Resultados de mocDE y los algoritmos del estado del arte en la familia de problemas DTLZ.

Problema	Indicador	Algoritmo A	Algoritmo B			
			mocDE	*NSGA − II*	*MOEA/D*	*PAES*
DTLZ1	I_C(A,B)	mocDE	—	**0.9904**	**0.6557**	0,0264
		NSGA-II	0,0017	—	0,3874	0,0003
		MOEA/D	0,2392	**0.4418**	—	0,0710
		PAES	**0.4356**	**0.6258**	**0.5366**	—
	$I_{\epsilon+}$(A,B)	mocDE	—	**0.0086**	**0.2152**	0,5467
		NSGA-II	2,7682	—	**1.7941**	2,8995
		MOEA/D	4,3779	3,7240	—	4,4261
		PAES	**0.3736**	**0.3630**	**0.3730**	—
	$I_{\epsilon*}$(A,B)	mocDE	—	**1.0114**	**1.7860**	**4.1740**
		NSGA-II	349976,4876	—	16571,9186	**82.7071**
		MOEA/D	3788,8168	**689.2635**	—	**89.0216**
		PAES	361903,4032	252385,3753	67254,1126	—
DTLZ2	I_C(A,B)	mocDE	—	**0.0222**	**0.2418**	**0.0061**
		NSGA-II	0,0044	—	**0.0360**	0,0027
		MOEA/D	0,0068	0,0046	—	0,0018
		PAES	0,0032	**0.0059**	**0.0160**	—
	$I_{\epsilon+}$(A,B)	mocDE	—	**0.1014**	**0.0437**	**0.0903**
		NSGA-II	0,1800	—	0,1787	**0.0933**
		MOEA/D	0,0979	**0.1100**	—	**0.1036**
		PAES	0,7331	0,7056	0,7331	—
	$I_{\epsilon*}$(A,B)	mocDE	—	**1.1735**	**1.0862**	**1.1523**
		NSGA-II	144748,4553	—	37431,2771	**9.8529**
		MOEA/D	67,0391	**15.4861**	—	**1.9911**
		PAES	674812,1680	252530,0816	324288,2939	—
DTLZ3	I_C(A,B)	mocDE	—	**0.9996**	**0.6180**	0,0420
		NSGA-II	0,0000	—	**0.4042**	0,0009
		MOEA/D	0,2357	0,3929	—	0,1050
		PAES	**0.3546**	**0.5249**	**0.4590**	—
	$I_{\epsilon+}$(A,B)	mocDE	—	**0.0012**	**0.5999**	1,4145
		NSGA-II	8,1380	—	**4.9019**	8,4637
		MOEA/D	12,6178	11,1733	—	12,6794
		PAES	**0.9886**	**0.9761**	**0.9877**	—
	$I_{\epsilon*}$(A,B)	mocDE	—	**1.0003**	**1.4620**	**2.9935**
		NSGA-II	416845,4902	—	7462,1463	**507.9522**
		MOEA/D	5891,1202	**1662.3479**	—	**182.9857**
		PAES	921766,6866	632961,9617	132743,2069	—
DTLZ4	I_C(A,B)	mocDE	—	**0.0188**	**0.1080**	0,0172
		NSGA-II	0,0065	—	**0.0229**	**0.0068**
		MOEA/D	0,0111	0,0100	—	0,0026
		PAES	**0.0614**	0,0038	**0.0384**	—
	$I_{\epsilon+}$(A,B)	mocDE	—	**0.1662**	**0.1425**	**0.0133**
		NSGA-II	0,1849	—	0,1846	**0.1174**
		MOEA/D	0,1586	**0.1559**	—	**0.0114**
		PAES	0,9638	0,9521	0,9623	—
	$I_{\epsilon*}$(A,B)	mocDE	—	**4.5330**	**27.1076**	**26.4156**
		NSGA-II	135583,1489	—	93819,9909	**93625.9635**
		MOEA/D	18451,1499	**896.8074**	—	**4.9309**
		PAES	894503,4408	452850,1570	490909,7774	—
DTLZ5	I_C(A,B)	mocDE	—	0,0020	**0.3023**	0,0214
		NSGA-II	**0.0782**	—	**0.1903**	**0.0430**
		MOEA/D	0,0377	0,0011	—	0,0143
		PAES	**0.0542**	0,0043	**0.1177**	—
	$I_{\epsilon+}$(A,B)	mocDE	—	0,0262	**0.0046**	**0.0216**
		NSGA-II	**0.0078**	—	**0.0075**	**0.0081**
		MOEA/D	0,0095	0,0265	—	**0.0220**
		PAES	0,5499	0,5499	0,5499	—
	$I_{\epsilon*}$(A,B)	mocDE	—	**1.0444**	**1.0077**	**1.0372**
		NSGA-II	80,7762	—	57,5902	**1.0133**
		MOEA/D	6,5056	**1.8791**	—	**1.0453**
		PAES	549916,7838	79694,1601	356232,9212	—

Tabla C.11: Resultados de mocDE y los algoritmos del estado del arte en la familia de problemas DTLZ (I).

Problema	Indicador	Algoritmo A	Algoritmo B			
			$mocDE$	$NSGA-II$	$MOEA/D$	$PAES$
DTLZ6	I_C(A,B)	mocDE	—	**0.9983**	**0.9451**	**0.0104**
		NSGA-II	0,0000	—	0,0000	0,0002
		MOEA/D	0,0000	**0.9937**	—	0,0058
		PAES	0,0012	**0.7497**	**0.3875**	—
	$I_{\epsilon+}$(A,B)	mocDE	—	**-0.0000**	**0.0037**	**0.0165**
		NSGA-II	1,3309	—	1,3205	1,2812
		MOEA/D	0,0798	**0.0062**	—	**0.0612**
		PAES	0,6345	**0.6345**	0,6345	—
	$I_{\epsilon*}$(A,B)	mocDE	—	**1.6239**	**4.2058**	**1.5164**
		NSGA-II	171,1133	—	97,0971	**23.2563**
		MOEA/D	9,3099	**1.1650**	—	**1.9309**
		PAES	572759,7626	68363,3602	317401,6423	—
DTLZ7	I_C(A,B)	mocDE	—	**0.1312**	**0.5262**	**0.0483**
		NSGA-II	0,0020	—	**0.2680**	0,0422
		MOEA/D	0,0008	0,0186	—	0,0156
		PAES	0,0048	**0.0794**	**0.1131**	—
	$I_{\epsilon+}$(A,B)	mocDE	—	**0.1542**	**0.0730**	**0.1259**
		NSGA-II	0,1736	—	**0.0940**	**0.0905**
		MOEA/D	0,7488	0,7118	—	**0.3311**
		PAES	1,9071	1,8674	1,5094	—
	$I_{\epsilon*}$(A,B)	mocDE	—	**1.1983**	**3.9230**	**1.1368**
		NSGA-II	37462,4683	—	1851,1962	**19.1559**
		MOEA/D	309,8735	**27.6673**	—	**4.8843**
		PAES	283791,0577	114238,3751	97017,9213	—

Tabla C.12: Resultados de mocDE y los algoritmos del estado del arte en la familia de problemas DTLZ (II).

Bibliografía

[1] A. Charnes and William W. Cooper. *Management models and industrial applications of linear programming.* Wiley, New York, 1961.

[2] S. S. Rao. Multiobjective optimization in structural design with uncertain parameters and stochastic processes. *AIAA Journal*, 22(11):1670–1678, 1984.

[3] K. Miettinen. *Nonlinear multiobjective optimization.* Kluwer Academic Publishers, Boston, 1999.

[4] H. Benson. *European Journal of Operational Research*, 176(3):1961–1964, 2007.

[5] L. A. Zadeh. Optimality and Non-Scalar-Valued Performance Criteria. *IEEE Transactions on Automatic Control*, 8:59–60, 1963.

[6] Mathematical Programming and Business Administrotion. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26(3):326–344, 1983.

[7] A. P. Wierzbicki. The use of reference objectives in multiobjective optimization-theoretical implications and practical experiences. *Int Inst Applied System Analysis*, (177):1–32, 1979.

[8] A. Jaszkiewicz and R. Słowinski. The 'Light Beam Search' approach – an overview of methodology and applications. *European Journal of Operational Research*, 113(2):300–314, March 1999.

[9] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley, New York, USA, 1966.

[10] David B. Fogel. An analysis of evolutionary programming. In *Proc. of the First Annual Conference on Evolutionary Programming*, pages 43–51, 1992.

[11] I. Rechenberg. *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution.* Frommann-Holzboog, 1973.

[12] John H. Holland. *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, USA, 1992.

[13] Rainer Storn and Kenneth Price. Differential evolution: A simple and efficient heuristic for global optimization over continuous space. *Journal of Global Optimization*, 11:341–359, 1997.

[14] Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[15] Hui Li and Qingfu Zhang. Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302, April 2009.

[16] Qingfu Zhang, Wudong Liu, and Hui Li. The Performance of a New Version of MOEA/D on CEC09 Unconstrained MOP Test Instances. In *2009 IEEE Congress on Evolutionary Computation (CEC'2009)*, pages 203–208, Trondheim, Norway, May 2009. IEEE Press.

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182 –197, April 2002.

[18] K. Ikeda, H. Kita, and S. Kobayashi. Failure of Pareto-based MOEAs: does non-dominated really mean near to optimal? In *Proceedings of the 2001 Congress on Evolutionary Computation*, volume 2, pages 957 –962 vol. 2, 2001.

[19] P. Larrañaga and J.A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[20] Georges Harik, Fernando G. Lobo, and David E. Goldberg. The compact genetic algorithm. In *IEEE Transactions on Evolutionary Computation*, volume 3, pages 523–528, 1998.

[21] Chang Wook Ahn and R.S. Ramakrishna. Elitism-based compact genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 7(4):367 – 385, August 2003.

[22] Ernesto Mininno, Francesco Cupertino, and David Naso. Real-valued compact genetic algorithms for embedded microcontroller optimization. *IEEE Transactions on Evolutionary Computation*, 12(2):203–219, 2008.

[23] W. Gautschi. Error function and Fresnel integrals. In M. Abramowitz and I. Stegun, editors, *Handbook of Mathematical Functions*, chapter 7, pages 295–329. Dover, New York, 1965.

[24] W. J. Cody, Jr. Rational Chebyshev approximations for the error function. 23(107):631–637, July 1969.

[25] E. Mininno, F. Neri, F. Cupertino, and D. Naso. Compact differential evolution. *IEEE Transactions on Evolutionary Computation*, 15(1):32 –54, February 2011.

[26] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8:173–195, 2000.

[27] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Test Problems for Evolutionary Multiobjective Optimization. In Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105–145. Springer, USA, 2005.

[28] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms - a comparative case study. pages 292–301. Springer, 1998.

[29] David A. Van Veldhuizen and David A. Van Veldhuizen. Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations. Technical report, Evolutionary Computation, 1999.

[30] O. Schütze, X. Esquivel, A. Lara, and C. A. Coello Coello. Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 16(4):504 –522, August 2012.

[31] J.R. Schott. *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization.* Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, 1995.

[32] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7:117–132, 2002.