CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN DE COMPUTACIÓN

Un Algoritmo Basado en Evolución Diferencial para Resolver Problemas Multiobjetivo

Tesis que presenta

Luis Vicente Santana Quintero

Para obtener el grado de:

Maestro en Ciencias

En la especialidad de

Ingeniería Eléctrica
Opción Computación

Director de la Tesis: **Dr. Carlos A. Coello Coello**

México, D. F. Noviembre de 2004

II

Agradecimientos

Gracias a mis padres, hermanas y seres queridos por su apoyo incondicional. Esta tesis está dedicada a ustedes.

Gracias a mi asesor, el Dr. Coello por bien aconsejarme en todo momento, además de transmitirme valiosos conocimientos.

Gracias a ti, Karina por tu cariño y apoyo, eres una persona en la que puedo confiar ciegamente.

Agradezco a mis compañeros, por hacer de mi estancia en la maestría un lugar más ameno y alegre, tanto, que llegase a disfrutarlo.

Gracias al CINVESTAV por su apoyo al permitirme cursar todos los cursos necesarios para lograr este trabajo de tesis y facilitarme las instalaciones en las que se desarrolló el mismo.

Gracias al CONACyT por la beca proporcionada durante mi estancia en el programa de maestría en el CINVESTAV. Este trabajo de tesis se derivó del proyecto CONACyT titulado "Nuevos Paradigmas en Optimización Evolutiva Multiobjetivo"(Ref. 34201-A) cuyo responsable es el Dr. Carlos A. Coello Coello.

Y por último, mas no menos importante, gracias a Gregorio por continuamente aportar nuevas ideas, además de que con sus resultados demuestra que nunca un algoritmo podrá llegar a ser tan bueno como se piensa.

Índice general

Lista de Figuras

Lista de Tablas

Índice de Algoritmos

Resumen

En 1995, Rainer Storn y Kenneth V. Price desarrollaron un algoritmo evolutivo para optimización global sobre espacios continuos llamado *Evolución Diferencial*. Este algoritmo realiza una cruza entre tres padres y suele converger rápidamente al óptimo (o a su vecindad) en problemas con una sola función objetivo y sin restricciones. Además, el desempeño de este algoritmo depende del control de pocos parámetros, lo cual facilita su uso en diversos problemas. Estas características hacen de la evolución diferencial un candidato idóneo para resolver problemas multiobjetivo. En los problemas multiobjetivo se busca optimizar dos o más funciones las cuales se encuentran en conflicto unas con otras, es decir, el mejorar una función significa empeorar el desempeño de las otras.

Este trabajo presenta el diseño de un algoritmo (en dos distintas versiones) que resuelve problemas de optimización multiobjetivo utilizando la evolución diferencial en la generación de nuevas soluciones. El algoritmo propuesto utiliza el esquema de la evolución diferencial para cruzar tres padres y obtener un solo hijo el cual compite con un padre para saber quién pasa a la siguiente generación. Además, el algoritmo mantiene dos poblaciones distintas: la población principal (que sirve para la selección de los padres) y una población secundaria. La primera versión del algoritmo maneja una malla adaptativa para retener las mejores soluciones obtenidas y distribuirlas uniformemente, y la segunda versión adopta el concepto de dominancia-ϵ para los mismos fines. Es importante mencionar que el algoritmo propuesto también contempla un esquema distinto en cuanto al manejo de restricciones permitiendo que algunas soluciones no factibles intervengan en la cruza ayudando a resolver eficientemente problemas multiobjetivo con restricciones (los cuales suelen ser más difíciles).

Para medir el desempeño del algoritmo, se implementaron una serie de métricas llamadas *medidas de desempeño*, las cuales ayudan a determinar la cercanía de los individuos obtenidos con respecto al verdadero frente, así como su la bondad de su distribución. Esto permite realizar una comparación cuantitativa directa con algoritmos representativos del estado del arte, tales como: PAES, NSGA-II y PDE. La comparación se realizó en un conjunto de funciones de prueba estándar, encontrándose que el algoritmo propuesto en este trabajo resulta muy competitivo en su desempeño promedio.

Abstract

In 1995, Rainer Storn and Kenneth V. Price developed an evolutionary algorithm for global optimization over continuous search spaces which was called "Differential Evolution". This algorithm performs a recombination among three parents and tends to converge very rapidly to the optimum (or its vicinity) in unconstrained single-objective problems. Furthermore, the performance of this algorithm relies on the control of few parameters, which facilitates its use in diverse problems. These characteristics make differential evolution an ideal candidate for solving multiobjective problems. In multiobjective problems, we aim to optimize two or more objective functions, which are in conflict with each other (i.e., improving one objective implies worsening the others).

This work presents the design of an algorithm (of which two different versions are proposed) that solves multiobjective optimization problems using differential evolution to generate new solutions. The proposed algorithm uses differential evolution's scheme to recombine three parents to obtain a single offspring which competes with a parent to determine who passes to the following generation. Furthermore, the algorithm keeps two different populations: the main population (which is used to select the parents) and a secondary population. The first version of the algorithm uses an adaptive grid to retain the best solutions found and to distribute them in an uniform way. The second version adopts the concept of $\epsilon-$dominance for that sake. It is important to mention that the proposed algorithm also contemplates a different scheme to handle constraints. This scheme allows some infeasible solutions to intervene in the recombination, helping to solve in a more efficient way constrained multiobjective optimization problems (which tend to be the most difficult to solve).

In order to evaluate the performance of the proposed algorithm, we implemented a series of performance measures (or metrics) which help to determine the closeness of the individuals obtained with respect to the true Pareto front. They also allow to determine how good (i.e., uniform) is their distribution. This allows a direct quantitative comparison with algorithms that are representative of the state-of-the-art, such as: PAES, NSGA-II and PDE. This comparison was performed on a standard set of test functions. In our study, we found that the proposed algorithm in this work is, on average, highly competitive.

Capítulo 1

Introducción

Los problemas de optimización han sido sumamente estudiados, debido a la importancia que tienen en la práctica. Algunos problemas han logrado resolverse satisfactoriamente mediante métodos matemáticos, como la optimización lineal. Sin embargo, el problema general de optimización global no lineal permanece como no resuelto, dado que no existe ningún método determinista que garantice converger siempre al óptimo de una función objetivo arbitraria con restricciones arbitrarias [5]. Por esa razón, las heurísticas han tomado un gran auge, y entre ellas destacan los algoritmos evolutivos. La computación evolutiva engloba a un conjunto de heurísticas que basan su funcionamiento en el mecanismo de selección natural propuesto por Charles Darwin, y luego extendido en el denominado Neo-Darwinismo.

En esta tesis se atacan los problemas de optimización multiobjetivo. Estos problemas tienen más de un objetivo a optimizar, y normalmente éstos se encuentran en conflicto unos con otros, por lo que el mejorar un solo objetivo significa empeorar el desempeño de los otros. A diferencia de los problemas de optimización global en los que la solución del problema es un punto único, en los problemas multiobjetivo se trata de encontrar un compromiso entre las distintas funciones objetivo, por lo que se suele obtener un conjunto de soluciones al que se denomina conjunto de óptimos de Pareto (en honor a Vilfredo Pareto quien definió por primera vez este tipo de soluciones).

Para resolver estos problemas multiobjetivo, en este trabajo se utilizó un tipo de algoritmo evolutivo llamado *Evolución Diferencial*, el cual fue desarrollado por Storn y Price en 1995 [39, 40] para optimización en espa-

cios continuos. En el algoritmo propuesto, se seleccionan tres individuos como padres. Uno de los padres es el principal y éste se perturba con el vector diferencia de los otros dos padres. De esta manera se van formando las nuevas soluciones del problema y se utiliza un archivo externo (o población secundaria) para retener a las mejores soluciones que se vayan encontrando a lo largo del proceso evolutivo.

Actualmente existen muchas técnicas para mantener las soluciones en compromiso obtenidas a lo largo del proceso evolutivo en un archivo externo, en este trabajo se realizaron dos distintas versiones en cuanto al manejo del archivo externo, la primera en la que se utiliza un mecanismo de $rejilla$ y la segunda en la que se utilizó el concepto de dominancia-ϵ. El resto de esta tesis está organizado de la siguiente manera:

En el capítulo 2 se proporcionan los conceptos básicos acerca del tipo de problemas que se abordan en la tesis, los paradigmas de la computación evolutiva y sus ventajas con respecto a otras técnicas.

En el capítulo 3 se mencionan algunos conceptos de optimización multiobjetivo, así como el trabajo previo más relevante relacionado con esta tesis. También se mencionan las diversas medidas de desempeño que existen para evaluar algoritmos evolutivos y la interpretación de las mismas.

En el capítulo 4 se describe la técnica de la Evolución Diferencial, incluyendo su historia, la descripción del algoritmo y sus diferentes variantes. Se habla también sobre los diversas técnicas que utilizan la evolución diferencial para resolver problemas multiobjetivo.

En el capítulo 5, se describen las dos técnicas propuestas para resolver problemas multiobjetivo: MyDE y ϵ-MyDE, detallando cada una de sus partes: codificación, selección, cruza, mutación, manejo de restricciones y manejo de la población secundaria para mantener la diversidad y distribución de las soluciones.

En el capítulo 6, se detallan las diez funciones de prueba con los que se realizaron las comparaciones de desempeño entre los dos algoritmos propuestos y distintos algoritmos representativos del estado del arte como lo son: NSGA-II, PAES y PDE. De estas 10 funciones de prueba, se muestran los resultados de desempeño para las diversas técnicas y un análisis detallado de ellos.

En el capítulo 7 se dan las conclusiones de este trabajo, señalando las ventajas y desventajas de cada una de las versiones de la técnica. También se indican algunas rutas posibles de trabajo a futuro que podría derivarse de esta tesis.

Capítulo 2

Optimización y Computación Evolutiva

Los algoritmos evolutivos son ampliamente aplicados a problemas de optimización con espacio de búsqueda muy grandes y complejos (p. ej. accidentados, con ruido, etc). Aquellos problemas donde los algoritmos deterministas no escalan apropiadamente o simplemente no funcionan suelen ser los más adecuados para utilizar algoritmos evolutivos. Debido a que los algoritmos evolutivos son heurísticas, no aseguran obtener el óptimo global en todas las ocasiones, pero normalmente obtienen una muy buena solución en un tiempo computacional considerablemente bajo, además de que no requieren un conocimiento específico sobre el problema a resolver [5].

2.1. Optimización

El principal objetivo de cualquier técnica de optimización es encontrar el óptimo global (o sea, la mejor solución posible) de un problema. Pero no en todos ellos se puede garantizar una convergencia al óptimo global. De forma matemática, el problema se puede expresar como sigue:

Minimizar $f(\vec{x})$ sujeto a:

$$\begin{aligned} g_i(\vec{x}) &\leq 0, \text{ para } i = 1, \ldots, p \\ h_j(\vec{x}) &= 0, \text{ para } j = 1, \ldots, n \end{aligned}$$

donde $\vec{x}$ es el vector solución, $f(\vec{x})$ es la función objetivo, $g_i(\vec{x})$ es el con-

junto de p restricciones de desigualdad, y $h_j(\vec{x})$ es el conjunto de n restricciones de igualdad.

Los problemas cuya complejidad está acotada por un polinomio (por ej. buscar un elemento en una lista no ordenada) son los denominados *problemas P*. Así, podemos decir que un problema pertenece a esta clase si puede ser resuelto en tiempo polinomial por una máquina de Turing *determinística* (significa que el algoritmo se comporta de manera predecible, que no tiene más que un camino a seguir).

Los *problemas NP* son aquellos que pueden ser resueltos en tiempo polinomial por una máquina de Turing no determinista.

También existen los *problemas NP completos*; en los que los algoritmos que sirven para resolverlos requieren un tiempo exponencial en el peor caso. Un ejemplo típico de estos problemas es el del viajero, el cual consiste en encontrar la ruta mínima en la cual se visiten todas las ciudades que se propongan. El mejor algoritmo que se conoce para este problema es de orden $O(n^2 2^n)$. En el problema del viajero el tamaño del espacio de búsqueda crece conforme se aumenta el número de ciudades, con la relación: $\frac{(n-1)!}{2}$, donde n es el número de ciudades.

Existen muchas técnicas clásicas para resolver problemas con ciertas características específicas. Los primeros problemas de optimización fueron los de programación lineal, pero en ellos se requiere que todas las funciones sean combinaciones lineales de las variables de decisión del problema. El método Simplex nos asegura encontrar el óptimo en problemas lineales específicos.

Existen también algunos métodos para resolver problemas no lineales, pero requieren información adicional. Por ejemplo, algunos métodos requieren la primera derivada de la función objetivo, la cual no siempre está disponible en los problemas del mundo real [19].

Actualmente no se conoce un método que asegure encontrar el óptimo global de problemas no lineales (en su caso más general) en un tiempo polinomial, por lo que se ha recurrido al uso de heurísticas para resolver estos problemas. La palabra *heurística* es derivada del griego *heuriskein*, que significa encontrar o descubrir. Actualmente, cuando hablamos de heurística nos referimos a cualquier técnica que mejore el desempeño promedio en la solución de un problema, aunque no mejore necesariamente el desempeño del peor caso [5].

Se han desarrollado varios paradigmas dentro de las heurísticas que

resuelven distintos tipos de problemas. Algunos de estos son:

- *Búsqueda tabú* [17]: utiliza una memoria para guiar la búsqueda, pues almacena las soluciones más recientes.
- *Recocido simulado*[25]: está basado en el enfriamiento de los cristales, y es un algoritmo que requiere de una temperatura inicial, final y una función de variación de la temperatura.
- *Escalando la colina*: se aplica a un punto, y se generan varios estados posibles alrededor de él escogiendo al mejor de ellos. Es una de las búsquedas más simples de implementar.

2.2. Computación Evolutiva

La computación evolutiva engloba a un conjunto de heurísticas que basan su funcionamiento en el mecanismo de la selección natural propuesto por Charles Darwin y luego extendido en el denominado Neo-Darwinismo [20].

En la computación evolutiva, una población está compuesta por varios individuos; un individuo es una solución a un problema y es codificado según las necesidades del problema (p. ej. puede ser una cadena binaria); el medio donde se desenvuelve este individuo es representado por la función objetivo; y las restricciones al problema nos dicen qué tan apto es el individuo para sobrevivir en ese medio.

A estos individuos (llamados padres) de la población se les aplican operadores probabilísticos (cruza y mutación) para obtener nuevos individuos (hijos) que mantienen algunas propiedades de los antecesores los cuales se conservan o se eliminan mediante una selección (deterministica o probabilística), este proceso se realiza con cada uno de los individuos de la población hasta formar una nueva población con nuevos individuos. Este proceso se repite durante un cierto número de iteraciones (llamadas generaciones en computación evolutiva).

Pero existen diversos paradigmas en la computación evolutiva, éstos son:

- Programación evolutiva
- Estrategias evolutivas

- Algoritmos genéticos

2.2.1. Programación evolutiva

Fue propuesta por Lawrence J. Fogel, en los años 1960's [30]. En este paradigma la inteligencia se ve como un comportamiento adaptativo. Fogel utilizó la "programación evolutiva"[15] para hacer evolucionar autómatas de estados finitos, de manera que fuesen capaces de predecir secuencias futuras de símbolos que recibirían. Fogel utilizó una función de recompensa"para indicar si un autómata era bueno o no para predecir un cierto símbolo. El algoritmo de la programación evolutiva se muestra en el Algoritmo 1.

Algoritmo 1 Programación evolutiva

```
Generar aleatoriamente una población
Evaluar la aptitud de la población
Repetir
    Aplicar operador de mutación a cada individuo de la
    población
    Evaluar cada hijo resultante de la mutación
    Realizar la selección mediante torneo entre el padre
    y el hijo
hasta que se cumpla la condición de paro
```

En la programación evolutiva no se aplica ningún operador de recombinación, pues simula la evolución a nivel de las especies y como se sabe, las especies distintas no se pueden recombinar para crear nuevos individuos.

2.2.2. Estrategias evolutivas

Fueron desarrolladas por Ingo Rechenberg en Alemania en 1964 [14], [4] en su intento por resolver un problema de hidrodinámica. La primera versión llamada (1+1)-EE o estrategia evolutiva de dos miembros, usaba

sólo un padre y generaba un solo hijo. Este hijo se mantenía sólo si era mejor que el padre. En la generación de los nuevos individuos se utlizaba la función:

$$\overline{x}_{t+1} = \overline{x}_t + N(0, \sigma)$$

donde *t* se refiere a la *generación* actual, y $N(0, \sigma)$ es un vector de números Gaussianos con una media de 0 y desviación estándar de σ. El algoritmo de las estrategias evolutivas se muestra en el Algoritmo 2.

Algoritmo 2 Estrategias Evolutivas

Generar aleatoriamente una población inicial
Evaluar la aptitud de la población
Repetir
    Aplicar operador de mutación a cada individuo de la población
    Aplicar operador de recombinación
    Evaluar cada hijo resultante
    Realizar la selección
hasta cumplir la condición de paro

Posteriormente, Rechenberg [22] introdujo la población a las estrategias evolutivas, y propuso la $(\mu + 1) - EE$, en la cual hay μ padres que generan un solo hijo, el cual puede reemplazar al peor padre de la población. Schwefel introdujo después el uso de múltiples hijos $(\mu + \lambda) - EE$ y la $(\mu, \lambda) - EE$. En el primer caso, en la selección se toman en cuenta tanto a hijos como a padres; en el segundo caso, en la selección sólo se toman en cuenta a los hijos.

La selección en las estrategias evolutivas es determinística, por lo que sólo los mejores individuos pasan a la siguiente generación. El principal operador es el de mutación y el operador de recombinación juega un papel secundario y puede omitirse.

Las estrategias evolutivas simulan el proceso evolutivo a nivel de individuos por lo que la recombinación es posible.

2.2.3. Algoritmos genéticos

Fueron desarrollados por John Holland a principios de los 1960's [21] en el contexto del aprendizaje de máquina, pero no se dieron a conocer hasta la publicación de su libro en 1975 [23]. Sin embargo, los algoritmos genéticos han sido utilizados mucho en optimización, convirtiéndose en una técnica muy popular actualmente. El algoritmo genético se muestra en el Algoritmo 3.

Algoritmo 3 Algoritmo Genético

Generar aleatoriamente una población inicial
Evaluar la aptitud de la población
Repetir
    Realizar la selección de los padres
    Aplicar operador de recombinación
    Aplicar operador de mutación a cada individuo de la población
    Evaluar cada hijo resultante
hasta que se cumpla la condición de paro

El algoritmo genético enfatiza la importancia del operador de recombinación (operador principal) sobre el de mutación y utiliza una selección probabilística.

La codificación binaria es la más común en los algoritmos genéticos, y dada su universalidad, los operadores básicos de un algoritmo genético se definen a partir de dicha representación. En la terminología adoptada en computación evolutiva, la cadena binaria que codifica a un conjunto de soluciones se le llama "cromosoma". A cada segmento de la cadena que codifica una variable se le llama "gene" y al valor de cada posición cromosómica se le llama "alelo" (puede ser "0" ó "1").

2.3. Ventajas de los algoritmos evolutivos

Las principales ventajas que presenta el uso de los algoritmos evolutivos en la resolución de problemas de optimización [18] son entre otras, las siguientes:

- Operan sobre una población (o conjunto de soluciones) lo que evita que la búsqueda se quede atascada en óptimos locales.
- No requieren conocimiento previo sobre el problema a resolverse.
- Pueden combinarse con otras técnicas de búsqueda para mejorar su desempeño.
- Permiten su paralelización de forma sencilla.
- Son conceptualmente fáciles de implementar y usarse.

Capítulo 3

Optimización Evolutiva Multiobjetivo

En el mundo real, la mayoría de los problemas tienen más de un objetivo a optimizar, y normalmente éstos se encuentran en conflicto unos con otros, por lo que el mejorar un solo objetivo significa empeorar el desempeño de los otros. Muchos de estos problemas suelen convertirse a mono objetivo (sólo un objetivo) considerando a todos los demás objetivos como restricciones del problema principal, dado que esto simplifica su solución.

3.1. Optimización multiobjetivo

El problema de optimización multiobjetivo (POM) es similar al problema de optimización global, excepto por el hecho de que en el caso multiobjetivo se intenta encontrar un vector de solución que optimice simultáneamente todas las funciones objetivo, sabiendo de antemano que estas funciones se encuentran en conflicto unas con otras y el mejorar una función significa empeorar el desempeño de las otras.

3.1.1. Variables de decisión

Las variables de decisión son aquellos valores numéricos que se eligen para un problema de optimización. Estos valores pueden ser denotados como: x_j para $j = 1, 2, \ldots, n$. Así entonces, el vector de n variables de

decisión $\overrightarrow{x}$ es representado por:

$$\overrightarrow{x} = [x_1, x_2, \ldots, x_n]^T$$

3.1.2. Restricciones

En la mayoría de los POM existen restricciones emulando las características particulares del entorno o de los recursos disponibles. Estas restricciones están representadas de forma matemática como desigualdades:

$$g_i(\overrightarrow{x}) \leq 0; i = 1, \ldots, m$$

o igualdades:

$$h_i(\overrightarrow{x}) = 0; i = 1, \ldots, p$$

3.1.3. Problemas multiobjetivo

El problema de optimización multiobjetivo se puede definir matemáticamente como:

Encontrar el vector $\overrightarrow{x}^*$ que satisfaga las *m* restricciones de desigualdad:

$$g_i(\overrightarrow{x}) \leq 0; i = 1, \ldots, m$$

las p restricciones de igualdad:

$$h_i(\overrightarrow{x}) = 0; i = 1, \ldots, p$$

y optimice el vector de funciones objetivo:

$$f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_k(\overrightarrow{x})$$

En otras palabras, se intenta determinar el conjunto de todos aquellos números que satisfagan las restricciones y que optimicen todas las funciones objetivo.

Las restricciones son las que definen la región factible del problema y cualquier vector $\overrightarrow{x}$ que esté en esta región se considera como una *solución factible*.

Existen 3 tipos de problemas multiobjetivo:

- Minimizar todas las funciones objetivo

- Maximizar todas las funciones objetivo
- Minimizar algunas y maximizar las funciones objetivo restantes

Por simplicidad, se suelen considerar sólo problemas del primer tipo (minimizar todas las funciones objetivo), transformando las funciones objetivo, en caso de ser necesario.

3.1.4. Óptimo de Pareto

En optimización multiobjetivo, el término *optimizar* cambia con respecto a la optimización mono-objetivo (global), pues se trata de encontrar un compromiso entre las distintas funciones objetivo en vez de una sola solución como en optimización global. Así entonces, Vilfredo Pareto [33] dio, hacia finales del siglo XIX, una definición más formal de óptimo en problemas multiobjetivo, el cual se conoce en la actualidad como *óptimo de Pareto*.
La definición formal es la siguiente:

Un vector de variables de decisión $\overrightarrow{x}^* \in F$ (donde F es la zona factible) es un óptimo de Pareto si no existe otro $\overrightarrow{x} \in F$ tal que: $f_i(\overrightarrow{x}) \leq f_i(\overrightarrow{x}^*)$ para toda i = 1,..., k y $f_j(\overrightarrow{x}) < f_j(\overrightarrow{x}^*)$ para al menos una j.

En otras palabras, "El óptimo de Pareto es aquel vector de variables en el cual no se pueden mejorar las soluciones del problema en una función objetivo sin empeorar cualquiera de las demás" [1].

Desafortunadamente, esto no nos proporciona una sola solución, sino que obtenemos un conjunto de soluciones llamado *Conjunto de Óptimos de Pareto*. El conjunto de vectores que corresponden a una solución incluida en el conjunto de óptimos de Pareto son llamados *no-dominados*.

3.1.5. Dominancia de Pareto

El término *Dominancia de Pareto* puede ser definido de la siguiente manera:

Un vector $\overrightarrow{u} = (u_1, \ldots, u_k)$ domina a otro $\overrightarrow{v} = (v_1, \ldots, v_k)$ si y sólo si u es parcialmente menor a v

Es decir que para que una solución domine a otra, ésta necesita ser estrictamente mejor en al menos un objetivo, y no peor en ninguno de ellos. Esto es, al comparar dos soluciones A y B, sólo pueden existir tres posibles soluciones:

- A domina a B

- A es dominada por B
- A y B no se dominan (son no dominadas entre sí)

3.1.6. Frente de Pareto

La representación de las funciones objetivo cuyos vectores son no dominados y además están en el conjunto de óptimos de Pareto es llamado el *Frente de Pareto*.
La definición formal es la siguiente:

Para un problema multiobjetivo dado $\overrightarrow{f}(x)$ y un conjunto de óptimos de Pareto P^*, el frente de Pareto (FP^*) es:

$$FP^* := \{\overrightarrow{f} = [f_1(x), \ldots, f_k(x)] | x \in P^*\}$$

De forma general, no existe un método eficiente para encontrar el frente de Pareto, y la mejor forma de hacerlo es probar todos y cada uno de los puntos en la zona factible (es decir, enumerar todas las soluciones posibles con una precisión dada). Obviamente, en muchas ocasiones el espacio de busqueda es tan grande que un proceso enumerativo es incosteable (computacionalmente hablando) y de ahí la necesidad de usar heurísticas como los algoritmos evolutivos para producir aproximaciones del frente de Pareto de un problema.

3.2. Algoritmos para optimización multiobjetivo

Debido a que la mayoría de los métodos de programación matemática que existen para optimización multiobjetivo operan sobre un solo individuo a la vez, se necesitan ejecutar en varias ocasiones para poder encontrar el conjunto de óptimos de Pareto de un POM. En contraste, los algoritmos evolutivos tienen la ventaja de trabajar con una población (o conjunto de soluciones), lo que permite generar varias soluciones no dominadas en una sola ejecución.

Veremos ahora algunas de las técnicas evolutivas que se han propuesto para optimización multiobjetivo, dividiéndolos según la clasificación de técnicas multiobjetivo propuesta por Cohon y Marks [7].

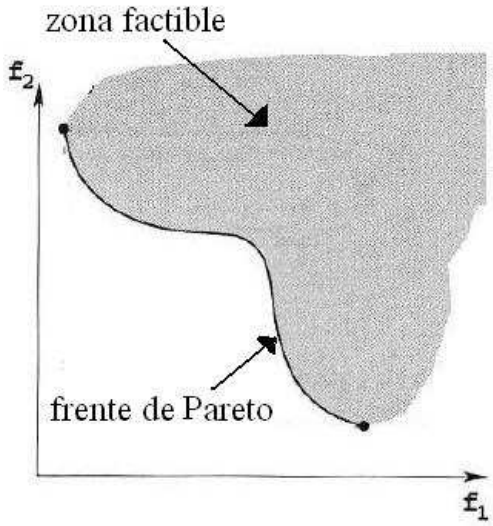

Figura 3.1: Gráfica de un frente de Pareto en un problema de optimización con dos funciones objetivo.

3.2.1. Técnicas *a priori*

Las preferencias del usuario tienen que ser conocidas antes de que comience la búsqueda. Algunos técnicas dentro de este grupo son:

- Orden lexicográfico.- se ordenan las funciones objetivo según su importancia y se va encontrando el óptimo comenzando por la más importante. El problema de esta técnica está en que se debe conocer el orden de importancia de las funciones objetivo y el resultado depende de dicho orden.
- Funciones de agregación lineal.- combinan los resultados de las distintas funciones objetivo en un único valor de aptitud. Este valor de aptitud se obtiene con una combinación lineal de las funciones objetivo. El problema es que se debe de introducir un factor de escalamiento para las funciones objetivo y en la práctica resulta difícil realizar una buena asignación de pesos para las funciones objetivo. Adicionalmente, este esquema no puede generar porciones cóncavas del frente de Pareto sin importar que pesos se usan.
- Funciones de agregación no lineales.- es una técnica que utiliza un

vector de metas para minimizar la diferencia entre los valores obtenidos. Esta técnica puede generar soluciones en las partes no convexas del frente de Pareto, pero desafortunadamente no son muy eficientes computacionalmente.

3.2.2. Técnicas progresivas

En estas técnicas, las preferencias se van dando conforme la búsqueda avanza y el tomador de decisiones indica si una solución le parece adecuada o no y el proceso actualiza las preferencias conforme el tomador de decisiones lo va indicando, guiando así el proceso de búsqueda.

Existen varias técnicas más en investigación de operaciones, pero aún no se han reportado versiones de ellas en computación evolutiva.

3.2.3. Tecnicas *a posteriori*

En estas técnicas, las preferencias se expresan al final y el tomador de decisiones recibe una información completa de los resultados para así entonces tomar la decisión que mejor le convenga. Es decir, los resultados intentan mostrar todos los compromisos posibles entre las funciones objetivo tratando de generar el verdadero frente de Pareto o al menos una aproximación razonablemente buena.

- Muestreo independiente.- consiste en hacer varias ejecuciones de la técnica para encontrar los puntos del frente de Pareto. En este caso, se utilizan diferentes valores de los pesos para las funciones objetivo para generar las diversas porciones del frente de Pareto. El problema de este tipo de enfoque es que se vuelve muy ineficiente para muchas funciones objetivo, pues crece exponencialmente al aumentar el número de combinaciones posibles.

- Selección por criterio.- se hacen sub poblaciones y cada una de éstas optimiza una sola función objetivo. Un ejemplo claro de este tipo es el algoritmo de Schaffer llamado VEGA (*Vector Evaluated Genetic Algorithm*) [36]. Un problema de este tipo de técnica es que no se obtienen necesariamente elementos del conjunto de óptimos de Pareto.

- Selección agregativa.- Se utilizan funciones agregativas para resolver los problemas, con la notable diferencia de que los pesos van cambiando conforme el número de generaciones va cambiando.
- Basados en Pareto.- La idea fue propuesta por Goldberg [1] y la idea básica es encontrar las soluciones no dominadas y asignar una aptitud con respecto a esta dominancia de Pareto. Estas técnicas, por la importancia que tienen, se discuten en mayor detalle en la sección siguiente.

3.3. Algoritmos basados en la jerarquización de Pareto

Goldberg [18] propuso utilizar la dominancia de Pareto como un criterio de selección para diseñar algoritmos evolutivos multiobjetivo. La idea principal es marcar a las soluciones que son no dominadas, asignarles una jerarquía más alta y retirarlos de la población. Este proceso continúa con las siguientes soluciones que son no dominadas y se les asigna otra jerarquía menos alta que la primemra. Así, este proceso se realiza hasta que se ha asignado una jerarquía a cada individuo de la población [9]. También se sugirió el uso de nichos para evitar la convergencia a un solo punto.

A continuación se muestran los algoritmos más representativos que utilizan selección basada en una jerarquización de Pareto.

3.3.1. MOGA

El Multi-Objective Genetic Algorithm (MOGA) fue propuesto por Fonseca y Fleming [16]. En este enfoque la jerarquía de un individuo depende del número de individuos en la población actual que lo dominan, es decir:

$$\text{jerarquia}(x_i, t) = 1 + p_i^{(t)}$$

donde $p_i^{(t)}$ es el número de individuos que dominan a x_i en la generación t. Así entonces los individuos no dominados tendrán una jerarquía de '1' . MOGA utiliza nichos para distribuir uniformemente las soluciones, además usa restricciones a la cruza y otros mecanismos cuya motivación es evitar una convergencia prematura.

Algoritmo 4 MOGA

Inicializar una población
Evaluar la aptitud de la población
Asignar una jerarquía basado en la Dominancia de Pareto
Realizar el cómputo de los nichos
Asignar una aptitud lineal escalable
Asignar una aptitud de compartición
para i = 1 hasta G

    Seleccionar con muestreo estocástico
    Cruza por un punto
    Mutación
    Evaluar la aptitud
    Asignar una jerarquía basado en la Dominancia de Pareto
    Realizar el cómputo de los nichos
    Asignar una aptitud lineal escalable
    Asignar una aptitud de compartición

terminar el ciclo

3.3.2. NSGA

El Non-Dominated Sorting Genetic Algorithm (NSGA) fue propuesto por Srinivas y Deb [38] y utiliza la idea original de Goldberg al jerarquizar la población por capas. Así entonces, los individuos que están en la primera capa tienen la máxima aptitud (por ser no dominados con respecto a toda la población) y se obtienen muchas copias de estos individuos con respecto al resto de la población. El NSGA usa también compartición de aptitud para distribuir los individuos a lo largo del frente de Pareto.

3.3.3. NSGA-II

El Non-Dominated Sorting Genetic Algorithm - II (NSGA - II) fue propuesto por Deb et al. [11]. Esta es una nueva versión del NSGA más eficiente (computacionalmente hablando) además de que se agrega elitismo y un operador que ayuda a mantener la diversidad sin requerir de parámetros adicionales.

3.3.4. PAES

La Pareto Archieved Evolution Strategy (PAES) fue propuesta por Knowles y Corne [8]. PAES consiste de una estrategia evolutiva de dos miembros (en la que un padre genera un solo hijo) en combinación con un archivo externo el cual almacena los individuos no dominados generados a lo largo del proceso evolutivo. Así, cuando se encuentra un individuo no dominado, se compara con los individuos del archivo externo, y en caso de que sea no dominado con respecto a éstos, una malla adaptativa se encarga de elegir a los individuos que salen del archivo externo. La malla ayuda a mantener diversidad y a distribuir uniformemente las soluciones no dominadas producidas. Además no requiere parámetros y su complejidad computacional es menor que la de los nichos.

3.3.5. SPEA

El Strength Pareto Evolutionary Algorithm (SPEA) fue propuesto por Zitzler y Thiele [45]. Este algoritmo hace uso de un archivo externo que contiene las soluciones no dominadas del problema. A cada generación, los individuos no dominados se copian al archivo externo y se les asigna un valor

Algoritmo 5 NSGA-II

Inicializar una población

Generar aleatoriamente una población

Evaluar la aptitud

Asignar un nivel basado en la dominancia de Pareto - "ordenar"

Generar la población siguiente:

Selección mediante un torneo binario

Recombinación y mutación

Para i = 1 hasta un número de generaciones

Para la población padre e hijo

- Asignar un nivel basado en la dominancia de Pareto y ordenar
- Generar el conjunto de frentes no dominados
- Sumar soluciones a la siguiente generación, empezando por la primera jerarquía y utilizar el factor de agrupamiento (crowding) en cada frente

Seleccionar los puntos en el frente más bajo y que estén fuera de la distancia del factor de agrupamiento

Crear la siguiente generación

Selección mediante un torneo binario

Recombinación y mutación

terminar el ciclo

Algoritmo 6 PAES

Repetir

Iniciar una población y agregarla al archivo externo

Mutar un padre p para producir un hijo h y evaluar su aptitud

Si (p domina a h) descartar h

de lo contrario, si (h domina a p)

Reemplazar p con h, y agregar h al archivo externo

de lo contrario, si (h es dominado por cualquiera del archivo externo)

Descartar h

de lo contrario, aplicar la prueba (p, h, archivo) para determinar si:

Se vuelve parte de la solución y saber dónde se agrega h al archivo.

mientras no se cumpla la condición de paro

extra el cual es proporcional al número de individuos que lo dominan. De tal forma, la aptitud de cada individuo se calcula de acuerdo con este valor extra de todos los individuos del archivo externo que lo dominen. Además, se adopta una técnica de cúmulos para mantener diversidad.

3.3.6. SPEA-2

El Strength Pareto Evolutionary Algorithm 2 (SPEA 2) fue propuesto por Zitzler et al. [44]. Este algoritmo tiene tres diferencias principales con respecto a su predecesor: para calcular la aptitud de los individuos se toman en cuenta la cantidad de individuos que lo dominan, y la cantidad de individuos que son dominados por él. También utiliza una densidad vecinal que guía la búsqueda más eficientemente. Finalmente usa un esquema del truncamiento del archivo externo para garantizar la preservación de las soluciones en los extremos.

Algoritmo 7 SPEA2

Inicializar una población P

Crear un archivo externo vacio E

Para i = 1 hasta G

Evaluar la aptitud de cada individuo en P y E

Copiar todos los individuos no dominados de P y E a E

Usar el operador de truncamiento para quitar los elementos de E cuando el tamaño del archivo sea muy grande

Si el tamaño del archivo no se ha excedido entonces usar los individuos dominados en P para llenar E

Desarrollar una selección con torneo binario con reemplazo para generar la siguiente generación

Aplicar cruza y mutación a los seleccionados

terminar el ciclo

3.4. Medidas de desempeño para algoritmos evolutivos multiobjetivo

Para realizar una evaluación cuantitativa de los resultados obtenidos por los diferentes algoritmos, se han ideado métricas las cuales nos ayudan a determinar la eficiencia con la que se resuelven los problemas multiobjetivo. Los aspectos que suelen ser medidos por las métricas son la cercanía de los individuos no dominados generados con respecto al verdadero frente, así como la distribución de los individuos no dominados a lo largo del frente. Algunas métricas son:

3.4.1. Tasa de error (TE)

Indica el porcentaje de individuos que están sobre el verdadero frente de Pareto. Matemáticamente se representa como:

$$TE = \frac{\sum_{i=1}^{n} e_i}{n}$$

donde n es el número de soluciones generadas por el algoritmo y

$$e_i = \begin{cases} 0 & \text{si el vector } i, \forall\, i = (1, \ldots, n) \text{ está en el frente de Pareto verdadero,} \\ 1 & \text{en otro caso} \end{cases}$$

Un valor de TE = 0, significa que todos los puntos obtenidos están en el verdadero frente de Pareto; y TE = 1 significa que ninguno lo está.

3.4.2. Distancia generacional (DG)

Indica qué tan lejos se encuentran las soluciones generadas por el algoritmo del verdadero frente de Pareto. Matemáticamente se representa como:

$$DG = \frac{(\sum_{i=1}^{n} d_i^p)^{\frac{1}{p}}}{n}$$

donde n es el número de soluciones generadas por el algoritmo, p = 2, y d_i es la distancia euclidiana entre cada vector y el individuo más cercano al verdadero frente de Pareto. Un resultado de 0 indica que se alcanzó el verdadero frente de Pareto y cualquier otro valor nos indica qué tan lejos se encuentra del mismo.

3.4.3. Distribución (D)

Indica qué tan bien distribuidos se encuentran los individuos no dominados generados por el algoritmo. Matemáticamente se representa como:

$$D = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\bar{d} - d_i\right)^2}$$

donde n es el número de soluciones generadas como resultado del algoritmo, y $d_i = min_j(|f_1^i(\vec{x}) - f_1^j(\vec{x})| + |f_2^i(\vec{x}) - f_2^j(\vec{x})|)$, $i, j = 1, \ldots, n$.

Un valor de 0 en esta métrica indica que todos los miembros del frente de Pareto generado están equidistantes.

3.4.4. Cobertura (C)

Indica si dos conjuntos de vectores de solución están juntos o no, o si alguno de ellos domina en su mayoría al otro o viceversa. Se define de la siguiente manera:

$$C(X', X'') = \frac{|\{a'' \in X''; \exists a' \in X' : a \succeq a''\}|}{|X''|}$$

donde $X', X'' \subseteq X'$ son dos conjuntos de vectores de solución. Si todos los puntos en X' dominan o son iguales a todos los puntos en X'', entonces $C = 1$. $C = 0$ implica lo contrario. Obsérvese que ésta no es una métrica de distancia para saber qué tan cerca están los conjuntos unos del otro.

Capítulo 4

Evolución Diferencial

La evolución diferencial es una rama de la computación evolutiva desarrollada por Rainer Storn y Kenneth Price [39, 40] para optimización en espacios continuos.

En la evolución diferencial (ED), las variables se representan mediante números reales. La población inicial se genera de forma aleatoria y se seleccionan tres individuos como padres. Uno de los padres es el *padre principal* y éste se perturba con el vector de los otros dos padres. Si el valor resultante es mejor que el elegido para reemplazo, entonces lo reemplaza. De otra forma, se retiene al padre principal.

4.1. Historia de la evolución diferencial

La evolución diferencial nació en 1994 en los intentos de Kenneth Price por resolver el polinomio de Chebychev, problema propuesto por Rainer Storn. En estos intentos fue cuando Kenneth Price concibió la idea de utilizar las diferencias entre los vectores para perturbar el vector de la población. Con esta idea principal, Storn y Price comenzaron a realizar simulaciones y pruebas para poder construir un método versátil y robusto como lo es hoy en día esta heurística. La comunidad de ED ha estado creciendo desde 1995 y cada día más investigadores trabajan con esta técnica. Es un deseo de estos investigadores que la ED se siga utilizando y estudiando por más personas alrededor del mundo. Por esta razón es que la ED no ha sido patentada de ninguna forma [34].

4.2. Algoritmo ED para optimización global

La ED es un método de búsqueda que utiliza N vectores:

$$x_{i,G}; i = 0, 1, 2, \ldots, N-1$$

como la población de cada generación (G). El valor de N no cambia durante el proceso de minimización. La población inicial se elige de manera aleatoria si no se conoce nada acerca del problema. Como regla, se asume una distribución uniforme para las decisiones aleatorias a tomar. La idea principal detrás de la ED es un nuevo esquema para generar vectores. La ED genera estos nuevos vectores cuando se suma la diferencia de pesos entre dos vectores miembros de la población a un tercer vector miembro. Si la aptitud del vector resultante es menor que el miembro de la población elegido entonces el nuevo vector reemplaza al vector con el cual fue comparado. Este vector a comparar puede ser (aunque no necesariamente lo es) parte del proceso de generación arriba mencionado. Además, el mejor vector $X_{mejor,G}$ se evalúa en cada generación G para no perder de vista el progreso durante el cual se hace la minimización.

Existen diferentes variantes en ED, las dos más prometedoras son las siguientes:

4.2.1. Esquema ED1

Para cada vector $\overrightarrow{x_{i,G}}; i = 0, 1, 2, \ldots, N-1.$, un vector de prueba $\overrightarrow{v}$ se genera de acuerdo a:

$$\overrightarrow{v} = \overrightarrow{x_{r1,G}} + F \cdot (\overrightarrow{x_{r2,G}} - \overrightarrow{x_{r3,G}})$$

con $r_1, r_2, r_3 \in [0, N-1]$, enteros y diferentes, y $F > 0$.

Los enteros $r_1,\ r_2$ y r_3 son elegidos aleatoriamente en el intervalo [0, N-1] y son diferentes entre sí. F es un factor real y constante que controla la amplificación en la variación diferencial $(\overrightarrow{x_{r2,G}} - \overrightarrow{x_{r3,G}})$. La figura 4.1 muestra un ejemplo bidimensional que ilustra los diferentes vectores que toman parte en ED1.

Para incrementar la diversidad de los resultados en los vectores, se usa el vector:

$$\overrightarrow{u} = (u_1), u_2, \ldots, u_D^T$$

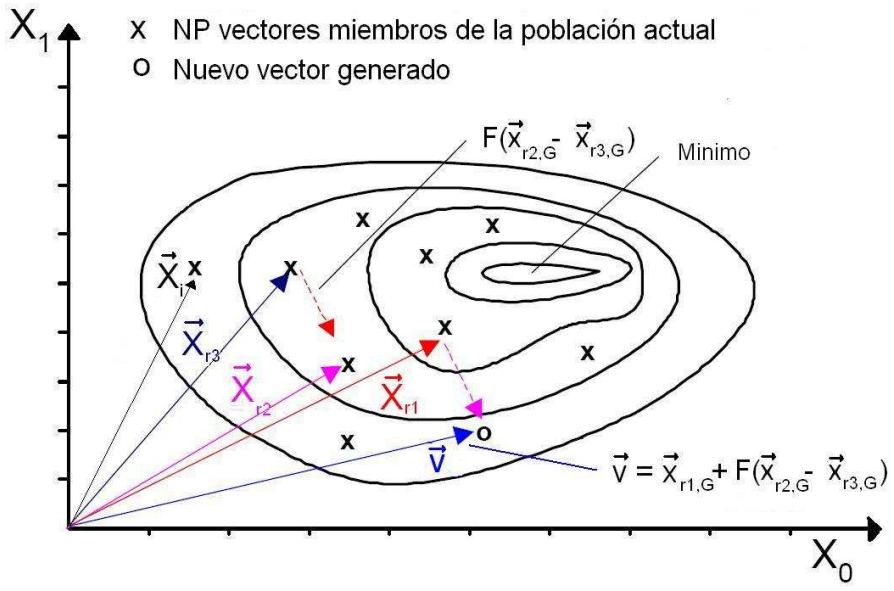

Figura 4.1: Ejemplo bidimensional de una función objetivo mostrando el proceso de generación en el esquema ED1.

con:

$$u_j = \begin{cases} v_j, & \text{para } j = \langle n \rangle_D, \langle n+1 \rangle_D, \ldots, \langle n+L-1 \rangle_D; \\ (x_{i,G})_j, & \text{de otra forma.} \end{cases}$$

donde los paréntesis $\langle\rangle_D$ denotan la función módulo con módulo D. Veamos un ejemplo:

Una cierta secuencia de elementos del vector en $\overrightarrow{u}$ son elementos idénticos a $\overrightarrow{v}$, los demás elementos de $\overrightarrow{u}$ adquieren el valor original de $\overrightarrow{x_{i,G}}$. Esta idea está ilustrada en la figura 4.2 para D = 7, n = 2 y L = 3. El índice *n* es elegido de forma aleatoria en el intervalo [0, D - 1]. El entero L está en el intervalo [0, D - 1] con la probabilidad $Pr(L = v) = (CR)^v.CR \in [0, 1]$ es la probabilidad de cruza y constituye una variable de control para el esquema DE1. Las decisiones aleatorias para ambos *n* y L se recalculan para cada nuevo vector de prueba $\overrightarrow{v}$.

Y para decidir si el nuevo vector $\overrightarrow{u}$ se vuelve miembro de la población en la generación G+1, se compara con el vector $\overrightarrow{x_{i,G}}$, y si tiene un menor valor al evaluarla en la función objetivo entonces $\overrightarrow{u}$ se hace $\overrightarrow{x_{i,G+1}}$; de otra forma el valor de $\overrightarrow{x_{i,G}}$ es retenido.

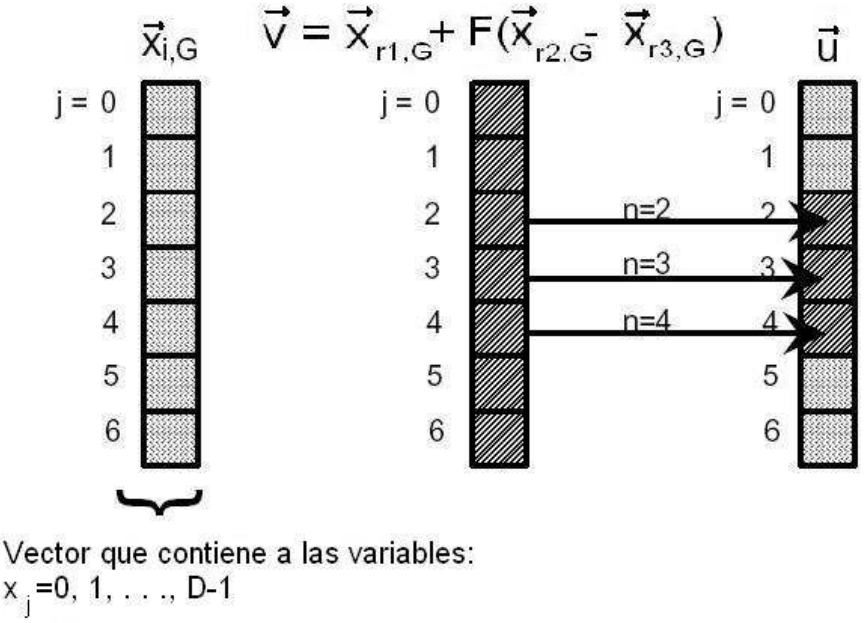

Figura 4.2: Ilustración del proceso de cruza para D = 7, n = 2 y L = 3

4.2.2. Esquema ED2

El esquema ED2 trabaja de igual forma que el ED1 pero genera el vector $\overrightarrow{v}$ de acuerdo a:

$$\overrightarrow{v} = \overrightarrow{x_{i,G}} + \lambda \cdot (\overrightarrow{x_{mejor,G}} - \overrightarrow{x_{i,G}}) + F \cdot (\overrightarrow{x_{r2,G}} - \overrightarrow{x_{r3,G}}),$$

introduciendo una variable de control adicional λ. La idea es incorporar al esquema al mejor vector de la población $\overrightarrow{x}_{mejor,G}$. Esta característica puede ser muy útil para las funciones objetivo no-críticas. La figura 4.3 ilustra el proceso de generación del vector. La construcción de $\overrightarrow{u}$ se hace usando $\overrightarrow{v}$ y $\overrightarrow{x_{i,G}}$, aunque el proceso de decisión es idéntico al ED1.

4.3. Diferentes estrategias de ED

Las diferentes estrategias que pueden adoptarse en el algoritmo de ED [34] dependen del tipo de problema al que se aplique. Las estrategias se basan en el vector a perturbar, el número de diferentes vectores considerados

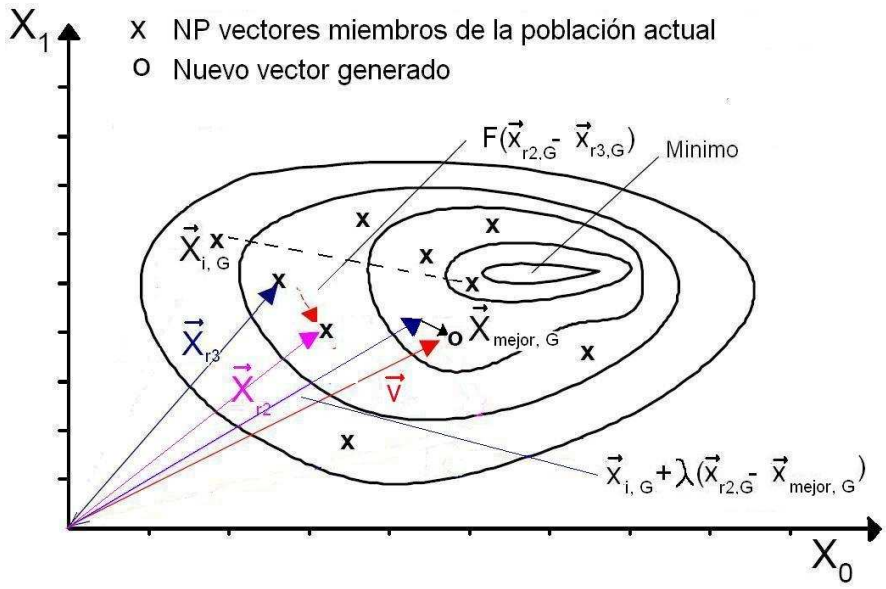

Figura 4.3: Ejemplo bidimensional de una función objetivo mostrando el proceso de generación en el esquema ED2

para la perturbación, y finalmente el tipo de cruza a utilizar. Los siguientes son las diez estrategias diferentes propuestas por Price & Storn [34].

1. ED/mejor/1/exp
2. ED/aleatorio/1/exp
3. ED/aleatorio-al-mejor/1/exp
4. ED/mejor/2/exp
5. ED/aleatorio/2/exp
6. ED/mejor/1/bin
7. ED/aleatorio/1/bin
8. ED/aleatorio-al-mejor/1/bin
9. ED/mejor/2/bin
10. ED/aleatorio/2/bin

La convención utilizada anteriormente es: *ED/x/y/z*.

- **ED** se refiere obviamente a la Evolución Diferencial
- **x** representa el vector a perturbar; puede ser el mejor vector de la generación anterior (mejor) o cualquier vector elegido aleatoriamente (aleatorio).
- **y** es el número de vectores considerados para la perturbación de **x**; Para la perturbación de un simple vector, se eligen aleatoriamente 3 diferentes vectores, y la diferencia de dos de ellos se suma al tercero. Para la perturbación de dos vectores diferentes, se eligen 5 vectores distintos; la diferencia de pesos para cada par de cualquiera de los cuatro primeros vectores se suma al quinto.
- **z** se entiende como el tipo de cruza a ser utilizada (exp: exponencial; bin: binomial). La cruza en la ED se realiza en **d** variables del vector dentro de un ciclo limitado por el valor de *CR*. En la cruza exponencial, la primera vez en la que un número aleatorio entre (0,1) supera el valor de CR se suspende este ciclo y las variables que queden por alterar quedan intactas. En la cruza binomial, la cruza se realiza en cada una de las **d** variables siempre y cuando al elegir un número aleatorio entre (0,1) éste sea menor que el valor de *CR*. Así que para valores altos de CR, la cruza exponencial y la binomial se comportan de forma similar.

La estrategia a adoptar para cada problema se determina de forma independiente *mediante ensayo*. De tal forma, una estrategia que funcione mejor para resolver un problema puede no trabajar bien cuando se aplica para otro problema distinto. Así tambien, los parámetros que se adoptan para un problema tienen que ser determinados mediante ensayo. Sin embargo, la estrategia (ED/aleatorio/1/bin) es la más comúnmente utilizada y es la que se adoptó en este trabajo. En la figura 4.4 se muestra un ejemplo en el que se ilustra cómo generar un nuevo vector en la población, y cómo determinar si pasa a la siguiente generación.

La explicación es la siguiente:

1 Se escoge un vector de forma aleatoria dentro de la población, el cual tiene una aptitud de *94*.

2 Se eligen dos vectores de forma aleatoria, uno tiene la aptitud de *5* y otro de *62*.

3 Se calcula la diferencia en sus pesos, esto en el espacio de las variables de los vectores, obteniéndose un nuevo vector con una aptitud distinta.

4 Un tercer vector es elegido aleatoriamente de la población. En este caso, tiene una aptitud de *77* y se realiza una suma de éstos en el espacio de las variables.

5 Este vector de prueba obtenido del paso 4 se cruza con el vector original, para obtener así un nuevo individuo que será el hijo de la cruza. Este nuevo vector tiene una aptitud de *44*.

6 Por último, se realiza una comparación entre el vector padre y el hijo, para determinar cuál es el que pasa a la siguiente generación. En este caso, el hijo (44) $<$ padre (94) y por lo tanto, el nuevo vector generado se copia a la nueva generación.

4.4. Trabajo previo en optimización multiobjetivo utilizando ED

Existen pocos trabajos que utilizan la Evolución Diferencial para resolver problemas multiobjetivo. A continuación se revisan las propuestas más representativas.

4.4.1. PDE

El Pareto Differential Evolution (PDE) fue desarrollado por H. A. Abbass y R. Sarker en 2002 [2] y es una adpatación del algoritmo de ED descrito con anterioridad con las siguientes modificaciones:

1. La población inicial se genera con una distribución gaussiana N(0.5, 0.15).
2. El parámetro F, se genera con una distribución Gaussiana N(0, 1).
3. La reproducción se realiza sólo con las soluciones no dominadas en cada generación.

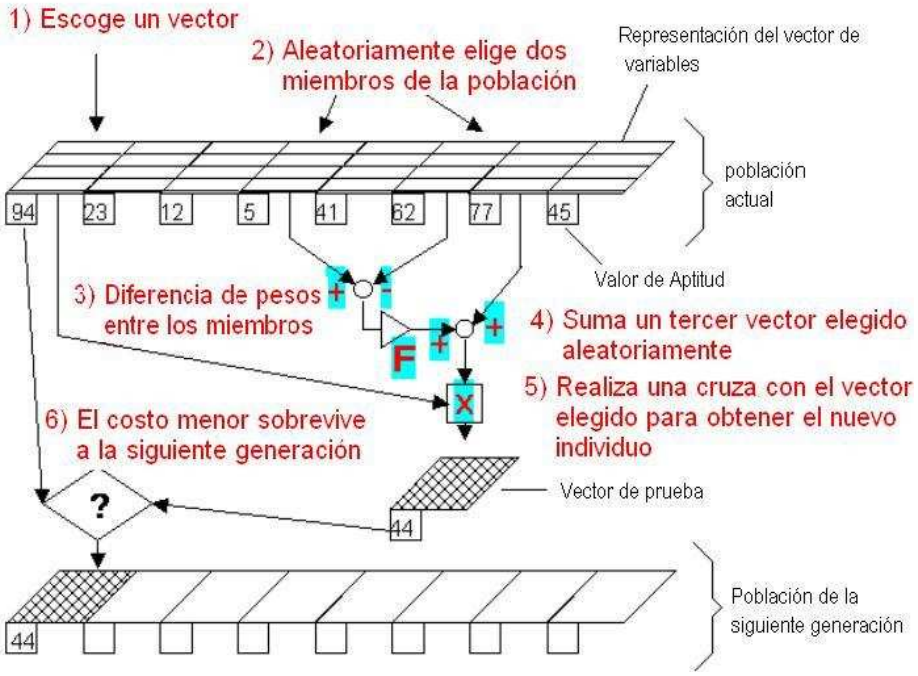

Figura 4.4: Ejemplo de la generación de los vectores utilizando la Evolución Diferencial con el esquema (DE/aleatorio/1/bin)

4. Los límites en las variables se preservan cambiando el signo de la variable si es menor a 0 o restando 1 si es mayor a 1, hasta que la variable esté en los límites permitidos.

5. El individuo generado se coloca en la población si domina a su padre.

El algoritmo se puede resumir de la siguiente manera: Se genera una población inicial, y todas las soluciones dominadas se quitan de la población y los que queden se utilizan para la reproducción. Tres padres se seleccionan de forma aleatoria y el hijo se genera de los tres padres y se coloca en la población si domina al padre principal; de otra forma una nueva selección se lleva a cabo y este individuo se olvida. Este proceso continúa hasta que la población se completa. Una versión genérica del algoritmo se muestra en el algoritmo 8.

Un número máximo de soluciones no dominadas en cada generación puede ser de 50. Si el máximo es excedido, entonces se adopta la siguiente función:

$$D(x) = \frac{(min\|x - x^i\| + min\|x - x^j\|)}{2}$$

donde $x \neq x^i \neq x^j$. Esto es, la *distancia vecinal* es la media euclidiana entre los dos puntos más cercanos. Las soluciones no dominadas con la distancia menor son removidos de la población hasta que el número de soluciones sea de 50.

4.4.2. MODE

El Pareto-Based multi-objective Differential Evolution (MODE) fue desarrollado por Feng Xue, Arthur C. Sanderson y R. Graves [43]. En este algoritmo se intenta simular la ED con los cambios pertinentes de la siguiente manera, tanto en la cruza, como en la evaluación de Pareto y en la selección.

Operador de cruza utilizando el esquema ED2

Como ya se ha visto con anterioridad, en el esquema ED2 se definen dos clases de vectores, el vector diferencial $\lambda \cdot (\overrightarrow{x_{mejor,G}} - \overrightarrow{x_{i,G}})$ y los vectores perturbadores. $F \cdot (\overrightarrow{x_{r2,G}} - \overrightarrow{x_{r3,G}})$,

$$\overrightarrow{v} = \overrightarrow{x_{i,G}} + \lambda \cdot (\overrightarrow{x_{mejor,G}} - \overrightarrow{x_{i,G}}) + F \cdot (\overrightarrow{x_{r2,G}} - \overrightarrow{x_{r3,G}}),$$

Algoritmo 8 PDE

1. Generar aleatoriamente una población inicial con distribución Gaussiana N (0.5, 0.15)

2. Repetir

a. Evaluar los individuos de la población y marcar los no dominados

b. Si el número de no dominados en la población es menor a tres repetir lo siguiente hasta que el número de no dominados sea mayor o igual a tres

i. encontrar una solución no dominada de los que no están marcados
ii. Marcar esta solución como no dominada

c. Si el número de individuos no dominados es mayor que el permitido, aplicar la 'distancia vecinal' hasta que el número de no dominados sea menor al permitido.

d. Borrar las soluciones dominadas de la población.

e. Repetir

i. Seleccionar aleatoriamente un indiviuo como padre principal α_1 y dos individuos α_2, α_3 como padres de soporte.
ii. Seleccionar una variable *j* aleatoriamente
iii. Para cada variable *i*

Con una probabilidad Uniforme (0, 1) o si i = j, hacer:

$$x_i^{hijo} \leftarrow x_i^{\alpha 1} + F \cdot (x_i^{\alpha 2} - x_i^{\alpha 3})$$

de otra forma:

$$x_i^{hijo} \leftarrow x_i^{\alpha 1}$$

donde cada variable *i* del padre principal, $x_i^{\alpha 1}$, es modificado sumándole la diferencia de las variables de los demás padres.

iv. Si el hijo domina al padre principal, lo reemplaza en la población.

f. Hasta que termine la población de tamaño M.

3. mientras la condición de terminación no se satisfaga, regresar al paso 2.

En la ED mono-objetivo, el vector diferencial es definido como el mejor vector de la población $\overrightarrow{x}_{mejor,G}$, y el individuo que está bajo la operación $\overrightarrow{x}_{i,G}$. El mejor individuo es usualmente el de la más alta aptitud. Sin embargo, en optimización multiobjetivo el propósito del algoritmo es encontrar un conjunto de soluciones (conjunto de óptimos de Pareto). En esta propuesta, se elige $\overrightarrow{x}_{i,G}$ de forma aleatoria, y debemos examinar si es un individuo dominado o no. Si es dominado, entonces el vector $\overrightarrow{x}_{mejor,G}$ se escoge aleatoriamente del conjunto de vectores que dominan a $\overrightarrow{x}_{i,G}$. Y el vector diferencial es la diferencia entre $(\overrightarrow{x}_{mejor,G} - \overrightarrow{x}_{i,G})$. Pero si $\overrightarrow{x}_{i,G}$ es no dominado, el vector diferencial se vuelve 0, por lo que sólo los vectores perturbadores causan efecto.

Los vectores perturbadores se definen aleatoriamente de los padres de la población.

Así entonces, el operador de cruza que se formula para optimización multiobjetivo es el que se muestra en la ecuación 4.1, aplicándose a cada alelo del individuo bajo cierta probabilidad de cruza (p_m):

$$p'_i = \begin{cases} p_i + F \cdot \sum(p_{i_a^k} - p_{i_b^k}), & \text{si } p_i \text{ es no-dominado;} \\ \gamma \cdot p_{mejor} + (1-\gamma)p_i + F \cdot \sum(p_{i_a^k} - p_{i_b^k}), & \text{de lo contrario.} \end{cases} \tag{4.1}$$

donde p' es el hijo, p_{mejor} es el mejor individuo elegido de la población padre, $\gamma \in [0, 1]$ representa el operador adicional, y K es el número de vectores de perturbación, F es el factor de perturbación, $p_{i_a^k}$ y $p_{i_b^k}$ son seleccionados aleatoriamente y son mutuamente distintos de la población padre.

Evaluación basada en Pareto

En este algoritmo, se utiliza la jeraquización de Pareto [18] para evaluar a los individuos. La jerarquía de 1 se asigna a las soluciones no dominadas

y se les marca y retira de la población. Un nuevo conjunto de soluciones no dominadas se jerarquizan con 2 y son removidas de la población. Se procede así sucesivamente hasta que todos los individuos queden marcados.

Operador de selección

En este algoritmo (MODE), se aplica una estrategia de selección $(\mu+\lambda)$, en la que ambos padres e hijos compiten para entrar a la siguiente generación, es decir, los individuos primero se comparan utilizando su jerarquización de Pareto. Los individuos con jerarquía más alta se seleccionan para la siguiente generación. Si la jerarquía es la misma, entonces se utiliza la métrica de agrupamiento (crowding) para elegir a uno de éstos.

Se utiliza la jerarquización de Pareto igual que en el NSGA-II, pero en vez de utilizar el operador de agrupamiento del NSGA-II, se le sustituye con otro parámetro extra σ_{crowd} para especificar que tan rodeada está una solución en el espacio de las funciones objetivo y así reducir su aptitud. Esta estrategia previene la entrada de elementos similares a la siguiente generación (evitando la convergencia prematura). La diferencia es que el operador de agrupamiento (utilizado en el NSGA-II) no necesita especificarse, en cambio este parámetro sí, y se sugiere un valor pequeño de σ_{crowd} para los problemas utilizados.

4.5. Algoritmo que maneja las funciones objetivo como restricciones

Existe también otro algoritmo que utliza la ED para resolver problemas multiobjetivo, desarrollado por B.V. Babu y Mathew Leenus Jehan [3].

Este algoritmo elige una función objetivo como la principal, y a las demás las considera como restricciones de la primera, utilizando un esquema de penalización cuando se violan estas restricciones, para tener así una sola aptitud y poder realizar la comparación directa entre los individuos. También utiliza el algoritmo de la ED de igual forma como si fuera para mono-objetivo, basándose en la estrategia DE/rand/1/bin para la generación de individuos.

Capítulo 5

Propuesta para usar ED en problemas multiobjetivo

La técnica propuesta en esta tesis consiste en un algoritmo que se basa en la cruza de Evolución Diferencial para resolver problemas de optimización multiobjetivo. Se hace notar que se desarrollaron dos distintas versiones de la técnica, una en la cual se utiliza el mecanismo de $rejilla$ (MyDE) y otra donde se usa dominancia-ϵ ($\epsilon - MyDE$) en la población secundaria. En el algoritmo 9 se muestra el pseudo código del esquema que proponemos para resolver problemas multiobjetivo usando evolución diferencial. Se hace notar que MyDE corresponde exactamente con el algoritmo 9 y que los cambios correspondientes a $\epsilon - MyDE$ son aplicables sólo a la población secundaria, por lo que el pseudo código del Algoritmo 9 permanece válido en este caso también.

5.1. Algoritmo de MyDE

En un principio se genera una población inicial aleatoria de tamaño P. Cada variable se genera mediante una distribución uniforme dentro de los intervalos de las variables de cada problema. Una forma de incluir el elitismo en los algoritmos evolutivos para optimización multiobjetivo es la de utilizar un archivo externo (o población secundaria) en donde se almacenan los individuos no dominados encontrados a lo largo del proceso evolutivo. El objetivo de esta población secundaria es conservar aquellas soluciones no dominadas encontradas a lo largo del proceso evolutivo ya

que, de no retenerse, podrían ser destruidas por los operadores evolutivos. En consecuencia, esto asegura que se almacenen las mejores soluciones encontradas de forma global (es decir, a lo largo de todo el proceso evolutivo). Al final del proceso evolutivo, los individuos que estén contenidos en el archivo externo serán el resultado que se presenta al tomador de decisiones (el llamado frente de Pareto producido por el algoritmo). Este archivo externo tiene un máximo de NP soluciones que son las que el tomador de decisiones propone obtener como salida del algoritmo.

Así entonces, asumiendo que todas las variables están dentro de los límites permisibles, y la población inicial está ya generada, se seleccionan 3 padres de forma aleatoria y un hijo es generado de estos tres padres; este hijo se compara con el padre y lo reemplaza si lo domina; de otra forma el padre es el que trasciende a la siguiente generación. Todas las soluciones no dominadas se marcan y se vuelven candidatas para entrar en el archivo externo. Una vez que son o no aceptadas en el archivo externo, éstas se convierten en padres de la siguiente generación hasta que el proceso finalice. Cada uno de los pasos del algoritmo 9 se explican a continuación.

5.1.1. Codificación

El algoritmo de la evolución diferencial propuesto por Storn y Price [40] utiliza una codificación real en la que cada variable del problema se representa en el cromosoma como un número real. Adoptando esta misma codificación, en esta propuesta se usa la codificación real en la representación de cada cromosoma.

5.1.2. Generación de la población inicial

La población inicial se genera de forma aleatoria con una distribución uniforme $U(0,1)$, garantizando que las variables del problema se encuentren dentro de sus límites permitidos. Esto se realiza de la siguiente manera:

$$x_i = \text{LI}_i + U(0,1) \cdot (\text{LS}_i - \text{LI}_i)$$

donde:
$j = 0, 1, 2, \ldots, v-1$. (v = número de variables de cada vector solución)
LI_j límite inferior para la variable i
LS_j límite superior para la variable i

Algoritmo 9 Técnicas propuestas: MyDE y ϵ - MyDE

Generar una población inicial de tamaño P

Evaluar la aptitud de la población inicial

Repetir

Repetir

Seleccionar 3 padres
Realizar la cruza utilizando la Evolución Diferencial
Realizar mutación uniforme
Evaluar la aptitud del hijo generado
Si el hijo es mejor que el padre, lo reemplaza en la población

hasta que se genere la siguiente población

Marcar a los individuos no dominados en la población

Agregar los no dominados al archivo externo en caso de que sean aceptados

mientras no se cumpla la condición de finalización

5.1.3. Selección

La selección se realiza de 2 maneras distintas, dependiendo del número de generaciones que hayan transcurrido y del parámetro sel_2. Este parámetro ejerce una presión de selección más alta ayudando a que se generen mejores hijos y, como consecuencia, contribuye a obtener mejores soluciones del problema. Esto sigue lo indicado por Deb [9] en el sentido de que la cercanía de los padres mejora mucho la explotación en los algoritmos evolutivos.

El parámetro sel_2 tiene un intervalo (0.2 - 1) (pues cuando el proceso evolutivo inicia, el archivo externo está vacío), el cual indica el número de generaciones que tienen que pasar hasta que se ejerza esta nueva presión de selección. Por ejemplo: si sel_2 = 0.6 y el número de generaciones $G_{max} = 200$, esto significa que durante las primeras 120 generaciones (o sea, el 60 % de G_{max}) se realiza una selección aleatoria y durante las últimas 80 generaciones se realiza una selección elitista. Esto es:

$$\text{Tipo de Selección} = \begin{cases} \text{Selección Aleatoria}, & \text{si } gen < (sel_2 * G_{max}); \\ \text{Selección Elitista}, & \text{de otra forma.} \end{cases}$$

donde:
gen.- es el número de generaciones
G_{max}.- es el número total de generaciones

En ambas selecciones, aleatoria y elitista, siempre se debe elegir un padre como referencia, el cual nos sirve para comparar el hijo generado por los tres padres con el primero. De tal forma, se busca que el padre de referencia sea aquél que coincida con el índice del hijo que se está generando. Es decir, si es el primer hijo que se genera de la nueva población, entonces se deberá tomar al padre con índice p_0 de la población y así sucesivamente hasta generar la siguiente población. Este mecanismo garantiza que todos los padres de la población primaria serán padres de referencia en una sola ocasión.

Los dos tipos de selección se explican a continuación:

1. **Selección Aleatoria.-** Se seleccionan 3 padres de la población primaria de forma aleatoria, siempre y cuando ninguno de los padres sean iguales.

2. **Selección Elitista.-** Se seleccionan 3 padres de la población secundaria de forma aleatoria, siempre y cuando ninguno de los padres sean iguales y cumplan con un factor de cercanía entre ellos $f_{cercania}$ el cual se ilustra en la figura 5.1. Si no existe ningún padre que cumpla con esta condición entonces se elige a cualesquiera. Este factor está dado por:

$$f_{cercania} = \frac{\sqrt{\sum_{i=0}^{FUN} (X_{i,max} - X_{i,min})^2}}{2^{FUN}}$$

donde:
FUN = número de funciones objetivo
$X_{i,max}$ = valor máximo de la i-ésima función objetivo del archivo externo
$X_{i,min}$ = valor mínimo de la i-ésima función objetivo del archivo externo
Es importante mencionar que este factor $f_{cercania}$ sólo se calcula un máximo de dos ocasiones: una cuando la malla se llena, y, en caso de que no se haya llenado, entonces se calcula con aquellos individuos que estén dentro del archivo externo en ese momento. Una vez que la malla se llene, se recalcula este factor por segunda ocasión. Sabemos que el calcularlo resulta costoso computacionalmente hablando, ya que se tienen que realizar comparaciones directas entre todos los individuos del archivo externo, y de ahí que sea importante no repetir muchas veces este cálculo.

En la figura 5.2 se ilustra gráficamente (en un problema con 2 funciones objetivo) la diferencia entre la selección aleatoria y la elitista cuando ya tenemos al primer padre seleccionado y se busca encontrar a los 2 restantes. Los individuos dentro del círculo son los candidatos a padres.

5.1.4. Cruza utilizando la ED

En el algoritmo propuesto, se utiliza el esquema ED_1 de la Evolución Diferencial detallado en el capítulo anterior (sección 3.2), utilizando la estrategia (ED/aleatorio/1/bin) también detallada anteriormente (sección 3.3). Esto es, que para cada padre (vector) $\overrightarrow{p_i}$; $i = 0, 1, 2, \ldots, P-1$(P = pobla-

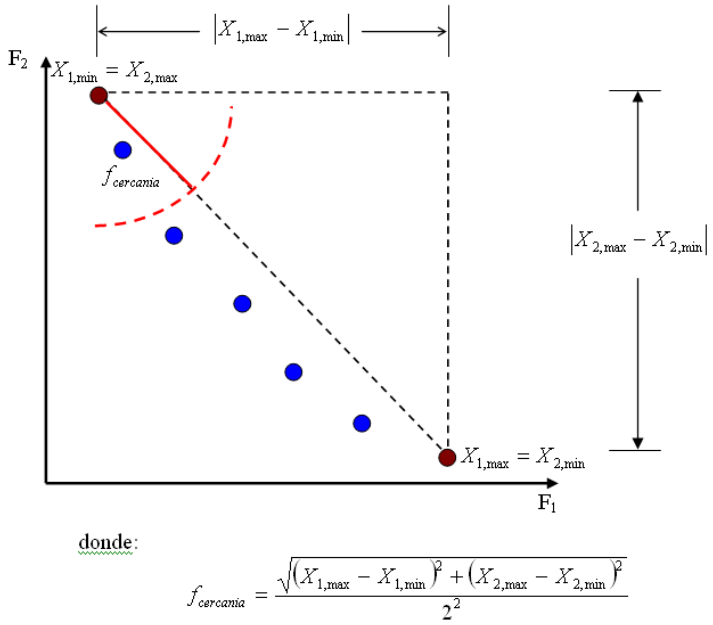

Figura 5.1: Parámetro $f_{cercania}$ para dos funciones objetivo

ción), el hijo (vector) $\overrightarrow{h}$ se genera de acuerdo a:

$$x \in U(0,1) \begin{cases} h_j = p_{r1,j} + F \cdot (p_{r2,j} - p_{r3,j}), & \text{si } x < p_{cruza}; \\ h_j = p_{ref,j}, & \text{de otra forma.} \end{cases}$$

donde:
$j = 0, 1, 2, \ldots, var - 1$. (var = número de variables de cada vector solución)
$p_{r1}, p_{r2}, p_{r3} \in [0, P-1]$, son enteros y diferentes, y $F > 0$.

Los enteros r_1, r_2 y r_3 son los índices de los padres seleccionados y ref es el índice del padre de referencia. F es un factor real y constante que controla la amplificación en la variación diferencial, por lo que valores muy cercanos al 1 hacen que el hijo se mueva lejos del padre p_{r1} y si tiene valores cercanos al 0 el hijo se parecerá mucho al padre p_{r1}. Si analizamos este factor considerando que se quiere que el hijo mejore casi siempre al padre, entonces si se aleja mucho de él se puede perder en la búsqueda y si se queda cerca no lo mejora sustancialmente. Por ende, un valor de $F = 0{,}5$ hace que el hijo mantenga las buenas propiedades del padre para mejorarlo sin cambiar lo suficiente como para empeorar la solución.

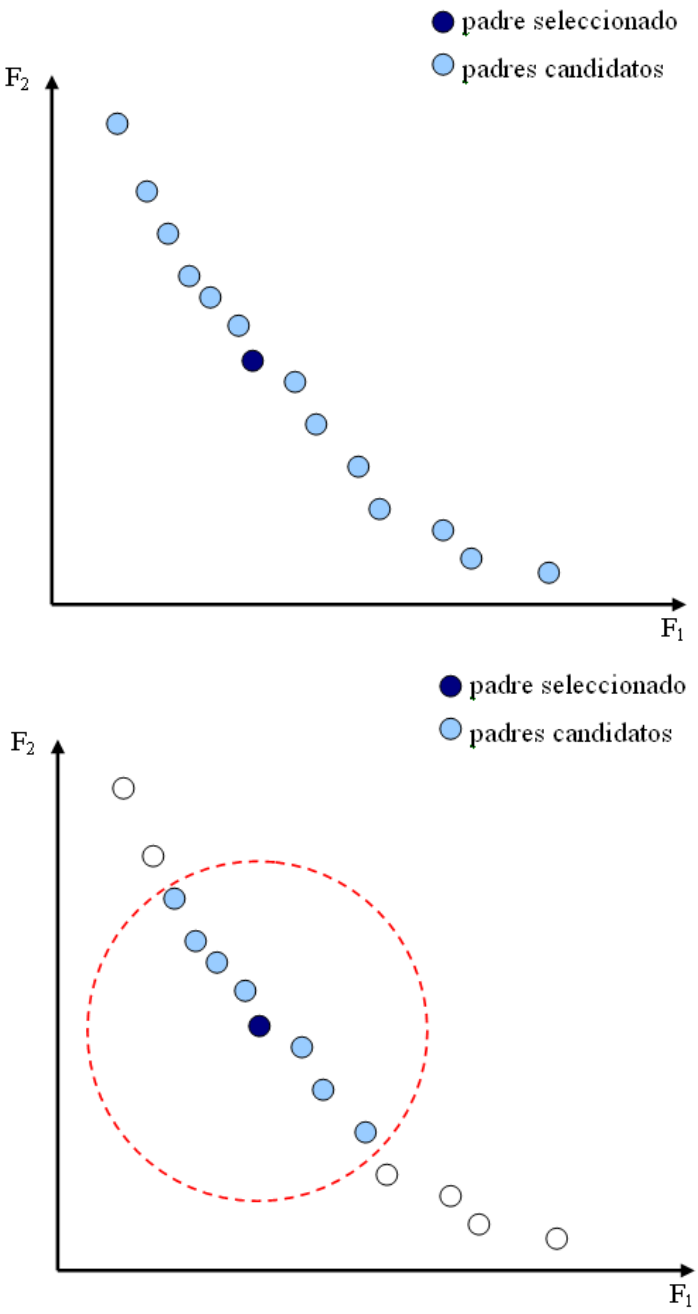

Figura 5.2: Selección aleatoria (arriba) y selección elitista (abajo)

5.1.5. Mutación uniforme

Aunque la Evolución Diferencial no maneja el operador de mutación en ninguno de sus esquemas, en este trabajo se incluyó este operador debido a que en problemas con restricciones y particularmente difíciles, la cruza por sí sola no lograba llegar a ciertas regiones del problema, y como consecuencia, no se generaban los frentes en algunas regiones. Dado que existe evidencia clara de que la mutación ayuda a la exploración de los algoritmos evolutivos, entonces se decidió incluirla en este trabajo. Se adoptó una mutación uniforme definida (para representación real) de la manera siguiente:

$$x \in U(0,1) \begin{cases} h_j = \text{LI}_j + U(0,1) \cdot (\text{LS}_j - \text{LI}_j)], & \text{si } x < p_{mutacion}; \\ h_j = p_j, & \text{de otra forma.} \end{cases}$$

donde:
$j = 0, 1, 2, \ldots, v-1$. (v = número de variables de cada vector solución)
LI_j y LS_j son los límites inferior y superior respectivamente de cada variable.

5.1.6. Hijo vs padre

Una vez que se ha generado un hijo, se tiene que realizar la evaluación de éste con respecto a sus funciones objetivo para conocer qué tan apto es. Después se compara con el padre de referencia, y se elige a uno de ellos, el cual trasciende a la siguiente generación. Es importante mencionar que el esquema que se maneja para los problemas con restricciones requiere que las restricciones se normalicen entre 0 y 1 para conocer la distancia real a la zona factible para todos aquellos individuos no factibles. La forma en que se realiza esta normalización es transparente para el usuario, pues el algoritmo se encarga de realizarla por sí solo. Este mecanismo de normalización se describe a continuación:

Para cada restricción C_i se utilizan dos variables auxiliares: CS y CI, las cuales guardarán el valor máximo y mínimo para cada restricción que se tenga. Por tanto, cada vez que se requiera normalizar una restricción NC_i, se realiza lo siguiente:

$$NC_i = \frac{\text{C}_i - \text{CI}_i}{\text{CS}_i - \text{CI}_i}$$

A continuación se darán a conocer las reglas de comparación entre el padre y el hijo para las funciones sin restricciones y después para aquellas que contienen restricciones.

Funciones sin restricciones

Primero se realiza una comparación de dominancia de Pareto entre padre e hijo:

* *si el padre domina al hijo*, se elige al padre
* *si el hijo domina al padre*, se elige al hijo
* *si son no dominados entre ellos*, se realiza un $flip$ (0.5) para saber quién trasciende a la siguiente generación.

Funciones con restricciones

Aquí primero se verifica si son o no factibles, y después se checa la dominancia de Pareto, y se ve entonces lo siguiente:

* *si el padre es no factible y el hijo es no factible*, se elige al individuo que esté más cerca de la zona factible.
* *si el padre es factible y el hijo es no factible*, se elige al hijo si y sólo si el hijo está a una distancia de $0{,}1$ de la zona factible y además se hace un $flip$ (0.5). De lo contrario, se elige al padre.
* *si el padre es no factible y el hijo es factible*, se elige al padre si y sólo si el padre está a una distancia de $0{,}1$ de la zona factible y además se hace un $flip$ (0.5). De lo contrario, se elige al hijo.
* *si el padre es factible y el hijo es factible*, se verifica la dominancia de Pareto y se elige tal y como si se tratara de un problema sin restricciones.

En el esquema anterior se busca que individuos que son no factibles pero que se encuentren muy cerca de alcanzar la zona factible, no sean descartados, pues suelen ser buenas soluciones que alcanzan regiones no exploradas del espacio de búsqueda ayudando a cubrir estas áreas que son particularmente difíciles en algunos problemas con restricciones. Se

ha demostrado previamente que el mantener individuos no factibles que estén en las orillas de la zona factible, ayuda a obtener individuos que estén en la frontera de factibilidad los cuales, muchas veces, constituyen regiones importantes del frente de Pareto verdadero [31, 37].

5.1.7. Mecanismo de aceptación en el archivo externo

El mecanismo que se adoptó en la primera versión del algoritmo propuesto en esta tesis es el de *rejilla* o *malla* muy parecido al que se adopta en PAES [8]. En este esquema se maneja una malla autoadaptativa que requiere de un parámetro m el cual define el número de hiper-cuadrados en los que se dividirá el frente de Pareto actual. En caso de que un individuo sea no dominado con respecto a todos aquellos que están en el archivo externo, se compara su hiper-cuadrado con aquél que tiene más individuos. Si este nuevo individuo radica en un hiper-cuadrado menos poblado, entonces se acepta y un miembro de los individuos que están en el hiper-cuadrado más poblado es retirado.

Insertar individuo en la Rejilla

Cada vez que se quiere insertar un nuevo elemento a esta rejilla se tienen que evaluar ciertos criterios de aceptación, éstos son:

1.- *si la rejilla está vacía*, siempre se acepta al nuevo individuo

2.- *si el nuevo individuo domina al menos a un individuo de la rejilla*, entonces el nuevo individuo se acepta y se retiran a todos aquellos individuos existentes que sean dominados por el nuevo individuo.

3.- *si el nuevo individuo es dominado por un individuo de la rejilla*, se rechaza al nuevo individuo.

4.- *si el nuevo individuo es no dominado con respecto a todos los individuos de la rejilla*, se acepta este nuevo individuo si y sólo si la rejilla no está llena. En caso de que la rejilla esté llena, entonces se verifica el hiper-cuadrado en el que se ubicaría al nuevo individuo y si no está en el hiper-cuadrado más poblado, entonces se acepta y se elimina un individuo del hiper-cuadrado más poblado.

Eliminación de individuos de la rejilla

Cuando se cumple el caso 4 antes indicado, al insertar un nuevo individuo se procede a eliminar a otro. El individuo a eliminarse será uno que esté dentro del hiper-cuadrado más poblado o con mayor número de elementos. En el caso de PAES, se elimina a cualquiera de ellos aleatoriamente. En esta propuesta se realiza un cálculo extra, en el que se mide una distancia vecinal para cada uno de los individuos que estén dentro del hiper-cuadrado más poblado. Así, el individuo que tenga la distancia vecinal más pequeña es el que se elimina. La función es la siguiente:

$$D(x) = \frac{(\text{mín}(|x - x_i|) + \text{mín}(|x - x_j|))}{2}$$

donde $x \neq x_i \neq x_j$. Esto es, la distancia vecinal es el promedio euclidiano entre sus dos puntos más cercanos. De tal forma x_i y x_j resultan ser los vecinos más próximos a cada uno de los individuos.

Este mecanismo nos ayuda a eliminar al peor de los individuos que estén dentro del hiper-cuadrado más poblado, ayudando a formar una mejor distribución de las soluciones no dominadas que conforman la solución final del problema. Aunque el realizar este cálculo se incrementa el costo computacional del algoritmo, dicho incremento no es significativo, pues en promedio se evalúa esta función para diez individuos que están dentro del hiper-cuadrado más poblado.

Actualización del parámetro de la rejilla

Este mecanismo se activa por primera vez cuando se alcanza el límite de individuos permitidos en el archivo externo. Se realiza un cálculo para conocer el tamaño de cada hiper-cuadrado. Esto es, se tienen que calcular los límites máximos y mínimos en cada función objetivo de los individuos que están dentro del archivo externo. Con esta información, se procede a dividir la región entre el parámetro m para conocer cuántos hiper-cuadrados se tendrán y de qué tamaño será cada uno. El cálculo del tamaño del hiper-cuadrado para cada función objetivo es el siguiente:

$$\text{tamaño}_i = \frac{X_{i,max} - X_{i,min}}{malla} \tag{5.1}$$

donde:
i = número de funciones objetivo

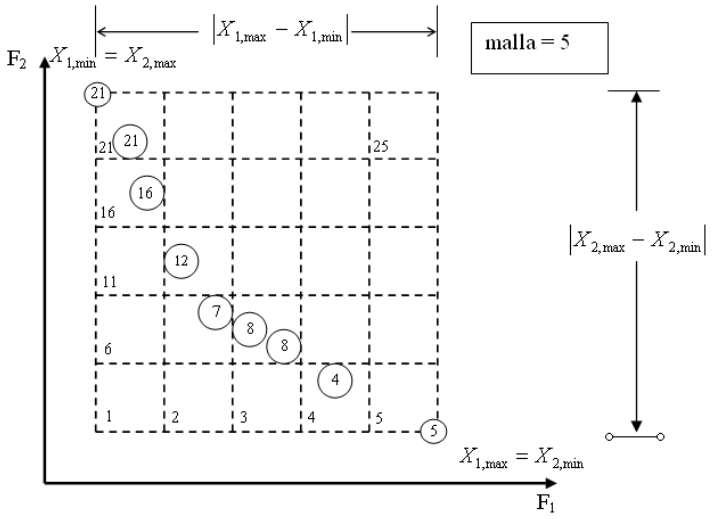

Figura 5.3: Asignación de hiper-cuadrados en la rejilla

$X_{i,max}$ = valor máximo de la i-ésima función objetivo del archivo externo
$X_{i,min}$ = valor mínimo de la i-ésima función objetivo del archivo externo

Como se puede notar, este parámetro se puede calcular dentro del mismo ciclo al del parámetro de $f_{cercania}$, pues requiere de los valores máximos y mínimos de las funciones objetivo.

Una vez que sabemos este tamaño de los hiper-cuadrados, procedemos a asignar a cada individuo un hiper-cuadrado (varios individuos pueden estar dentro de uno). En caso de que un individuo caiga en una zona que no está cubierta por la rejilla, se recalculan sus límites tanto inferior como superior según sea el caso. Cada vez que se actualiza la rejilla se tienen que recalcular los hiper-cuadrados asignados para todos los individuos, pues el tamaño de la rejilla ha cambiado.

En la figura 5.3 se puede observar cómo se realiza el cálculo del tamaño de los hiper-cuadrados y la asignación de los individuos a cada uno de ellas para un problema con dos funciones objetivo. En el ejemplo se fija el parámetro $malla = 5$ y como consecuencia se forman 25 hiper-cuadrados de dimensión 2, por ser de dos funciones objetivo.

5.2. ϵ - MyDE

En esta segunda versión del algoritmo, todos los mecanismos descritos quedan de la misma forma excepto porque ahora se utiliza un método diferente para preservar a los individuos en el archivo externo en lugar de la malla autoadaptativa. A este método se le denomina dominancia-ϵ (ϵ-dominance), y fue propuesto por Laumanns et al. [28].

La codificación, selección, cruza, mutación y manejo de restricciones se mantienen tal y como se explicó en detalle en las secciones anteriores. Lo único que cambia es la forma en que se conservan a los soluciones del archivo externo, la cual se explica a continuación.

5.2.1. Concepto de dominancia-ϵ

En esta sección se definen algunos conceptos relacionados con la dominancia -ϵ. Para los algoritmos presentados en esta parte se asume que todos los objetivos se minimizan. Sin embargo, con simples modificaciones a las definiciones de dominancia, estos algoritmos pueden ser usados en maximización o problemas combinados de minimización y maximización.

Relación de Dominancia

La relación de dominancia de Pareto se definió en la sección 3.1.6 y se ilustra en la figura 5.4 (izquierda) a fin de entender mejor su diferencia con respecto a la dominancia-ϵ

Si $f, g \in \Re^m$, entonces f se dice que domina a g, denotado como: $\vec{f} \prec \vec{g}$, si y sólo si:

$$1. \forall i \in \{1, \ldots, m\} : f_i \leq g_i$$

$$2. \exists j \in \{1, \ldots, m\} : f_j < g_j$$

Así, de la misma forma, el concepto de dominancia-ϵ se ilustra en la figura 5.4 (derecha) y se define de la siguiente manera.

Si $f, g \in \Re^m$, entonces $\vec{f}$ se dice que ϵ-domina a $\vec{g}$ para algún ϵ >0, denotado como: $f \prec_\epsilon g$, si y sólo si para todo $i \in$ {1, . . . , m }.

$$(1 - \epsilon) \cdot f_i \leq g_i$$

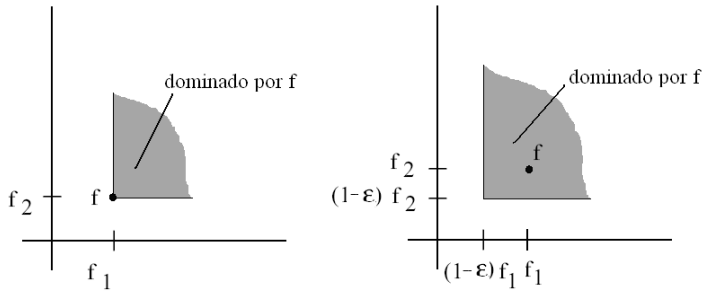

Figura 5.4: Concepto de dominancia (izq) y dominancia-ϵ (der)

Puede verse que ésta es una forma relajada de dominancia que amplía la zona de dominancia. Su uso se explica a continuación.

5.2.2. Mecanismo de aceptación en el archivo externo usando dominancia-ϵ

Para el nuevo individuo que es candidato a ingresar al archivo externo, se le compara con cada uno de los miembros del archivo externo usando el criterio dominancia-ϵ. A continuación se describe brevemente este procedimiento el cual se ha utilizado en otro algoritmo llamado $\epsilon - MOEA$ desarrollado por Deb en 2003 [12].

A cada solución dentro del archivo externo se le asigna un arreglo de identificación ($\mathbf{B} = (B_1, B_2, \ldots, B_M)^T$, donde M es el número total de funciones objetivo) como sigue:

$$B_j(f) = \begin{cases} (\lfloor (f_j - f_j^{min})/\epsilon_j \rfloor), & \text{para minimizar } f_j; \\ (\lceil (f_j - f_j^{min})/\epsilon_j \rceil), & \text{para maximizar } f_j. \end{cases}$$

donde: f_j^{min} es el valor mínimo posible del j-ésimo objetivo y ϵ_j es la tolerancia permitida en el j-ésimo objetivo, ambos parámetros los debe definir el usuario [28]. Este arreglo de identificación divide todo el espacio de las funciones objetivo en hiper-cajas, cada una de las cuales tiene un tamaño ϵ_j en el j-ésimo objetivo. En la figura 5.4 se ilustra que la solución P ϵ-domina a toda la región $ABCDA$ (en el caso de minimización), mientras que la definición original de dominancia sólo permitiría que P dominara la región $PECFP$. El arreglo de identificación para P son las coordenadas

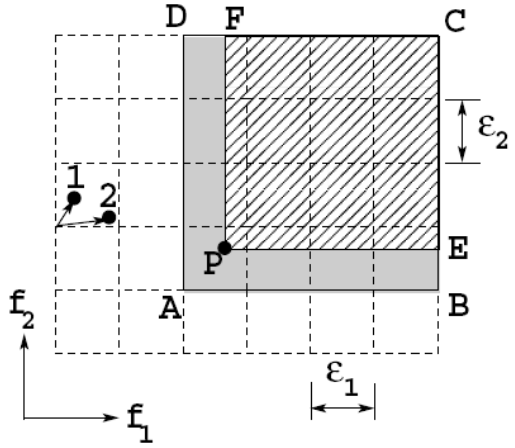

Figura 5.5: Concepto de dominancia-ϵ (para minimizar f_1 y f_2)

del punto A en el espacio de los objetivos. Con el arreglo de identificación calculado para el nuevo individuo a ingresar c_1 y para cada miembro del archivo externo a, se procede a verificar si se cumple uno de los 4 casos siguientes (ilustrados en la figura 5.6):

1. Si el arreglo de identificación B_a de cualquier miembro del archivo externo a domina al nuevo individuo c_i, entonces esto significa que el nuevo individuo es ϵ-dominado por un miembro de los del archivo externo y por lo tanto *no es aceptado*. Este caso se ilustra en la figura 5.6 (a).

2. Si B_{c_i} del nuevo individuo domina a cualquier B_a del archivo externo, este miembro es eliminado y el nuevo individuo *es aceptado*. Este caso se ilustra en la figura 5.6 (b).

   Si ninguno de los dos casos arriba mencionados ocurre, significa que el nuevo individuo es ϵ-no dominado con respecto a los miembros del archivo externo. Entonces surgen dos posibles casos:

   a) El nuevo individuo comparte el mismo vector **B** con algún individuo del archivo externo (esto significa que pertenecen a la misma hiper-caja). Estos individuos se compararan con la dominancia convencional de Pareto. Si el nuevo individuo domina al

miembro del archivo o es no dominado con respecto al miembro del archivo pero se encuentra más cerca al vector **B** (en términos de distancia euclidiana) que el miembro del archivo , entonces el *nuevo individuo es aceptado*. Ambos casos se ilustran en la figura 5.6 (c). Las soluciones 1 y 2 en la figura 5.5 también ilustran este caso, en el que ambas soluciones ocupan la misma hiper-caja (o tienen el mismo vector **B**) y éstas son no dominadas (usando dominancia de Pareto). Pero la solución 1 tiene una menor distancia euclidiana al vector **B**, por lo que se retiene la solución 1 y se elimina la solución 2.

b) Este caso se ilustra en la figura 5.6 (d). Es importante hacer notar que existe la condición de que sólo puede existir una única solución con un cierto vector **B**. Esto significa que cada hiper-caja en el conjunto de óptimos de Pareto puede ser ocupada sólo por una única solución, derivándose por tanto 2 propiedades: (i) se mantienen soluciones bien distribuidas y (ii) el archivo final contendrá todas las soluciones que fueron aceptadas y por lo tanto, no es necesario especificar el límite superior del tamaño de archivo externo. Más bien, estará acotado en función del valor escogido para ϵ.

Una de las complicaciones de este mecanismo es la de escoger un valor apropiado de ϵ, pues este valor dicta la cardinalidad de las soluciones obtenidas. Si se elige un valor grande de ϵ_i, se obtendrán menos soluciones, y viceversa. Aún así, esta complicación resulta ser una ventaja si consideramos el punto de vista del tomador de decisiones. Esta propiedad puede resultar particularmente útil para problemas con alta dimensionalidad (con más funciones objetivo) en los que usualmente existe un gran número de soluciones en el conjunto de óptimos de Pareto. Así, escogiendo un valor adecuado de ϵ, el tomador de decisiones puede especificar el número de soluciones que desee en cada función objetivo, permitiendo así la manipulación del número de soluciones que se necesite según su conveniencia.

5.2.3. Comparación de ambos mecanismos utilizados

Aunque ambos mecanismos, el de la $malla$ y el de dominancia-ϵ parezcan muy similares, existen algunas diferencias que vale la pena destacar.

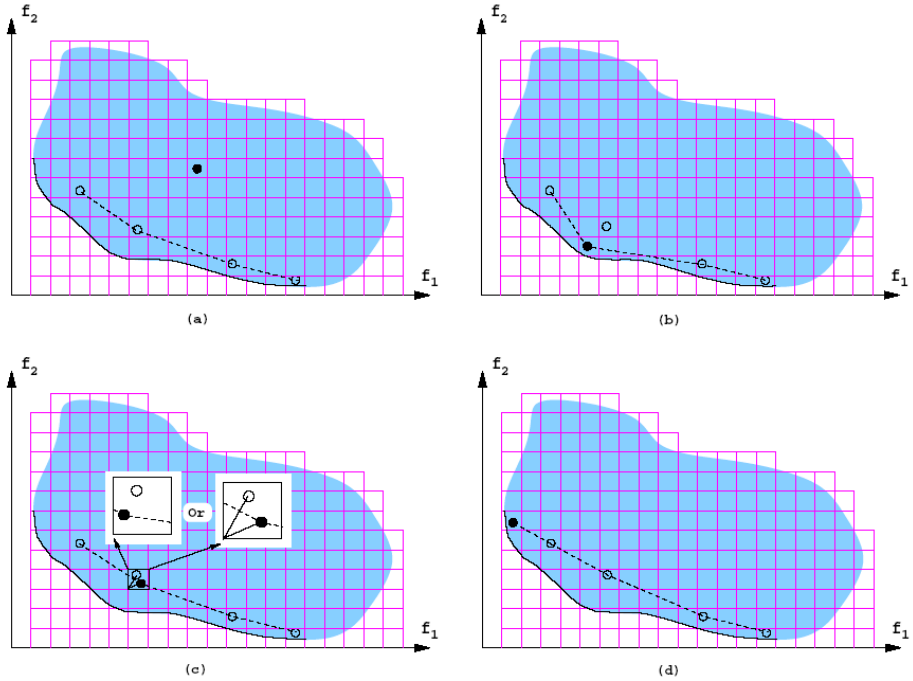

Figura 5.6: Concepto de dominancia-ϵ (para minimizar f_1 y f_2)

Ambos mecanismos dividen el espacio de las funciones objetivo en un determinado número de hiper-cajas.

En ambos mecanismos, el nuevo individuo se compara continuamente con la población secundaria para su inclusión. En caso de que este nuevo individuo sea no dominado con todos los miembros del archivo, se compara con la hiper-caja que tiene mayor número de individuos en ella. Si este nuevo individuo radica en un área menos poblada es aceptado y se elimina un individuo que esté en la hiper-caja más poblada. El concepto de dominancia-ϵ implementado en el segundo mecanismo junto con el chequeo de vectores **B**, no permiten que dos soluciones no dominadas con una diferencia menor a ϵ_i en la i-ésima función objetivo sean presentadas como partes de la solución o archivo final. En la otra forma (en la malla), se permite que más de un individuo esté presente en cada hiper-caja. Esto significa que menos soluciones no dominadas pueden ser parte de la solución en el archivo externo cuando se usa el mecanismo de dominancia-ϵ en lugar de la $malla$. Así entonces, el uso de dominancia-ϵ no sólo reducirá el tamaño del conjunto de óptimos de Pareto, sino que también ayuda cuando un usuario no está interesado en obtener demasiadas soluciones no dominadas. Por esta misma razón el tiempo computacional se espera que sea más pequeño cuando se usa dominancia-ϵ, pues no se tiene que estar buscando al individuo que tendrá que ser eliminado, sino que básicamente se hace un reemplazo de individuos en caso de que compartan la misma hiper-caja.

En cuanto a la distribución de soluciones, el mecanismo de dominancia-ϵ puede llegar a ser mejor que el de m siempre y cuando se elijan los valores correctos de ϵ.

En la práctica, puede ser difícil elegir un valor adecuado de ϵ debido a que en muchas ocasiones el usuario no tiene el conocimiento necesario para ajustarlo. Debido a esto, en esta tesis se propuso un mecanismo de ajuste para el valor de ϵ, es decir, el usuario no experimentado dará el número de soluciones que requiera, y el propio algoritmo ajusta los valores del vector-$\vec{\epsilon}$ para que, en promedio, se obtenga ese número de soluciones requeridas. Debe tomarse en cuenta que la distribución de soluciones final puede degradarse debido a la forma especial de algunos frentes de Pareto (en especial, las zonas que están muy horizontales y/o verticales), pero creemos que este esquema es un buen compromiso para situaciones en las que el usuario no desea ajustar manualmente el vector-ϵ.

5.2.4. Cálculo del vector $\vec{\epsilon}$

Si deseamos conocer el valor del vector $\vec{\epsilon}$ y el usuario nos ha pedido un cierto número de soluciones no dominadas del problema, entonces, el algoritmo tiene que realizar un pre-cálculo en el cual genera N soluciones dentro de los límites de las variables de decisión a fin de calcular la relación existente entre las diversas funciones objetivo. El número de soluciones N que se pueden generar también son a elección del usuario (considerando que a mayor número de soluciones, mejor aproximación se hace del vector, pero se requiere más tiempo).

Una vez que se generan las N soluciones, se determinan los máximos y los mínimos valores para las funciones objetivo, y se calcula el valor del vector $\vec{\epsilon}$ usando la siguiente función:

$$\epsilon_i = \frac{(X_i^{max} - X_i^{min})}{l}$$

donde: $l = 256$ si son dos funciones objetivo, o $l = 64$ si son 3 funciones objetivo.

Los datos del valor len se obtuvieron al realizar varias pruebas con los diferentes tipos de frente con las diversas funciones, además de probar con muchos valores. De estas pruebas se observó que este dato puede fijarse en 256 si se maneja un espacio con dos funciones objetivo y en 64 si se manejan tres funciones objetivo. Para un número mayor de funciones objetivo, este valor tendría que determinarse empíricamente. El valor que se obtiene resulta aproximado y por eso es que se realiza una ajuste posterior para aproximar el valor del vector-$\vec{\epsilon}$, a fin de acercarse más a su valor verdadero.

Cuando el frente tiene una forma convexa y continua, se generan en promedio 40 soluciones en el frente; en frentes cóncavos y continuos se generan en promedio 25 soluciones. En frentes desconectados, este valor promedio disminuye.

Así entonces, ya que tenemos un vector-$\vec{\epsilon}$ aproximado, se ejecuta el algoritmo en 5 generaciones, y se verifica el número de soluciones dentro del archivo externo. Si el número de individuos es menor al deseado, entonces se puede reducir el vector-$\vec{\epsilon}$ permitiendo que más individuos formen parte del archivo. Después de 5 generaciones más, se vuelve a realizar el conteo de soluciones y, si es necesario, se reduce nuevamente el vector-$\vec{\epsilon}$. Este proceso se realiza cada 5 generaciones hasta que el número solicitado

de soluciones en el archivo rebase el número solicitado originalmente por el usuario. Es importante mencionar que en los problemas con restricciones, se recomienda dejar correr el algoritmo durante 15 generaciones para actualizar este valor, pues estos problemas requieren de más número de iteraciones para estabilizar la población de no dominados en el archivo externo.

Como se explicó anteriormente, el mecanismo de aceptación en dominancia -ϵ sólo permite agregar soluciones y no eliminarlas, por lo que si se generan más soluciones de las que pidió el usuario, resulta imposible quitarlas debido a restricciones propias del mecanismo.

También es importante aclarar que resulta muy difícil encontrar una relación precisa entre el valor del vector-$\vec{\epsilon}$ y el número de soluciones requeridas por el tomador de decisiones, pues ésta depende sustancialmente de la forma que tenga el frente de Pareto del problema (continuo o discontinuo, cóncavo o convexo). Lo que se sugiere por tanto es aproximar el valor del vector-$\vec{\epsilon}$ conforme se van generando las soluciones del problema. Este mecanismo funcionó adecuadamente para los problemas de prueba que se realizaron, aunque no se garantiza su funcionamiento en un problema arbitrario.

Capítulo 6

Comparación de resultados

Para validar las dos variantes de la técnica propuesta en esta tesis, se adoptó un conjunto de 10 funciones de prueba (5 funciones sin restricciones y 5 con restricciones). Inicialmente presentaremos los resultados para las funciones de prueba sin restricciones. A fin de permitir una comparación cuantitativa de resultados, usaremos las métricas descritas en la sección 3.4. Debido a que las métricas no siempre reflejan el comportamiento de un algoritmo evolutivo multiobjetivo, se proporcionarán también representaciones gráficas de los frentes de Pareto (en la mediana con respecto a la métrica de ER) para cada algoritmo. Las técnicas comparadas son:

1. Non-Dominated Sorting Genetic Algorithm - II (NSGA - II). Este algoritmo fue descrito en la sección 3.3.3 propuesto por Deb et al. [11].

2. Pareto Archieved Evolution Strategy (PAES). Este algoritmo fue descrito en la sección 3.3.4 propuesto por Knowles y Corne [8].

3. Pareto Differential Evolution (PDE). Este algoritmo fue descrito en la sección 4.4.1 propuesto por H. A. Abbass y R. Sarker [2].

Se realizó la comparación con estas técnicas, pues se considera que son los algoritmos representativos del estado del arte. El NSGA-II es actualmente uno de los más competitivos para resolver problemas multiobjetivo, PAES utiliza una malla autoadaptativa al igual que una de las variantes de la técnica propuesta, y PDE es el primer algoritmo que utiliza la evolución diferencial para resolver problemas multiobjetivo y del que más información se cuenta.

6.1. Funciones de prueba sin restricciones

La primera función fue propuesta por Rendón [42]; el segundo problema multiobjetivo fue propuesto por Kursawe en [27]; el tercer problema fue propuesto por Deb en [10]; el cuarto problema fue propuesto por Lis [29] y la última función fue propuesta por Deb [13].

Las funciones de prueba se muestran a continuación.

MOP1. Este problema fue propuesto en [42].

Minimizar:

$$\vec{f}(x) = (f_1(x), f_2(x))$$

donde:

$$\begin{aligned} f_1(\vec{x}) &= \frac{1}{(x^2 + y^2 + 1)} \\ f_2(\vec{x}) &= x^2 + 3y^2 + 1 \end{aligned}$$

y:

$$-3 \leq x, y \leq 3$$

Este problema posee un frente de Pareto convexo, su conjunto de óptimo está conectado, y es considerado como un problema "sencillo" pues su espacio de búsqueda no es tan grande y las funciones objetivo no son particularmente complicadas.

MOP2. Este problema fue propuesto en [27].

Minimizar:

$$\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}))$$

donde:

$$\begin{aligned} f_1(\vec{x}) &= \sum_{i=1}^{n-1} \left(-10e^{(-0,2)\sqrt{x_i^2 + x_{i+1}^2}} \right) \\ f_2(\vec{x}) &= \sum_{i=1}^{n} (|x_i|^a + 5\sin(x_i^b)) \end{aligned}$$

con:

$$\begin{aligned} a &= 0{,}8 \\ b &= 3 \\ n &= 3 \end{aligned}$$

y:

$$-5 \leq x_1, x_2, x_3 \leq 5$$

Este problema tiene el frente de Pareto desconectado, y el conjunto de óptimos de Pareto también es desconectado. Sus 4 regiones de búsqueda desconectadas, y la escalabilidad del problema en su número de variables de decisión hacen a éste un problema complejo [6].

MOP3. Este problema fue propuesto en [10].

Minimizar:

$$\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}))$$

donde:

$$\begin{aligned} f_1(\vec{x}) &= x_1 \\ f_2(\vec{x}) &= \frac{g(x_2)}{x_1} \end{aligned}$$

con:

$$g(x_2) = 2 - e^{-(\frac{x_2-0{,}2}{0{,}004})^2} - 0{,}8e^{-(\frac{x_2-0{,}6}{0{,}4})^2}$$

y:

$$0{,}1 \leq x_1, x_2 \leq 1$$

Este problema tiene el frente de Pareto convexo, y el conjunto de óptimos de Pareto está conectado. Sin embargo, este problema es multi-frontal (es decir, cuenta con varios frentes de Pareto "locales", a los que un algoritmo podría converger prematuramente si no tiene una buena capacidad de exploración).

MOP4. Este problema fue propuesto en [29].

Minimizar:

$$\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}))$$

donde:

$$\begin{aligned} f_1(\vec{x}) &= \sqrt[8]{x^2 + y^2} \\ f_2(\vec{x}) &= \sqrt[4]{(x - 0{,}5)^2 + (y - 0{,}5)^2} \end{aligned}$$

y:

$$-5 \leq x, y \leq 10$$

Este problema tiene el frente de Pareto desconectado y cóncavo, y el conjunto de óptimos de Pareto está desconectado también.

MOP5. Este problema fue propuesto en [13].

Minimizar:
$\vec{f_1} = (1+g(x_M))(cos(x_1\pi/2))(cos(x_2\pi/2))...(cos(x_{M-2}\pi/2))(cos(x_{M-1}\pi/2)),$
$\vec{f_2} = (1+g(x_M))(cos(x_1\pi/2))(cos(x_2\pi/2))...(cos(x_{M-2}\pi/2))(sen(x_{M-1}\pi/2)),$
$\vec{f_3} = (1 + g(x_M))(cos(x_1\pi/2))(cos(x_2\pi/2))...(sen(x_{M-2}\pi/2)),$

$$\vdots \qquad \vdots$$

$\vec{f_{M-1}} = (1 + g(x_M))(cos(x_1\pi/2))(sin(x_2\pi/2)),$
$\vec{f_M} = (1 + g(x_M))(sin(x_1\pi/2)).$

con:

$$\begin{aligned} g(x_M) &= \sum_{x_i \in X_M} (x_i - 0{,}5)^2 \\ M &= 3, \\ k &= 10 \\ n &= M + k - 1 \end{aligned}$$

y:

$$0 \leq x_i \leq 1 \quad \forall \quad i = 1, 2, ..., n$$

Este problema es fácilmente escalable tanto en el número de funciones como en el de variables, por lo que permite evaluar si un algoritmo es capaz de resolver problemas con un espacio de búsqueda muy grande.

Parámetro	MyDE	NSGA-II	PAES	e-MyDE	PDE
P	100	100	100	100	100
NP	100	100	100	100 (auto)	100
G_{max}	50	50	50	50	50
P_c	0.95	0.8	nd	0.95	nd
P_m	1/N	1/N	0.05	1/N	nd
F	0.5	nd	nd	0.5	N(0, 1)
elitismo	0.2	nd	nd	0.2	nd
malla	(20^d)	nd	(2^{d+8})	nd	nd

donde: N = número de variables
d = número de funciones
nd = no requiere parámetro

Tabla 6.1: Parámetros utilizados en los diversos algoritmos para realizar las pruebas y comparaciones.

6.2. Resultados

Los parámetros utilizados para obtener los resultados que se muestran más adelante para cada uno de los distintos algoritmos se muestran en la tabla 6.1.

El parámetro P (población de cada generación) y G_{max} (número de generaciones) son constantes para todos los algoritmos, pues como se intenta comparar de manera justa el desempeño de éstos al resolver los problemas, debe efectuarse el mismo número de evaluaciones de la función de aptitud con cada uno de ellos. En este caso se realizaron 5,000 evaluaciones en los problemas con dos funciones objetivo y 15,000 evaluaciones en los problemas con 3 funciones objetivo.

El parámetro NP es el tamaño máximo del archivo externo, al cual se le da un valor de 100 el cual es razonable para encontrar una solución apropiada. En el caso de ϵ-MyDE, se realiza un cálculo previo a la solución del problema a fin de ajustar los valores de $epsilon$ que se requieren, obteniendo así una aproximación del número de soluciones no dominadas de NP.

P_c y P_m son los porcentajes de aplicación de la cruza y la mutación (cuando se requiera).

El parámetro F es propio de la Evolución Diferencial y sólo lo requieren aquellos algoritmos que utilizan la ED.

El parámetro de $elitismo$ lo usan los algoritmos propuestos para mejorar la búsqueda una vez que hayan pasado cierto número de generaciones.

Para las pruebas reportadas en esta tesis se usó un valor de 0.2 con el cual se obtuvo buen desempeño.

La $malla$ se usa para denotar el número de nodos que tendrá la rejilla para aquellos algoritmos que la necesiten.

A continuación se muestran las gráficas que comparan el frente de Pareto verdadero con los resultados de nuestra técnica y los distintos algoritmos en cada una de las funciones de prueba. Los frentes de Pareto verdaderos fueron generados por enumeración y las ejecuciones mostradas fueron realizadas con semillas de números aleatorios generadas al azar, mostrando en la gráfica la mediana con respecto a la métrica de TE.

Junto con las gráficas, se muestran dos tablas:

1. la primera corresponde a los resultados estadísticos de las métricas realizando 20 ejecuciones de cada algoritmo.

2. la segunda es el resultado de la métrica de Cobertura (C) aplicada a la unión de los 20 archivos de cada algoritmo para obtener 5 archivos (uno por cada algoritmo). Estos 5 archivos se comparan entre ellos utilizando la métrica C, primero realizando C $(alg1,\ alg2) = Tabla_{alg1,alg2}$ y después C $(alg2, alg1) = Tabla_{alg2,alg1}$. El resultado final se anota en la tabla correspondiente.

6.2.1. MOP1

Las gráficas correspondientes al MOP1 se muestran en las figuras 6.1, 6.2, 6.3 y el resultado estadístico de la aplicación de las métricas se muestra en la tabla 6.2.

Las gráficas anteriores nos ayudan a evaluar el desempeño de los algoritmos de forma visual. Se ve por ejemplo, que todos los algoritmos se acercaron al frente de Pareto verdadero. La principal diferencia estriba en que no todos se encuentran igualmente distribuidos a lo largo del frente. MyDE cubre todo el frente; NSGA-II, PAES y PDE no lo cubren de forma pareja pues existen pocas soluciones en la parte baja del frente y ϵ-MyDE sí cubre bien el frente aunque en las orillas existen pocas soluciones.

De forma cuantitativa, se utilizan las métricas y se observa que ambos algoritmos propuestos mejoran a los otros 3 algoritmos en TE y en DG, aunque en Distribución el NSGA-II muestra los mejores resultados.

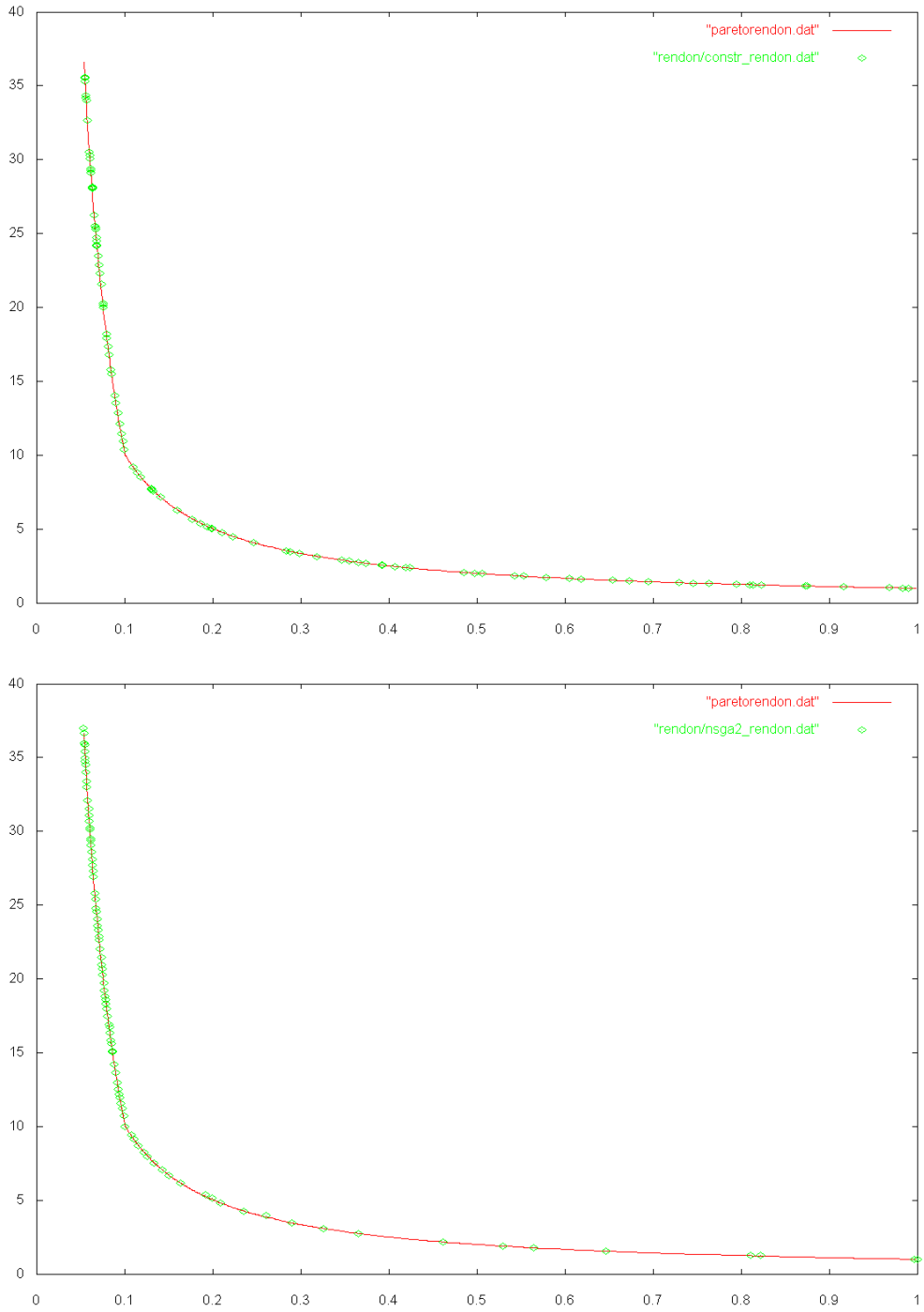

Figura 6.1: Frentes de Pareto generados por MyDE (arriba) y NSGA-II (abajo) para la primera función de prueba (MOP1).

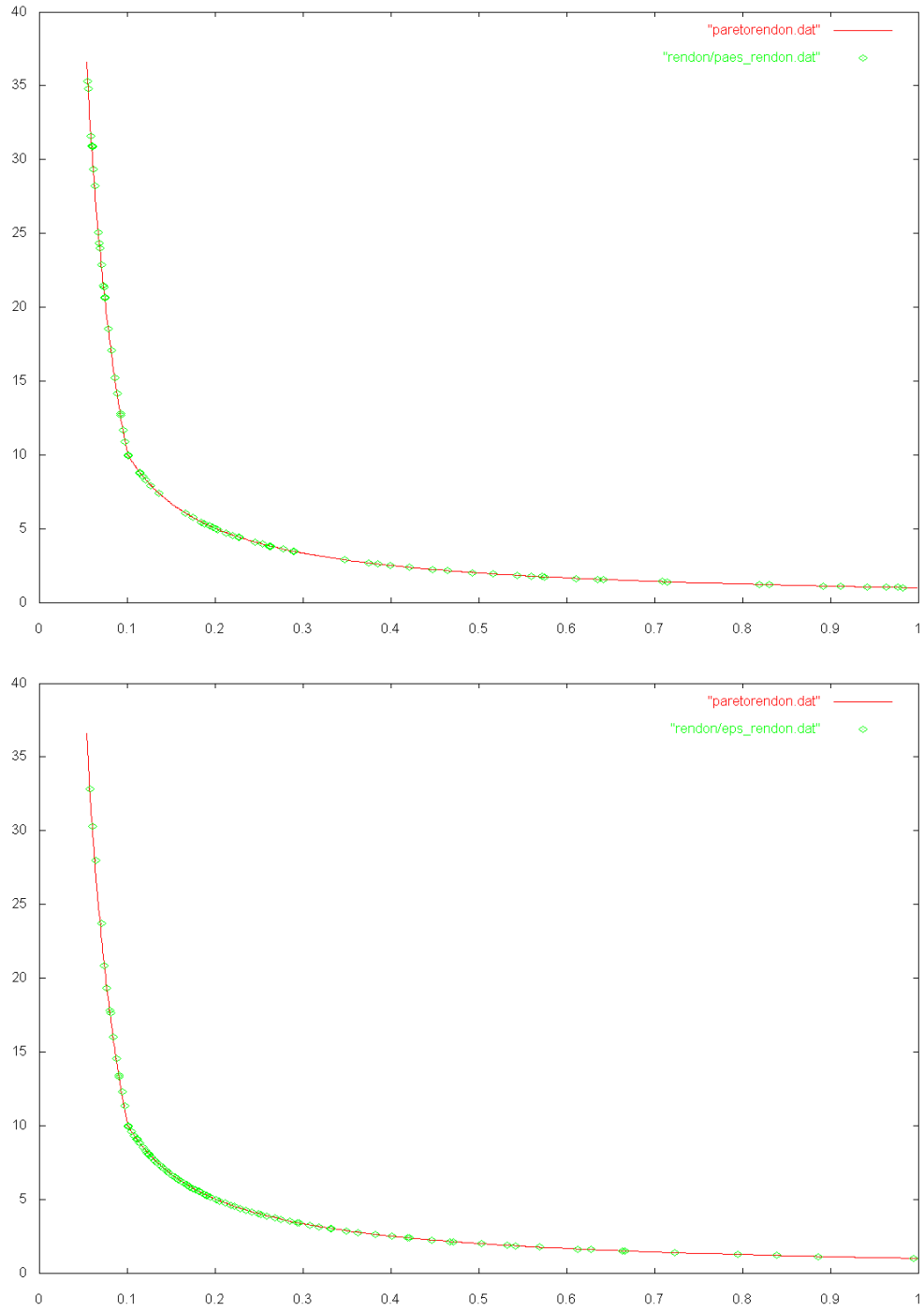

Figura 6.2: Frentes de Pareto generados por PAES (arriba) y ϵ-MyDE (abajo) para la primera función de prueba (MOP1).

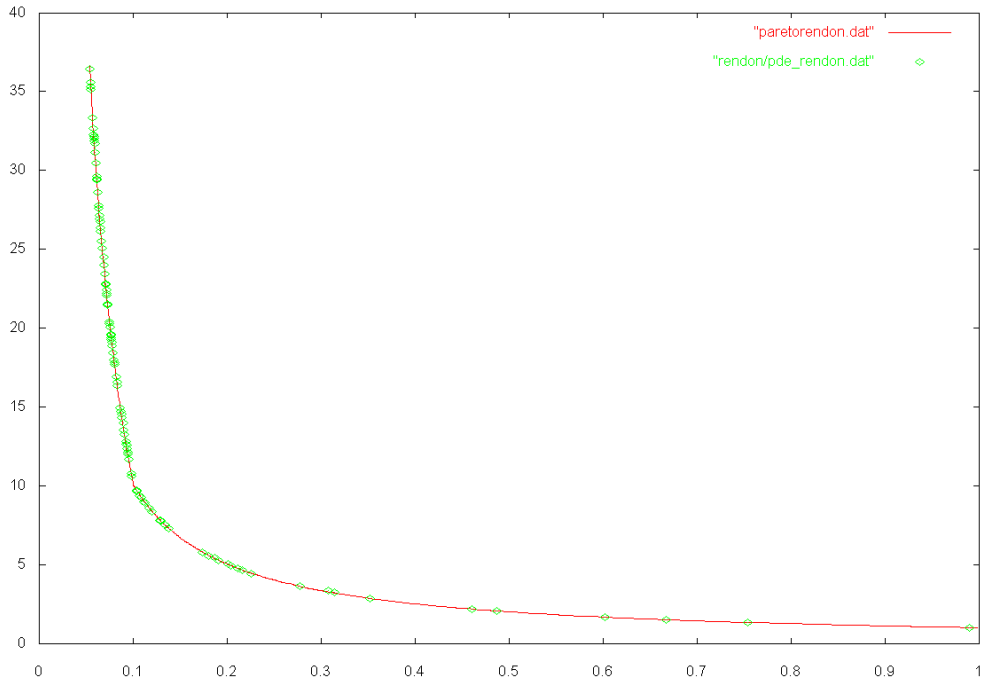

Figura 6.3: Frente de Pareto generado por PDE para la primera función de prueba (MOP1).

Métrica - Algoritmo		media	mejor	peor	desv. Estándar
TE	MyDE	**0.115**	**0.05**	0.2	0.03590924
	NSGA-II	0.2155	0.09	0.32	0.06004165
	PAES	0.255	0.15	0.42	0.0712667
	ϵ-MyDE	0.1415345	0.102041	**0.193878**	**0.02609219**
	PDE	0.3235	0.17	0.5	0.08591765
DG	MyDE	0.02623102	0.0149515	0.0312497	0.00419942
	NSGA-II	0.0313512	0.0284657	0.0368552	**0.00232362**
	PAES	0.01389363	0.00518896	0.0238161	0.00475453
	ϵ-MyDE	**0.01065205**	0.00345107	**0.0179821**	0.00439519
	PDE	0.01472316	**0.00070112**	0.0317935	0.01215242
D	MyDE	0.22160595	0.187436	0.258016	0.02003885
	NSGA-II	**0.17284205**	0.143833	**0.190148**	**0.01090074**
	PAES	0.3123755	0.147196	0.501412	0.091733
	ϵ-MyDE	0.51309125	0.349183	0.74987	0.09661623
	PDE	0.20278115	**0.0391446**	0.896068	0.18802779

C	MyDE	NSGA-II	PAES	ϵ-MyDE	PDE	Dominan
MyDE		0.366337	0.463861	0.217524	0.507921	**0.38891075**
NSGA-II	0.291584		0.392574	0.17728	0.466337	0.33194375
PAES	0.172277	0.24802		0.192053	0.445545	0.26447375
ϵ-MyDE	0.173267	0.203465	0.407426		0.458911	0.31076725
PDE	0.251485	0.385644	0.45	0.214468		0.32539925
Son Dominados	0.22215325	0.3008665	0.42846525	**0.20033125**	0.4696785	

Tabla 6.2: Resultados de Tasa de Error (TE), Distancia Generacional (DG), Distribución (D) y Cobertura (C) para el primer problema (MOP1).

Métrica - Algoritmo		media	mejor	peor	desv. Estándar
TE	MyDE	0.272	0.17	0.39	0.0589915
	NSGA-II	0.4575	0.31	0.64	0.0907208
	PAES	0.449	0.27	0.64	0.1026439
	ϵ-MyDE	**0.15792013**	**0.0654206**	**0.23301**	**0.04288708**
	PDE	0.5085	0.25	0.64	0.0958219
DG	MyDE	0.00397841	0.00324389	0.00636706	0.00084727
	NSGA-II	0.00416653	0.00308047	0.0061631	0.00086198
	PAES	0.00434111	0.00208384	0.00802109	0.00172447
	ϵ-MyDE	**0.00327069**	**0.00288871**	**0.00374484**	**0.00025872**
	PDE	0.00467552	0.00366814	0.00667008	0.000966
D	MyDE	0.09773811	0.0640408	0.114672	0.01709315
	NSGA-II	**0.06180859**	**0.047978**	0.11722	0.01663756
	PAES	0.1709759	0.088026	0.258871	0.0403215
	ϵ-MyDE	0.09448982	0.0548549	**0.109133**	**0.01243547**
	PDE	0.10522477	0.0687954	0.145438	0.0224935

C	MyDE	NSGA-II	PAES	ϵ-MyDE	PDE	Dominan
MyDE		0.765842	0.694059	0.388327	0.765842	0.6535175
NSGA-II	0.395545		0.559901	0.234375	0.633663	0.455871
PAES	0.434653	0.663861		0.267004	0.679703	0.51130525
ϵ-MyDE	0.661881	0.824752	0.759901		0.825743	**0.76806925**
PDE	0.420297	0.676733	0.595545	0.238511		0.4827715
Son Dominados	0.478094	0.732797	0.6523515	**0.28205425**	0.72623775	

Tabla 6.3: Resultados de Tasa de Error (TE), Distancia Generacional (DG), Distribución (D) y Cobertura (C) para el segundo problema (MOP2).

En cuanto a la métrica de cobertura (C) en la que se comparan directamente los algoritmos entre sí, el que mejor resultados obtuvo fue MyDE seguido por ϵ-MyDE ya que son los que cubren en mayor proporción a los demás algoritmos.

6.2.2. MOP2

Las gráficas correspondientes al MOP2 se muestran en las figuras 6.4, 6.5, 6.6 y el resultado estadístico de la aplicación de las métricas se muestra en la tabla 6.3.

Gráficamente, se ve que MyDE cubre perfectamente todo el frente de Pareto verdadero y muestra una buena distribución de los puntos. NSGA-II nunca llegó al punto de la extrema izquierda (-20, 0) ni tampoco cubre todo el frente. PAES deja muchas regiones sin llenar. ϵ-MyDE es el que cubre mejor todas las regiones, estando sobre el frente de Pareto verdadero y mostrando una excelente distribución. PDE también logra una buena aproximación, pero su distribución es regular porque deja al descubierto

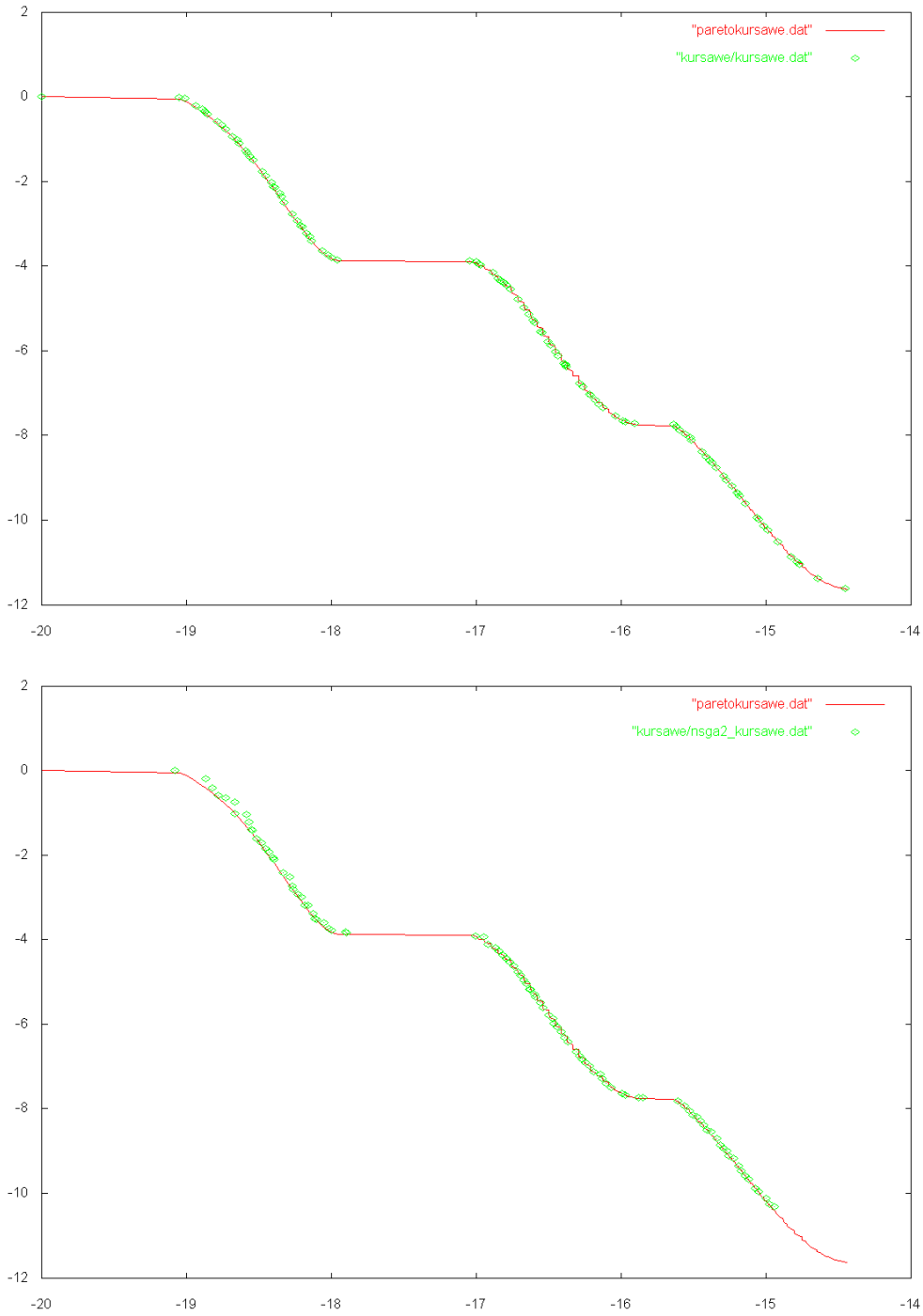

Figura 6.4: Frentes de Pareto generados por MyDE (arriba) y NSGA-II (abajo) para la segunda función de prueba (MOP2).

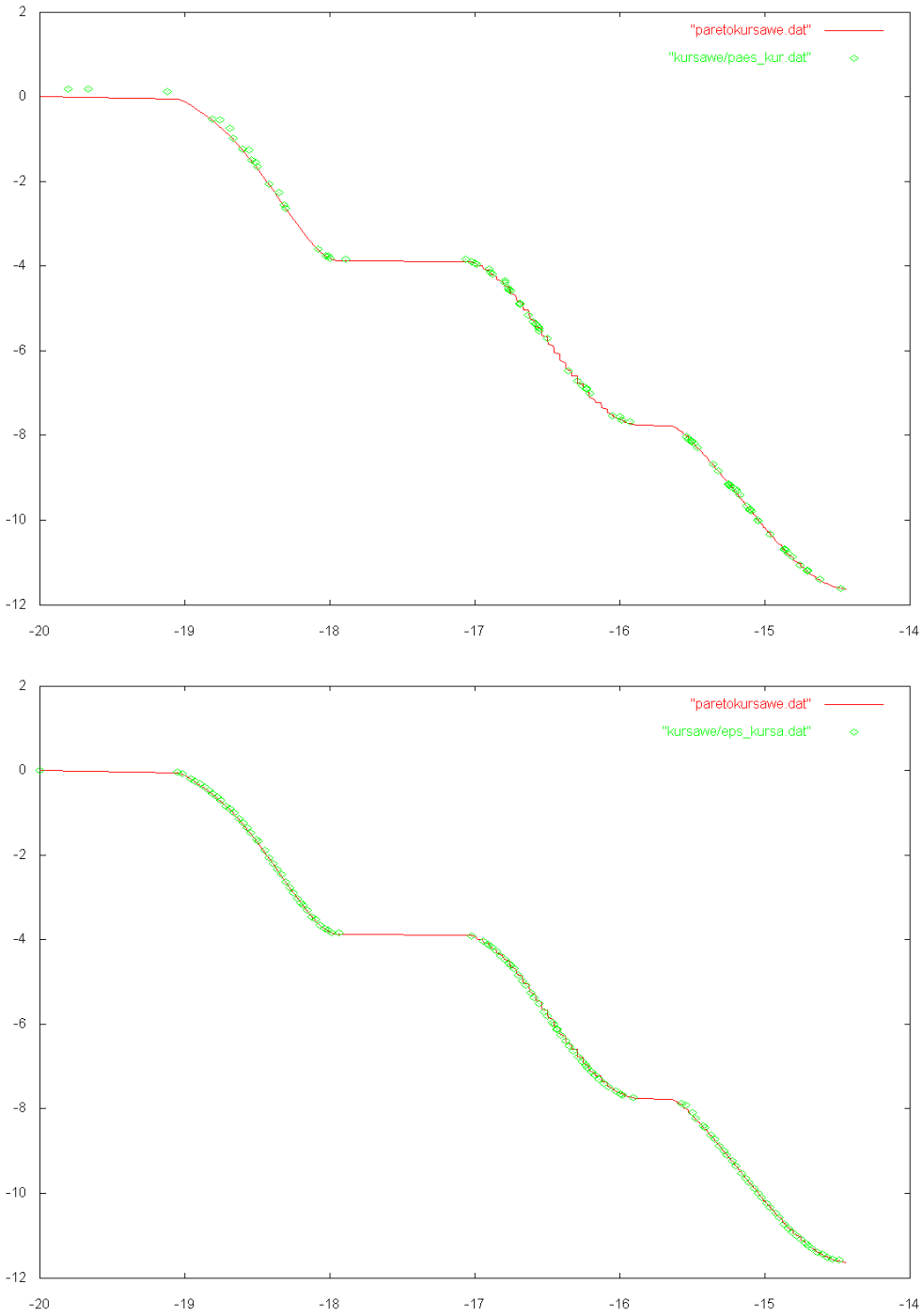

Figura 6.5: Frentes de Pareto generados por PAES (arriba) y ϵ-MyDE (abajo) para la segunda función de prueba (MOP2).

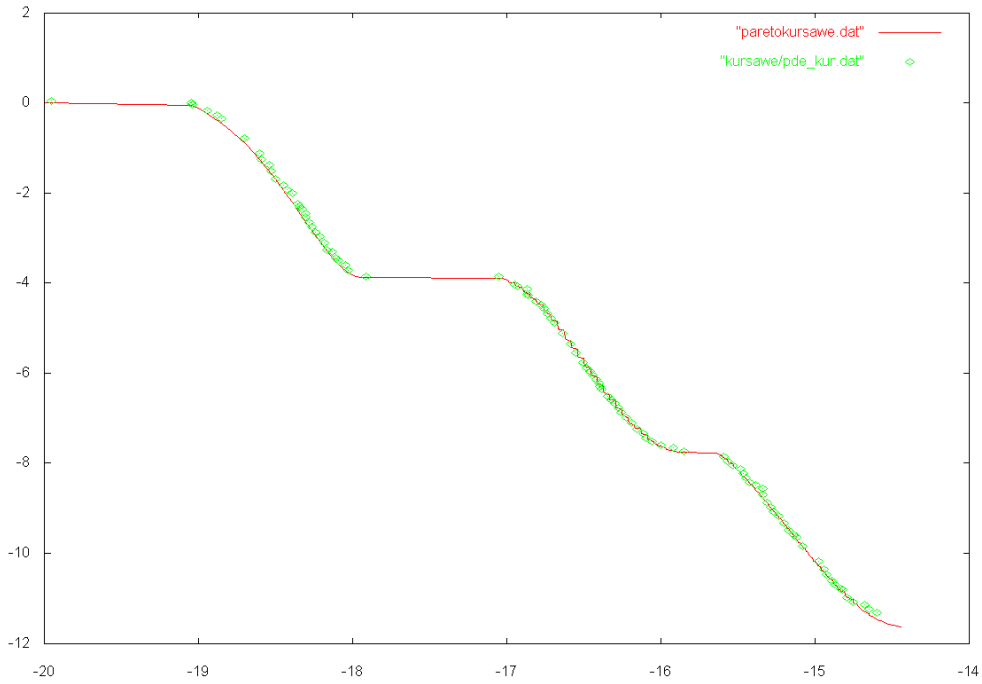

Figura 6.6: Frente de Pareto generado por PDE para la segunda función de prueba (MOP2).

Métrica - Algoritmo		media	mejor	peor	desv. Estándar
TE	MyDE	**0.0005**	**0**	**0.01**	**0.00223607**
	NSGA-II	0.00288569	0.205945	1	0.07853142
	PAES	0.412	0.11	1	0.25193149
	ϵ-MyDE	0.00184096	0.0506528	0.02	0.00728447
	PDE	0.85	0	1	0.36634755
DG	MyDE	**0.004671**	0.00426992	**0.00512812**	**0.00025457**
	NSGA-II	0.07853142	0.00288569	0.205945	0.09408827
	PAES	0.09439276	0.00342838	0.25918	0.07471209
	ϵ-MyDE	0.00728447	**0.00184096**	0.0506528	0.08512233
	PDE	0.16741538	0.00204314	0.20732	0.05819178
D	MyDE	0.04136209	0.0326072	**0.0510895**	**0.00531609**
	NSGA-II	**0.03992726**	0.0296285	0.0562794	0.00955353
	PAES	0.08321454	**0.0240431**	0.239377	0.0468284
	ϵ-MyDE	0.08512233	0.0319032	0.5076	0.1420395
	PDE	0.05213913	0.0284482	0.0851937	0.01131328

C	MyDE	NSGA-II	PAES	ϵ-MyDE	PDE	Dominan
MyDE		0.527228	0.667327	0.067088	0.94604	0.55192075
NSGA-II	0.0658416		0.573762	0.0413223	0.94604	0.406741475
PAES	0.029703	0.456436		0.0170151	0.917327	0.355120275
ϵ-MyDE	0.116337	0.556931	0.657921		0.952475	**0.570916**
PDE	0.00940594	0.411881	0.314356	0.00729217		0.185733778
Son Dominados	0.055321885	0.488119	0.5533415	**0.033179393**	0.9404705	

Tabla 6.4: Resultados de Tasa de Error (TE), Distancia Generacional (DG), Distribución (D) y Cobertura (C) para el tercer problema (MOP3).

algunos espacios y no cubre la parte baja del frente.

De forma cuantitativa, se utilizan las métricas y se observa que ambos algoritmos propuestos mejoran a los otros 3 algoritmos en TE y en DG. En distribución, todos muestran casi los mismos resultados con una ligera ventaja del NSGA-II.

En cuanto a la métrica de cobertura (C), el que mejores resultados obtuvo fue ϵ-MyDE seguido por MyDE ya que son los que dominan a más individuos de los demás algoritmos y los que menos son dominados por los demás algoritmos.

6.2.3. MOP3

Las gráficas correspondientes al MOP3 se muestran en las figuras 6.7, 6.8, 6.9 y el resultado estadístico de la aplicación de las métricas se muestra en la tabla 6.4.

Con esta función, las gráficas muestran que PDE no se acerca siempre al verdadero frente, al igual que NSGA-II (aunque en la gráfica sí llegó, no

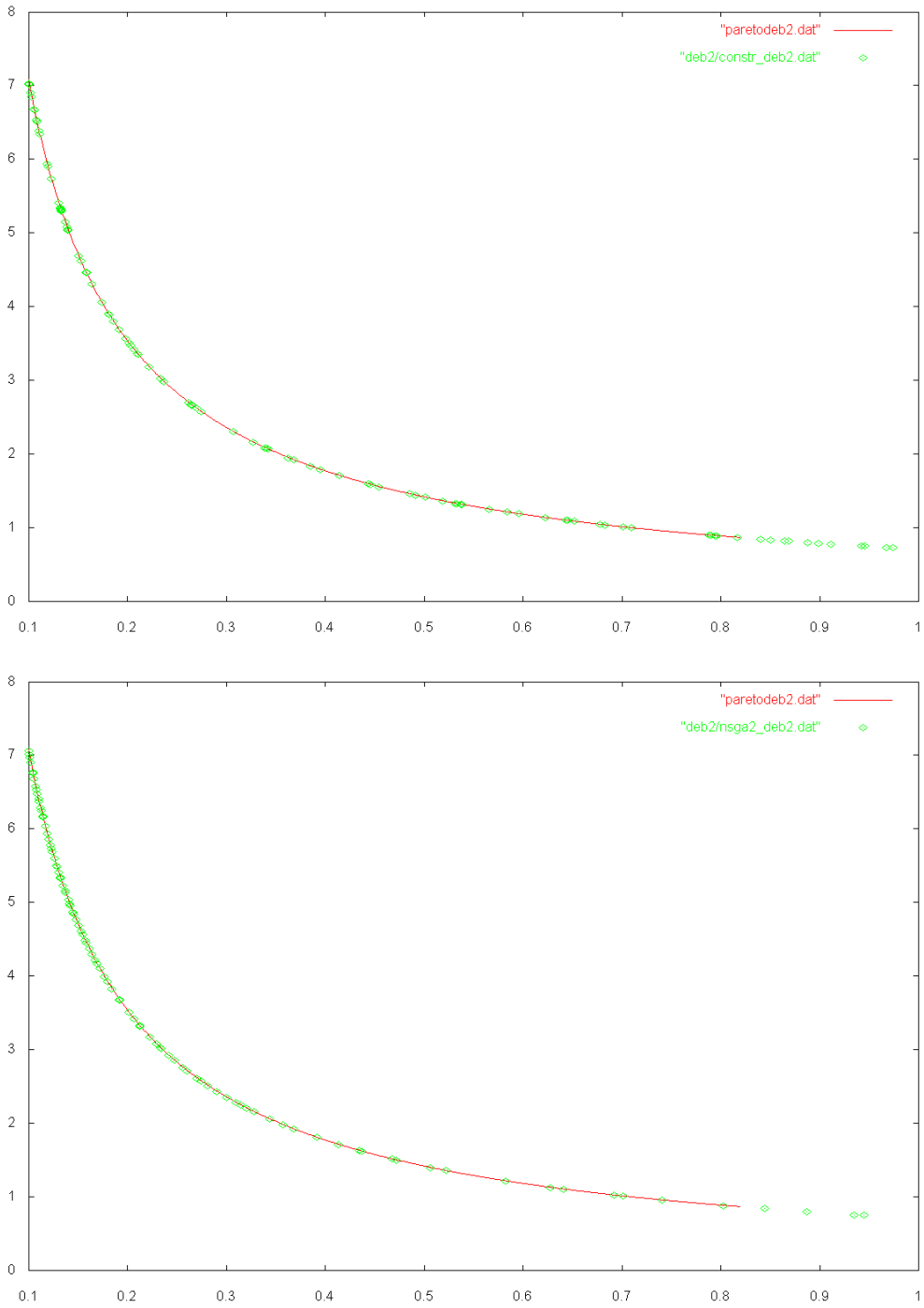

Figura 6.7: Frentes de Pareto generados por MyDE (arriba) y NSGA-II (abajo) para la tercer función de prueba (MOP3).

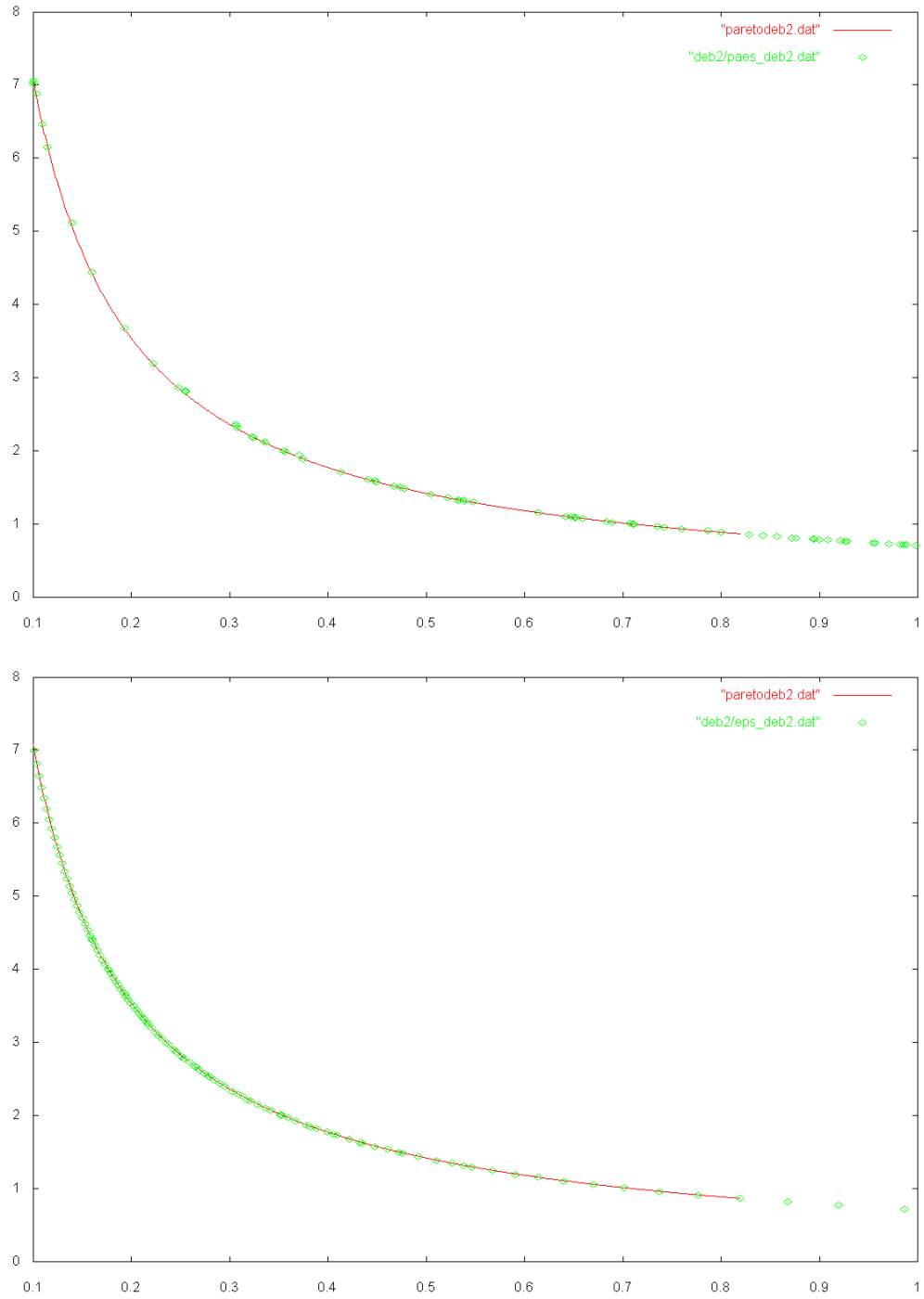

Figura 6.8: Frentes de Pareto generados por PAES (arriba) y ϵ-MyDE (abajo) para la tercer función de prueba (MOP3).

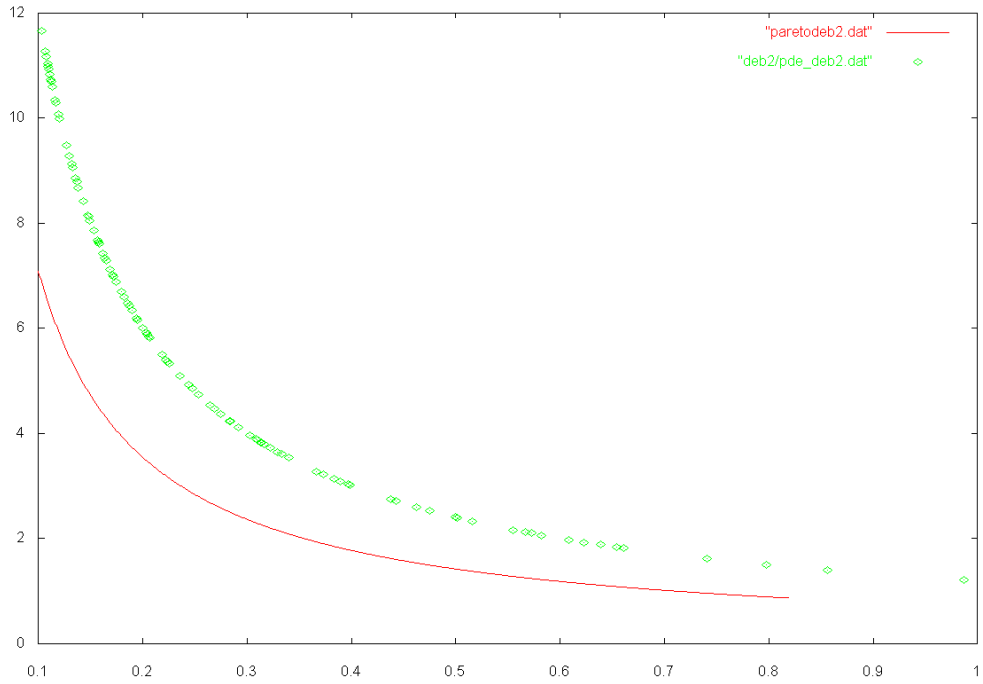

Figura 6.9: Frente de Pareto generado por PDE para la tercer función de prueba (MOP3).

Métrica - Algoritmo		media	mejor	peor	desv. Estándar
TE	MyDE	**0.027**	0.01	**0.07**	0.01657519
	NSGA-II	0.402	0.22	0.77	0.12262909
	PAES	0.2995	0.15	0.52	0.11038092
	ϵ-MyDE	0.0303365	**0**	0.0813953	**0.02042603**
	PDE	0.3455	0.25	0.44	0.04559259
DG	MyDE	0.00115992	0.00060329	0.00167614	0.00030229
	NSGA-II	0.0014961	0.00046401	0.00358142	0.00083619
	PAES	0.00059576	0.000336	0.00105481	0.00020763
	ϵ-MyDE	0.0091616	0.0205025	0.00332202	0.01277354
	PDE	**0.00036064**	**0.00026505**	**0.00045818**	**4.6411E-05**
D	MyDE	0.00863301	0.00708468	0.012076	**0.00138018**
	NSGA-II	**0.00652123**	**0.00266103**	**0.0106271**	0.0023083
	PAES	0.01284339	0.00270814	0.0235331	0.00501697
	ϵ-MyDE	0.01277354	0.0091616	0.0205025	0.00338974
	PDE	0.00798303	0.00473584	0.0134835	0.00258773

C	MyDE	NSGA-II	PAES	ϵ-MyDE	PDE	Dominan
MyDE		0.684653	0.558692	0.181305	0.635644	0.5150735
NSGA-II	0.125248		0.43685	0.101695	0.505941	0.2924335
PAES	0.0905941	0.567327		0.0837185	0.523762	0.3163504
ϵ-MyDE	0.202475	0.678713	0.589401		0.671287	**0.535469**
PDE	0.0717822	0.551485	0.463596	0.0780688		0.291233
Son Dominados	0.122524825	0.6205445	0.51213475	**0.111196825**	0.5841585	

Tabla 6.5: Resultados de Tasa de Error (TE), Distancia Generacional (DG), Distribución (D) y Cobertura (C) para el cuarto problema a resolver.

en todas las corridas llega al frente de Pareto verdadero). Los algoritmos que muestran una mejor distribución son ϵ-MyDE y NSGA-II.

De forma cuantitativa, se utilizan las métricas y se observa que ambos algoritmos propuestos mejoran a los otros 3 algoritmos en TE y en DG, aunque en Distribución el NSGA-II muestra mejores resultados.

En cuanto a la métrica de cobertura (C) en la que se comparan directamente los algoritmos, el que mejores resultados obtuvo fue ϵ-MyDE seguido por MyDE, ya que son los que dominan a más individuos de los demás algoritmos y los que menos son dominados por ellos.

6.2.4. MOP4

Las gráficas correspondientes al MOP4 se muestran en las figuras 6.10, 6.11, 6.12 y el resultado estadístico de la aplicación de las métricas se muestra en la tabla 6.5.

En las gráficas, se ve que MyDE cubre todo el frente de Pareto verdadero y muestra buena distribución; NSGA-II no llega a cubrir todo el frente

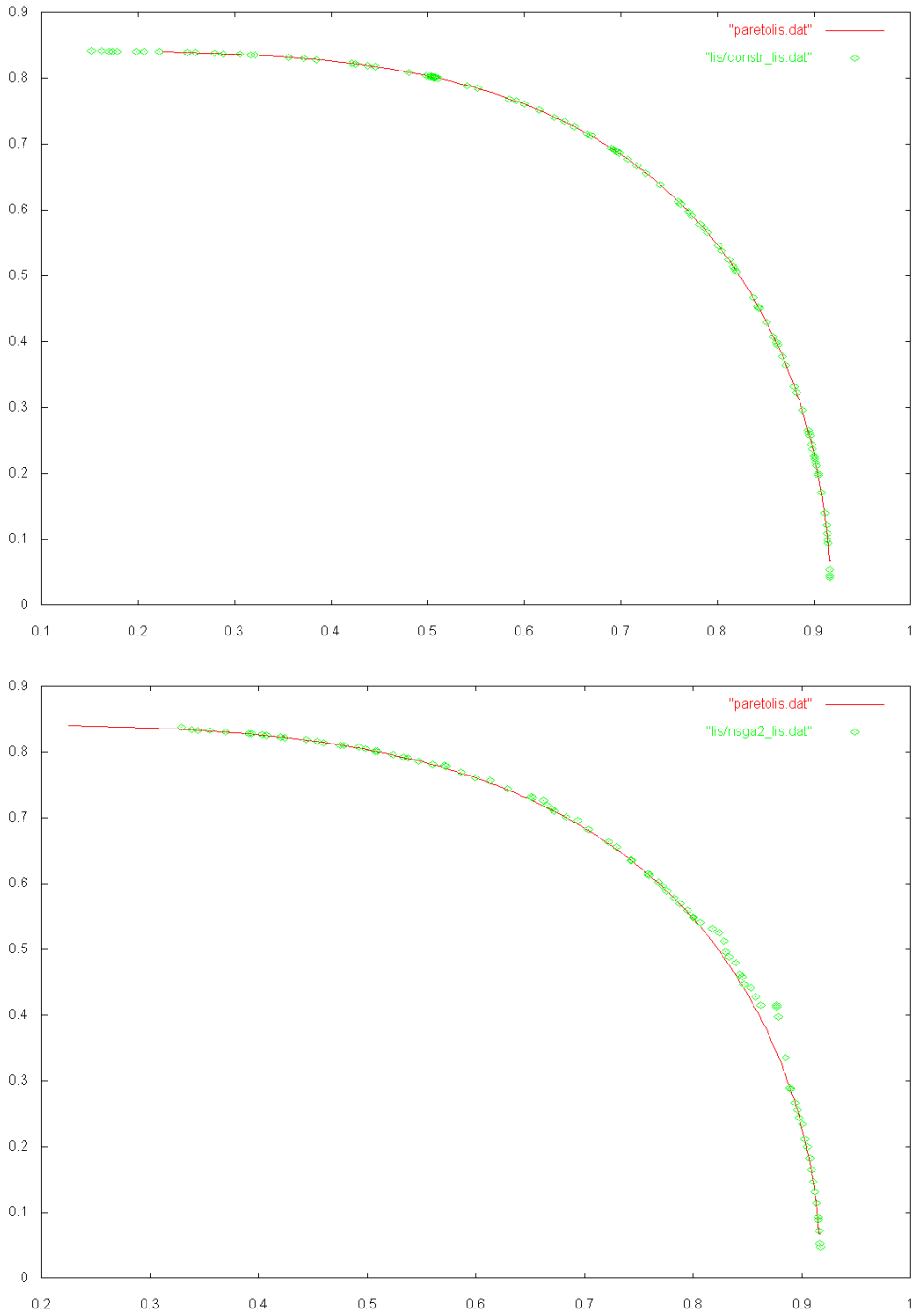

Figura 6.10: Frentes de Pareto generados por MyDE (arriba) y NSGA-II (abajo) para la cuarta función de prueba (MOP4).

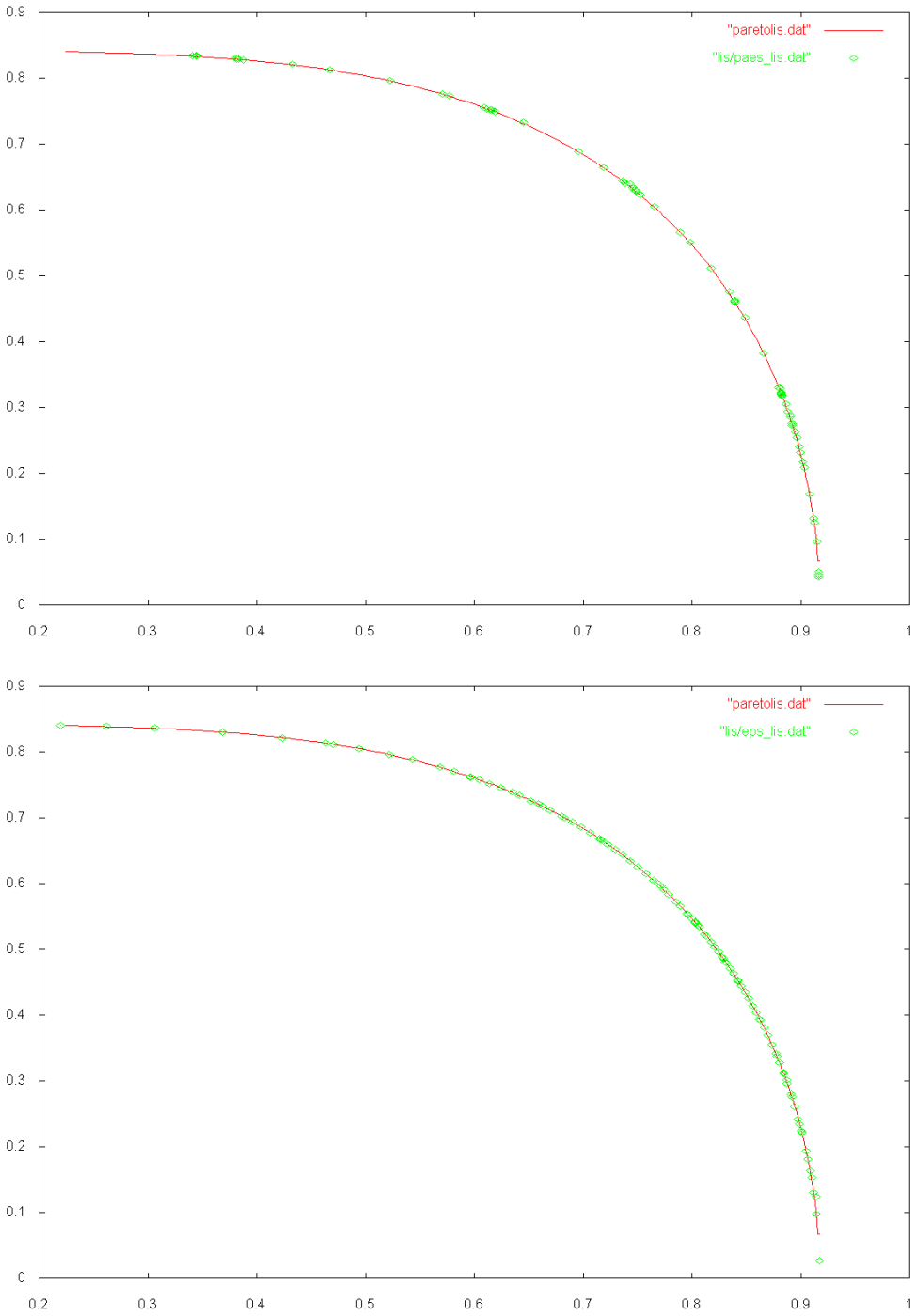

Figura 6.11: Frentes de Pareto generados por PAES (arriba) y ϵ-MyDE (abajo) para la cuarta función de prueba (MOP4).

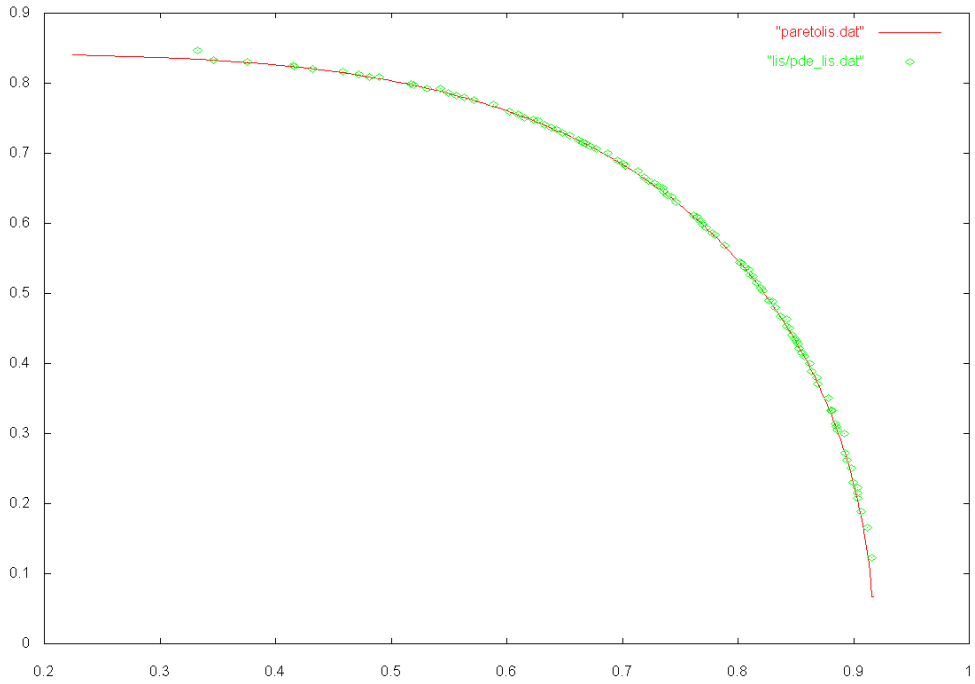

Figura 6.12: Frente de Pareto generado por PDE para la cuarta función de prueba (MOP4).

pues no cubre las orillas y muestra una regular distribución de soluciones. PAES sí cubre el frente pero no tiene una buena distribución además de que no alcanza uno de los extremos. ϵ-MyDE es el que mejor cubre todas las regiones, alcanzando el frente de Pareto verdadero y mostrando una excelente distribución. PDE también se acerca al frente y muestra una buena distribución pero no alcanza los extremos.

De forma cuantitativa, se utilizan las métricas y se observa que ambos algoritmos propuestos mejoran a los otros 3 algoritmos en TE. En DG, casi todos los algoritmos obtienen los mismos resultados, y en Distribución el NSGA-II tiene los mejores resultados.

En cuanto a la métrica de cobertura (C), el que mejores resultados obtuvo fue ϵ-MyDE seguido por MyDE ya que son los que dominan a más individuos de los demás algoritmos y los que menos son dominados por los demás algoritmos.

6.2.5. MOP5

Las gráficas correspondientes al MOP5 se muestran en las figuras 6.13, 6.14, 6.15 y el resultado estadístico de la aplicación de las métricas se muestra en la tabla 6.6.

En las gráficas se ve, por ejemplo, que no todos los algoritmos se acercaron al verdadero frente, pues existen varios puntos que no se encuentran cerca del verdadero frente. El mejor frente es el de NSGA-II con una buena distribución y acercándose mucho al frente de Pareto verdadero. MyDE y ϵ-MyDE también tienen buena distribución aunque no se acercan del todo al frente, y el PAES y el PDE se quedan más lejos del frente.

De forma cuantitativa, se utilizan las métricas y se observa que NSGA-II es el que obtiene mejores resultados tanto en TE como en DG. En distribución, todos los algoritmos muestran resultados aceptables con una ligera ventaja del PDE.

En cuanto a la métrica de cobertura (C) en la que se comparan directamente los algoritmos, el que mejor resultados obtuvo fue NSGA-II seguido por y ϵ-MyDE ya que son los que cubren en mayor proporción a los demás algoritmos.

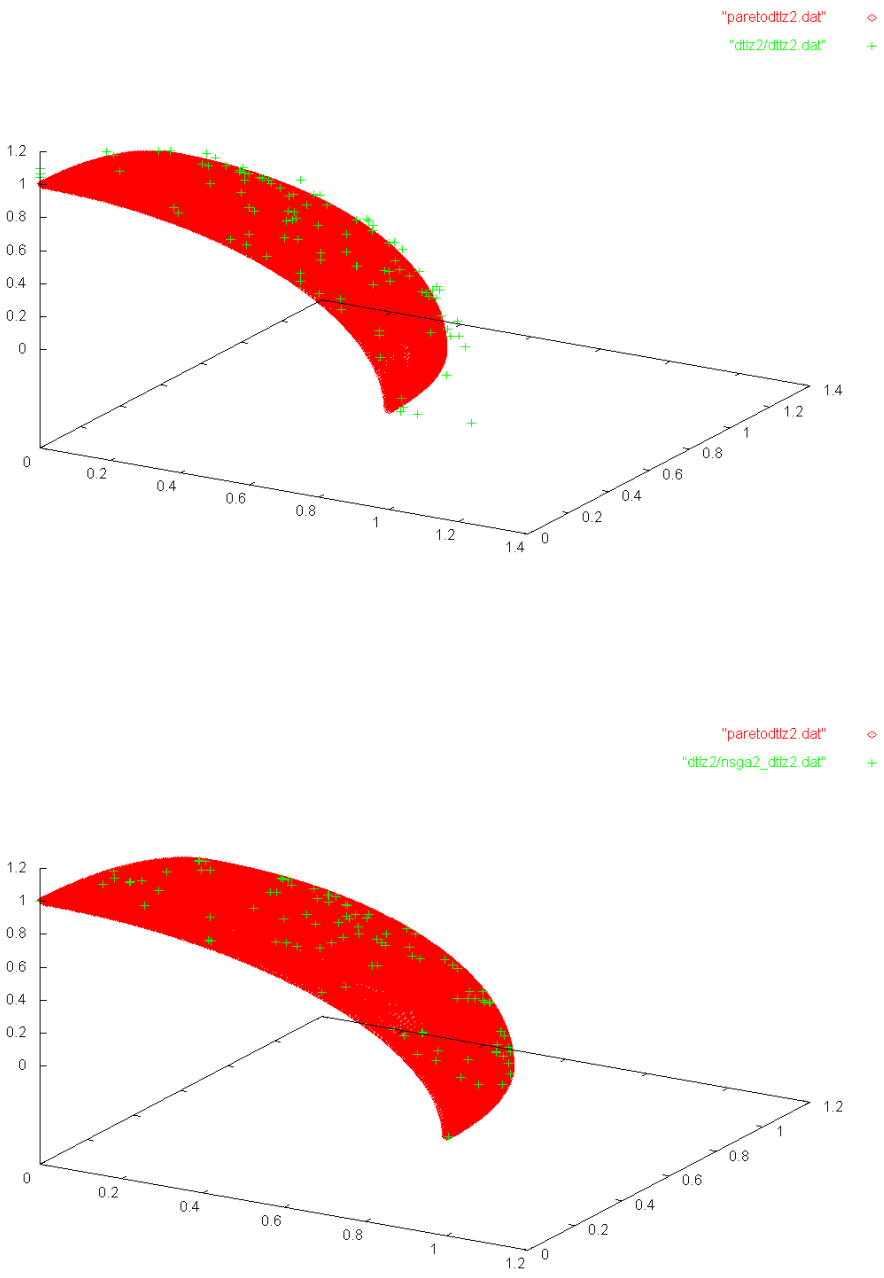

Figura 6.13: Frentes de Pareto generados por MyDE (arriba) y NSGA-II (abajo) para la quinta función de prueba (MOP5).

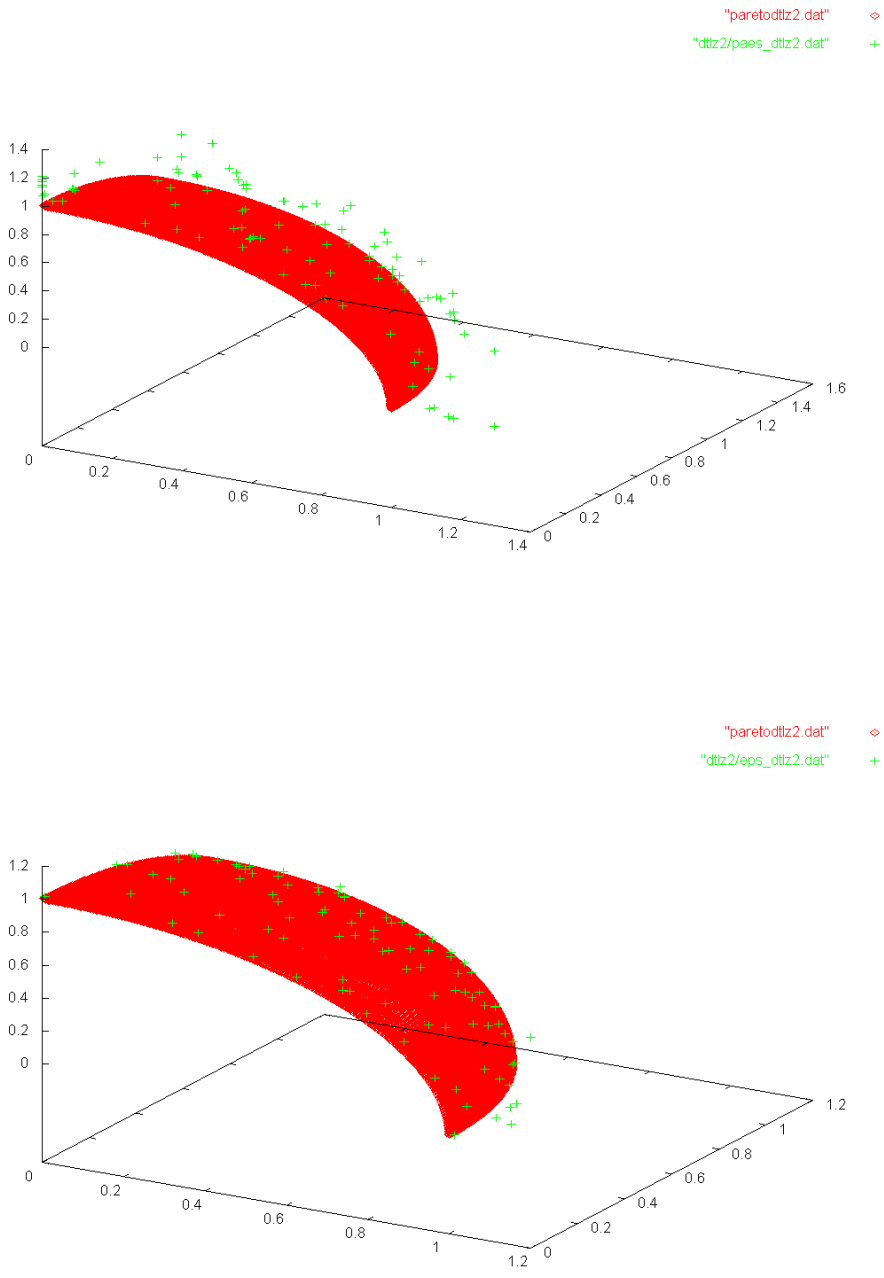

Figura 6.14: Frentes de Pareto generados por PAES (arriba) y ϵ-MyDE (abajo) para la quinta función de prueba (MOP5).

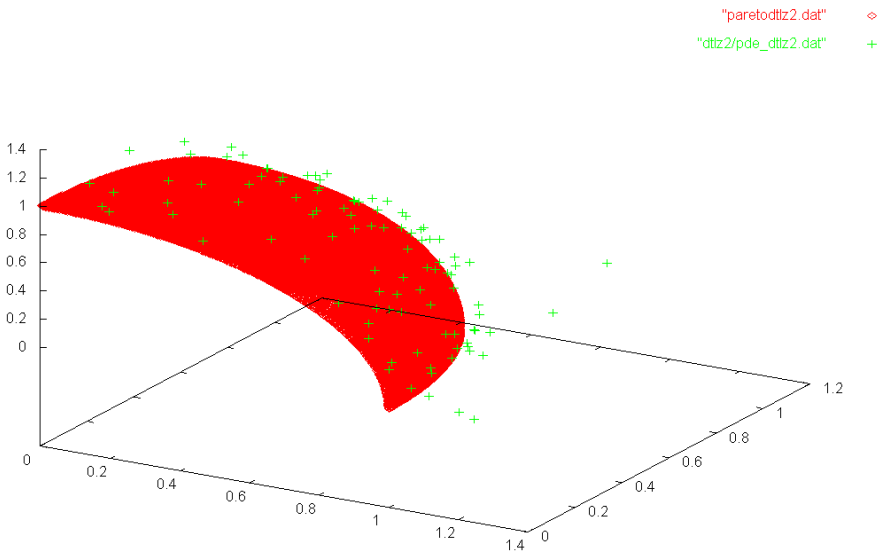

Figura 6.15: Frente de Pareto generado por PDE para la quinta función de prueba (MOP5).

Métrica - Algoritmo		media	mejor	peor	desv. Estándar
TE	MyDE	0.9445	0.89	0.99	0.0248098
	NSGA-II	**0.8115**	**0.24**	0.99	0.23713809
	PAES	0.8235	0.74	**0.93**	0.05008151
	ϵ-MyDE	0.96067525	0.891304	1	0.02852256
	PDE	0.97	0.91	1	**0.024279**
DG	MyDE	0.00932166	0.00388723	0.0172288	0.00424019
	NSGA-II	**0.00063919**	**0.00034657**	**0.00146953**	**0.00028381**
	PAES	0.0147785	0.010867	0.0223971	0.0025757
	ϵ-MyDE	0.01016063	0.00361315	0.0168273	0.00405528
	PDE	0.0068891	0.00549602	0.00878221	0.00104458
D	MyDE	0.07103481	0.0563019	0.116349	0.01384389
	NSGA-II	0.06109077	0.0535216	**0.0732101**	0.00611728
	PAES	0.0800341	0.073786	0.0960632	**0.00595046**
	ϵ-MyDE	0.08257325	0.0484829	0.125147	0.02383408
	PDE	**0.05093182**	**0.00748259**	0.111133	0.03242853

C	MyDE	NSGA-II	PAES	ϵ-MyDE	PDE	Dominan
MyDE		0.00049505	0.8737	0.167887	0.692079	0.433540263
NSGA-II	0.785644		0.94106	0.406904	0.919307	**0.76322875**
PAES	0.10495	0		0.0910042	0.349505	0.1363648
ϵ-MyDE	0.503465	0.0267327	0.770183		0.74505	0.511357675
PDE	0.239109	0.00049505	0.70629	0.144874		0.272692013
Son Dominados	0.408292	**0.0069307**	0.82280825	0.2026673	0.67648525	

Tabla 6.6: Resultados de Tasa de Error (TE), Distancia Generacional (DG), Distribución (D) y Cobertura (C) para el quinto problema (MOP5).

6.3. Conclusiones sobre los resultados en problemas sin restricciones

De forma general, se puede observar en base a los resultados que MyDE obtiene en todas las métricas y gráficamente que es un algoritmo competitivo, pues siempre se acerca al frente de Pareto verdadero, y en la mayoría de los problemas (MOP1, MOP2, MOP3 y MOP4) lo cubre con una buena distribución. Aunque en la métrica de distribución el que mejores resultados obtuvo en general fue el NSGA-II, también es cierto que pierde mucho los extremos del frente además de que no cubre todas las regiones del verdadero frente. ϵ-MyDE al igual que MyDE obtiene muy buenos resultados en todos los problemas (aunque no supera a todos los algoritmos en todos los problemas) pero su desempeño es muy consistente, además de presentar frentes con buena distribución y no deja espacios grandes sin cubrir.

En el problema MOP5, MyDE y ϵ-MyDE siguen obteniendo buenos resultados aunque el NSGA-II es el que se desempeña mejor, pues se acerca mucho mejor al verdadero frente logrando una muy buena distribución aunque el tiempo de ejecución se incrementa mucho debido a la dimensionalidad del problema.

6.4. Funciones de prueba con restricciones

Procedemos ahora a validar los algoritmos propuestos en problemas con restricciones.

MOPC1. Este problema fue propuesto por Kita [26]

Maximizar:

$$\vec{f}(x) = (f_1(x,y), f_2(x,y))$$

donde:

$$\begin{aligned} f_1(x,y) &= -x^2 + y, \\ f_2(x,y) &= \frac{1}{2}x + y + 1 \end{aligned}$$

sujeto a:

$$\begin{aligned} 0 &\geq \frac{1}{6}x + y - \frac{13}{2}, \\ 0 &\geq \frac{1}{2}x + y - \frac{15}{2}, \\ 0 &\geq 5x + y - 30 \end{aligned}$$

y:

$$0 \leq x, y$$

Este problema tiene un frente de Pareto desconectado y cóncavo, y su conjunto de óptimos de Pareto también está desconectado, Adicionalmente, el conjunto de óptimos de Pareto se encuentra en el límite entre la zona factible y la no factible.

MOPC2. Este problema fue propuesto por Osyczka [32]

Minimizar:

$$\vec{f}(x) = (f_1(\vec{x}), f_2(\vec{x}))$$

donde:

$$\begin{aligned} f_1(\vec{x}) &= -\{25(x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 1)^2 + (x_4 - 4)^2 + (x_5 - 1)^2\}, \\ f_2(\vec{x}) &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \end{aligned}$$

sujeto a:

$$\begin{aligned} 0 &\leq x_1 + x_2 - 2, \\ 0 &\leq 6 - x_1 - x_2, \\ 0 &\leq 2 - x_2 + x_1, \\ 0 &\leq 2 - x_1 + 3x_2, \\ 0 &\leq 4 - (x_3 - 3)^2 - x_4, \\ 0 &\leq (x_5 - 3)^2 + x_6 - 4 \end{aligned}$$

y:

$$\begin{aligned} 0 &\leq x_1, x_2, x_6 \leq 10, \\ 0 &\leq x_3, x_5 \leq 5, \\ 0 &\leq x_4 \leq 6 \end{aligned}$$

Este problema posee un frente de Pareto desconectado, y su conjunto de óptimos de Pareto también está desconectado, por lo que se considera un problema muy difícil de resolver. El conjunto de óptimos de Pareto tiene 5 regiones desconectadas, lo cual requiere una buena capacidad de exploración por parte del algoritmo.

MOPC3. Este problema fue propuesto por Golinski en [24]

Su objetivo es encontrar el mínimo del volumen de una caja de velocidades, y también el peso mínimo, sujeto a varias restricciones. El problema se muestra en la Figura 6.16 y tiene 7 variables que representan:

x_1 ancho de la cara del engrane, cm

x_2 módulo de diente, cm

x_3 número de dientes en el engrane

x_4 longitud del eje 1 entre los cojinetes, cm

x_5 longitud del eje 2 entre los cojinetes, cm

x_6 diámetro del eje 1, cm

x_7 diámetro del eje 2, cm

Matemáticamente el problema está dado por:

Minimizar:

$$\vec{f}(x) = (f_1(\vec{x}), f_2(\vec{x}))$$

donde:

$$\begin{aligned} f_1(\vec{x}) &= 0{,}7854x_1x_2^2(3{,}3333x_3^2 + 14{,}9334x_3 - 43{,}0934) - \\ &\quad 1{,}5079(x_6^2 + x_7^2)x_1 + 7{,}477(x_6^3 + x_7^3) + 0{,}7854(x_4x_6^2 + x_5x_7^2) \\ f_2(\vec{x}) &= \frac{\sqrt{(\frac{745 \cdot x_4}{x_2x_3})^2 + 1{,}69e^7}}{0{,}1x_6^3} \end{aligned}$$

sujeto a:

$$
\begin{aligned}
0 &\geq \frac{1}{x_1 \cdot x_2^2 \cdot x_3} - \frac{1}{27}, \\
0 &\geq \frac{1}{x_1 \cdot x_2^2 \cdot x_3^2} - \frac{1}{397{,}5}, \\
0 &\geq \frac{x_3^3}{x_2 \cdot x_3 \cdot x_6^4} - \frac{1}{1{,}93}, \\
0 &\geq \frac{x_5^3}{x_2 \cdot x_3 \cdot x_7^4} - \frac{1}{1{,}93}, \\
0 &\geq x_2 \cdot x_3 - 40, \\
0 &\geq \frac{x_1}{x_2} - 12, \\
0 &\geq 5 - \frac{x_1}{x_2}, \\
0 &\geq 1{,}9 - x_4 + 1{,}5x_6, \\
0 &\geq 1{,}9 - x_5 + 1{,}1x_7, \\
1300 &\geq f_2, \\
1100 &\geq \frac{\sqrt{(\frac{745 \cdot x_5}{x_1 \cdot x_2})^2 + 1{,}575e^8}}{0{,}1 \cdot x_6^3}
\end{aligned}
$$

y:

$$
\begin{aligned}
2{,}6 &\leq x_1 \leq 3{,}6, \\
0{,}7 &\leq x_2 \leq 0{,}8, \\
17 &\leq x_3 \leq 28 \\
7{,}3 &\leq x_4, x_5 \leq 8{,}3, \\
2{,}9 &\leq x_6 \leq 3{,}9, \\
5{,}0 &\leq x_7 \leq 5{,}5
\end{aligned}
$$

Este problema tiene una dimensionalidad grande en el número de variables y restricciones, lo que hace que también sea más costoso (computacionalmente hablando) el evaluar la función de aptitud. Se calcula que si se realiza el cálculo del frente de Pareto verdadero por enumeración, dividiendo cada variable en 100 partes iguales, se espera que el proceso tarde en un solo procesador (P4 a 2.4 GHz y 512 RAM) más de 600 días.

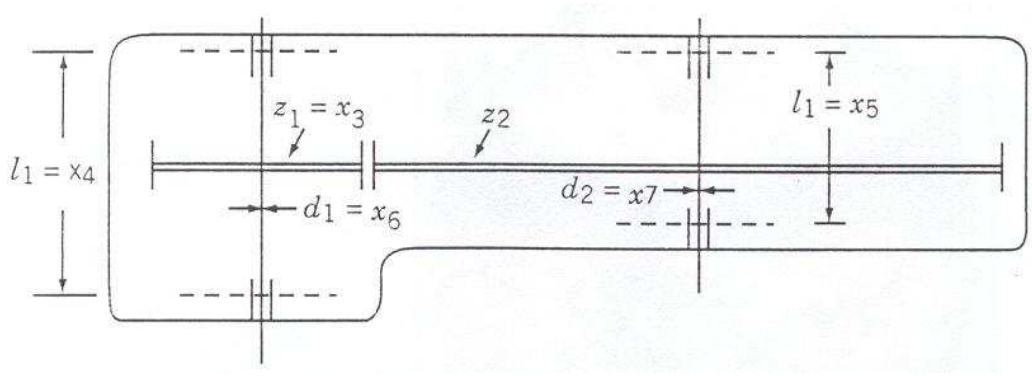

Figura 6.16: Ilustración del problema MOPC3

MOPC4. Este problema fue propuesto por Tamaki en [41]

Maximizar:

$$\vec{f}(x) = (f_1(x,y,z), f_2(x,y,z), f_3(x,y,z))$$

donde:

$$\begin{aligned} f_1(x,y,z) &= x, \\ f_2(x,y,z) &= y, \\ f_3(x,y,z) &= z \end{aligned}$$

sujeto a:

$$0 < x^2 + y^2 + z^2$$

y:

$$0 \leq x, y, z \leq 1$$

Este problema posee un frente de Pareto conectado y el cual forma una superficie curva. Su conjunto de óptimos de Pareto también forma una superficie curva. Se consideró la inclusión de este problema para observar el comportamiento de los algoritmos en problemas con más de 2 funciones objetivo, pues en este caso se producen más soluciones no dominadas, además de ser mayor el espacio de búsqueda.

MOPC5. Este problema fue propuesto en [35].

Una viga necesita ser soldada con otra y debe llevar una carga de F = 6000 lb la cual se aplica en un extremo de la misma. El objetivo del diseño es reducir al mínimo el costo de fabricación y reducir al mínimo la desviación final. El problema se ilustra en la figura 6.17 y la expresión matemática es la siguiente:

Minimizar:

$$\vec{f}(x) = (f_1(\vec{x}), f_2(\vec{x}))$$

donde:

$$\begin{aligned} f_1(\vec{x}) &= 1{,}10471h^2l + 0{,}04811tb(14{,}0 + l), \\ f_2(\vec{x}) &= \delta(\vec{x}) \end{aligned}$$

sujeto a:

$$\begin{aligned} g_1(\vec{x}) &\equiv 13600 - \tau(\vec{x}) \geq 0, \\ g_2(\vec{x}) &\equiv 30000 - \sigma(\vec{x}) \geq 0, \\ g_3(\vec{x}) &\equiv b - h \geq 0, \\ g_4(\vec{x}) &\equiv P_c - 6000 \geq 0. \end{aligned}$$

La deflección está dada por $\delta((\vec{x})$:

$$\delta(\vec{x}) = \frac{2{,}1952}{t^3 b}$$

La primera restricción se cerciora de que la tensión desarrollada en el soporte de la viga sea menor que la fuerza permisible del material (13,600 PSI).

La segunda restricción se cerciora de que la tensión normal desarrollada en el soporte de la viga sea menor que la fuerza permisible del material de la viga (30,000 PSI).

La tercera restricción se cerciora de que el grueso de la viga no sea menor que el grueso del extremo de la viga soldada.

La cuarta restricción se cerciora de que la carga permisible (a lo largo de la dirección de t) de la viga sea mayor que la carga aplicada.

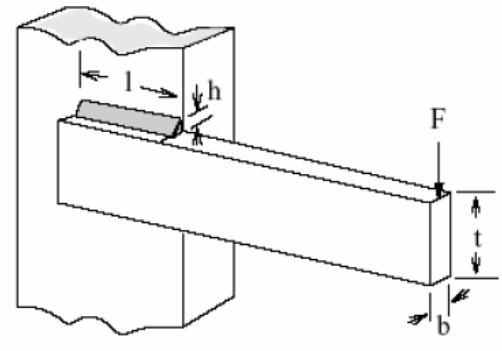

Figura 6.17: Ilustración del problema MOPC5

Las constantes se definen de la siguiente manera:

$$\begin{aligned}
\tau(\vec{x}) &= \sqrt{\frac{(\tau')^2 + (\tau'')^2 + (l\tau'\tau'')}{\sqrt{0{,}25(l^2 + (h+t)^2)}}}, \\
\tau' &= \frac{6000}{\sqrt{2}hl}, \\
\tau'' &= \frac{6000(14 + 0{,}5l)\sqrt{0{,}25(l^2 + (h+t)^2)}}{2\{0{,}707hl(l^2/12 + 0{,}25(h+t)^2)\}}, \\
\sigma(\vec{x}) &= \frac{504000}{t^2 b}, \\
P_c(\vec{x}) &= 64746{,}022(1 - 0{,}0282346t)tb^3.
\end{aligned}$$

Las variables son inicializadas dentro del siguiente rango:

$$\begin{aligned}
0{,}125 &\leq h, b \leq 5{,}0, \\
0{,}1 &\leq l, t \leq 10{,}0.
\end{aligned}$$

Este problema posee un frente de Pareto convexo, pero es multifrontal (o sea, tiene varios frentes de Pareto falsos), por lo que algunos algoritmos pueden quedar atrapado en un frente distinto del verdadero.

Parámetro	MyDE	NSGA-II	PAES	e-MyDE	PDE
P	100	100	100	100	100
NP	100	100	100	100 (auto)	100
G_{max}	250	250	250	250	250
P_c	0.95	0.8	nd	0.95	nd
P_m	1/N	1/N	0.05	1/N	nd
F	0.5	nd	nd	0.5	N(0, 1)
elitismo	0.7	nd	nd	0.7	nd
malla	20^d	nd	2^{d+8}	nd	nd

donde: N = número de variables
d = número de funciones
nd = no requiere parámetro

Tabla 6.7: Parámetros utilizados en los diversos algoritmos para realizar las pruebas y comparaciones.

6.5. Resultados a los problemas con restricciones

Los parámetros utilizados para obtener los resultados que se muestran más adelante para cada uno de los distintos algoritmos se muestran en la tabla 6.7.

Casi todos los parámetros quedaron exactamente igual que los utilizados en los problemas sin restricciones, excepto por G_{max} (número de generaciones) que cambió de 50 a 250. Este cambio se debe a que los problemas con restricciones son más complejos y se requiere un número mayor de evaluaciones para encontrar una buena aproximación de frente de Pareto. Además, se movió el parámetro $elitismo$, el cual cambió de 0.2 a 0.7, pues en estas funciones necesita una mucho mayor exploración que la explotación, por la dificultad de los mismos.

A continuación se muestran las gráficas que comparan el frente de Pareto verdadero con los resultados de nuestra técnica y los distintos algoritmos adoptados en cada una de las funciones de prueba con restricciones. El manejo de restricciones para aquellos algoritmos que no lo proporcionan un mecanismo explícito (PAES y PDE) en su versión original, se les implementó un manejo muy similar al que se maneja con la propuesta de esta tesis(ver sección 5.1.6). Los frentes de Pareto verdaderos fueron generados por enumeración y las ejecuciones mostradas fueron realizadas con semillas de números aleatorios generadas al azar.

Junto con las gráficas, se muestran dos tablas:

1. la primera corresponde a los resultados estadísticos de las métricas

Métrica - Algoritmo		media	mejor	peor	desv. Estándar
TE	MyDE	0.633	0.44	0.88	0.10120849
	NSGA-II	0.316	0.19	0.53	0.09394287
	PAES	0.745	0.53	1	0.14099272
	e-MyDE	**0.01640604**	**0**	**0.0550459**	**0.01635306**
	PDE	0.8395	0.35	1	0.21211901
DG	MyDE	0.01691935	0.00269066	0.133402	0.03088439
	NSGA-II	0.01470336	0.00136745	0.0902511	0.02729836
	PAES	0.06363661	0.00172828	0.384466	0.09234686
	e-MyDE	**0.00491203**	0.0014577	0.0464165	0.0097998
	PDE	0.00576845	**0.0003873**	**0.032941**	**0.00769517**
D	MyDE	0.16569178	0.0461003	1.29876	0.29770626
	NSGA-II	0.0925237	0.00705687	0.56773	0.1688492
	PAES	0.25630758	0.0352046	3.11293	0.67529755
	e-MyDE	0.07363313	0.0394301	0.492726	0.09905315
	PDE	**0.01728939**	**3.6065E-05**	**0.2736**	**0.06115304**

C	MyDE	NSGA-II	PAES	ϵ-MyDE	PDE	Dominan
MyDE		0.444554	0.8	0.0304183	0.897634	0.54315158
NSGA-II	0.799801		0.766832	0.0427757	0.666667	0.56901893
PAES	0.690512	0.346535		0.34173	0.83179	0.55264175
ϵ-MyDE	0.948833	0.806436	0.926238		0.928498	**0.90250125**
PDE	0.109786	0.0965347	0.431188	0.00522814		0.16068421
Son Dominados	0.637233	0.42351493	0.7310645	**0.10503804**	0.83114725	

Tabla 6.8: Resultados de Tasa de Error (TE), Distancia Generacional (DG), Distribución (D) y Cobertura (C) para el primer problema, (MOPC1).

correspondientes a 20 ejecuciones de cada algoritmo.

2. la segunda es el resultado de la métrica de Cobertura (C) aplicada a la unión de los 20 archivos de cada algoritmo para obtener 5 archivos (uno por cada algoritmo). Estos 5 archivos se comparan entre ellos utilizando la métrica C, primero realizando C $(alg1, alg2) = Tabla_{alg1,alg2}$ y después C $(alg2, alg1) = Tabla_{alg2,alg1}$. El resultado final se anota en la tabla correspondiente.

6.5.1. MOPC1

Las gráficas correspondientes al MOPC1 se muestran en las figuras 6.18, 6.19, 6.20 y el resultado estadístico de la aplicación de las métricas se muestra en la tabla 6.8.

Hay que enfatizar que en este problema se intentan maximizar las funciones objetivo, por lo que el verdadero frente está por encima de los frentes generados. En las gráficas se ve por ejemplo, que todos los algoritmos

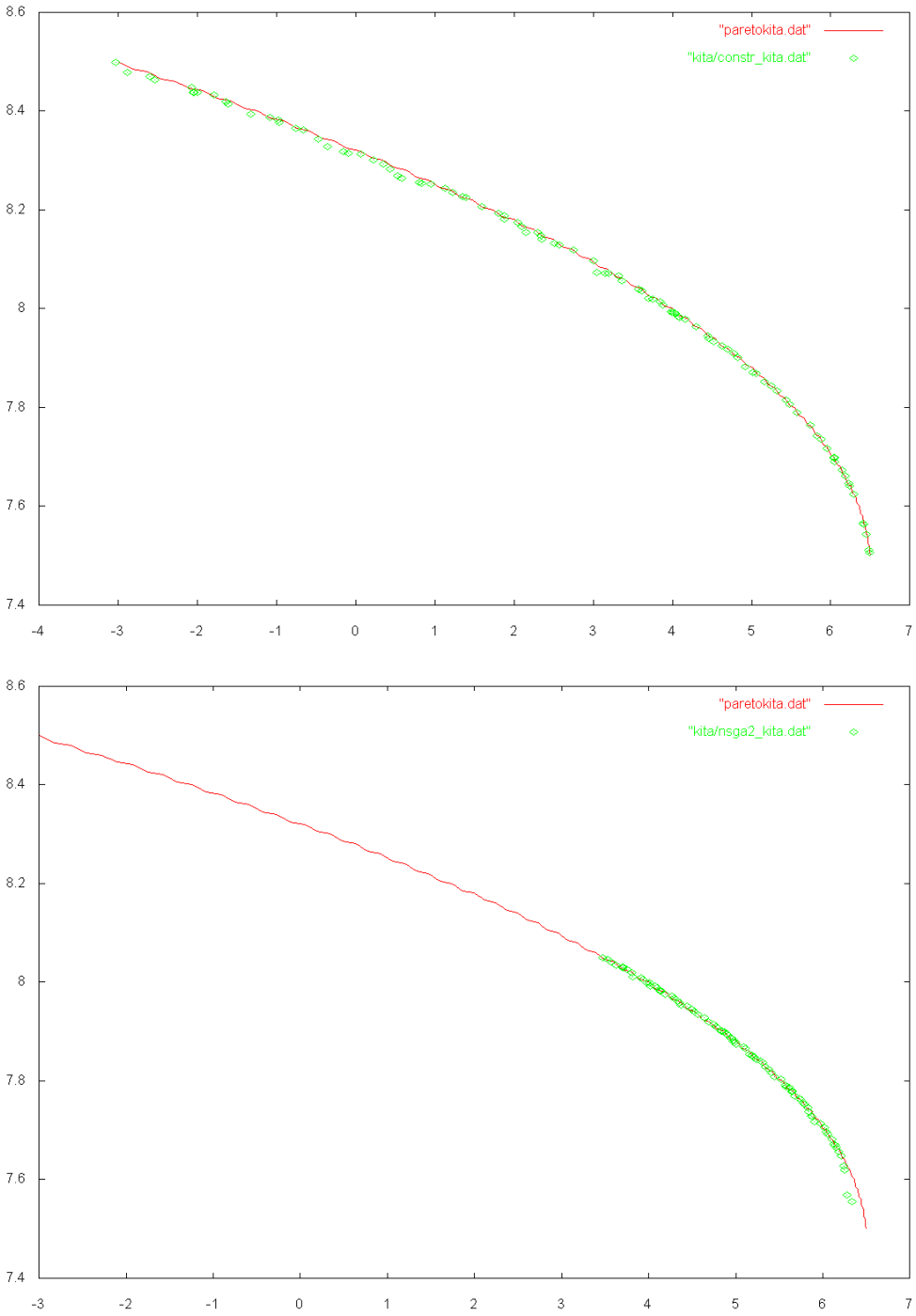

Figura 6.18: Frentes de Pareto generados por MyDE (arriba), NSGA-II (abajo) para la primera función de prueba (MOPC1).

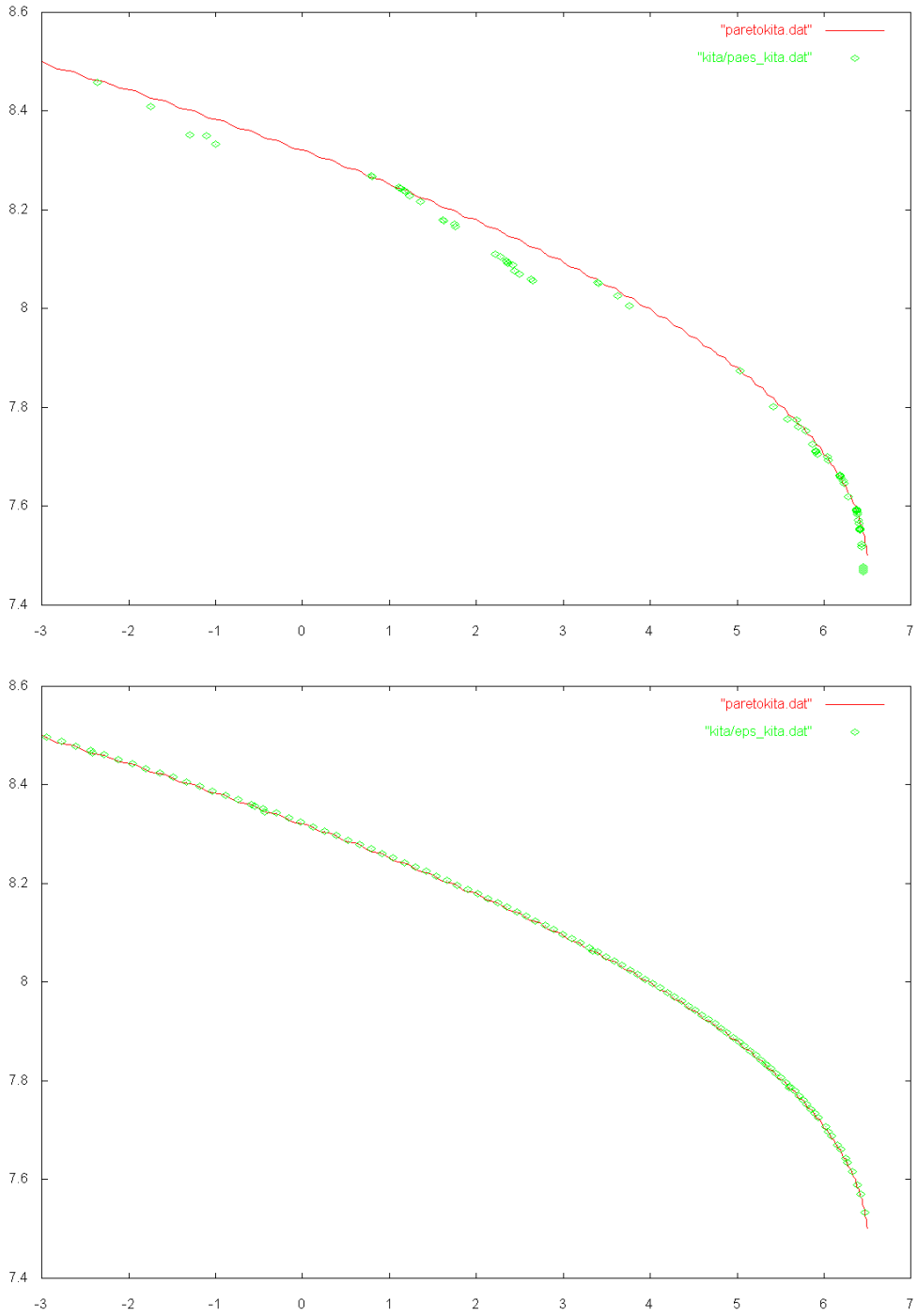

Figura 6.19: Frentes de Pareto generados PAES (arriba), ϵ-MyDE (abajo) para la primera función de prueba (MOPC1).

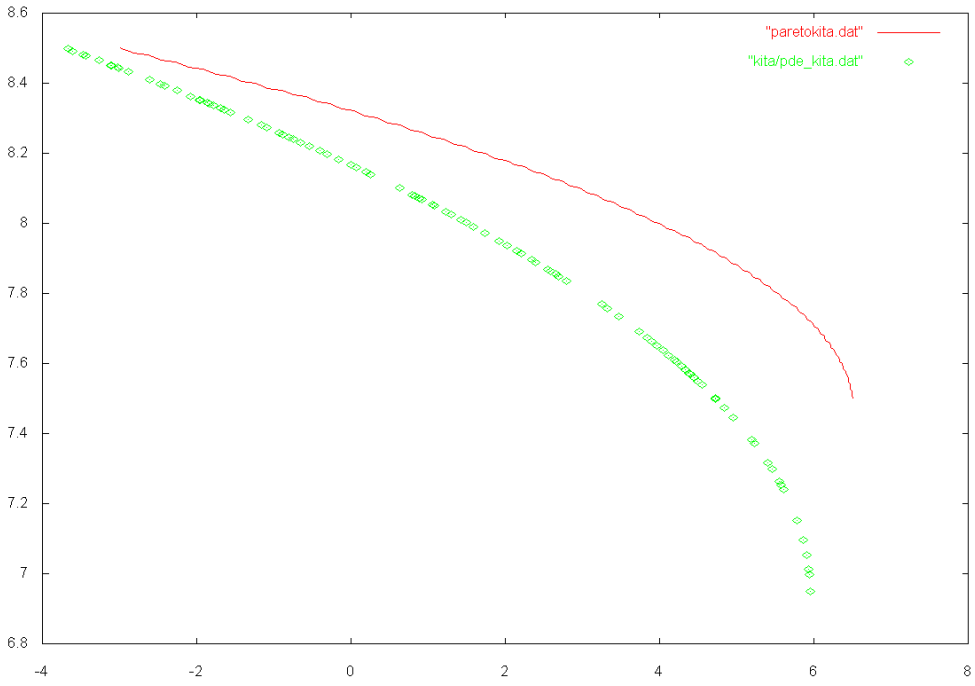

Figura 6.20: Frente de Pareto generado PDE para la primera función de prueba (MOPC1).

se acercaron al verdadero frente, excepto el PDE que se queda atrapado en otra región y no se acerca. De los que llegaron, MyDE cubre todo el frente y muestra una buena distribución aunque existen regiones en las que hay huecos. El NSGA-II aproxima bastante bien el frente de Pareto verdadero mostrando una buena distribución sólo que no cubre el frente en su totalidad, pues se queda en una sola región. PAES sí se acerca a todo el verdadero frente, pero su distribución es muy mala, además de que existen regiones que no cubre. ϵ-MyDE es el algoritmo que mejor se ve, pues cubre bien todo el verdadero frente y su distribución es muy buena.

De forma cuantitativa, se utilizan las métricas y se observa que ϵ-MyDE es el mejor en TE y en DG, aunque en Distribución el PDE muestra los mejores resultados. También podemos concluir que MyDE se acerca mucho al verdadero frente, aunque no lo cubre totalmente (por lo que la métrica TE es alta).

En cuanto a la métrica de cobertura (C) en la que se comparan directamente los algoritmos entre sí, el que mejores resultados obtuvo fue ϵ-MyDE ya que es el que cubre en mayor proporción a los demás algoritmos, además de que es el que menos es cubierto por los demás.

6.5.2. MOPC2

Las gráficas correspondientes al MOPC2 se muestran en las figuras 6.21, 6.22, 6.23 y el resultado estadístico de la aplicación de las métricas se muestra en la tabla 6.9.

En las gráficas se ve que MyDE cubre casi todas las regiones del frente verdadero pero no llega a cubrirlo en su totalidad y su distribución es aceptable. NSGA-II es el que mejor desempeño muestra, pues sí llega a cubrir el frente verdadero y su distribución es muy buena, aunque en algunas ejecuciones no logra converger al frente verdadero. PAES no se acerca mucho al frente verdadero, aunque muestra soluciones en todas sus regiones, y no es muy buena su distribución. ϵ-MyDE se acerca mucho al verdadero frente sin llegar a cubrirlo, además de que no alcanza todas las regiones del frente, aunque su distribución se ve bien. PDE es el que peor se ve, pues no se acerca al verdadero frente ni cubre las regiones del mismo.

De forma cuantitativa, se utilizan las métricas y se observa que NSGA-II es el mejor en TE y ϵ-MyDE en DG, esto significa que el primero cuando

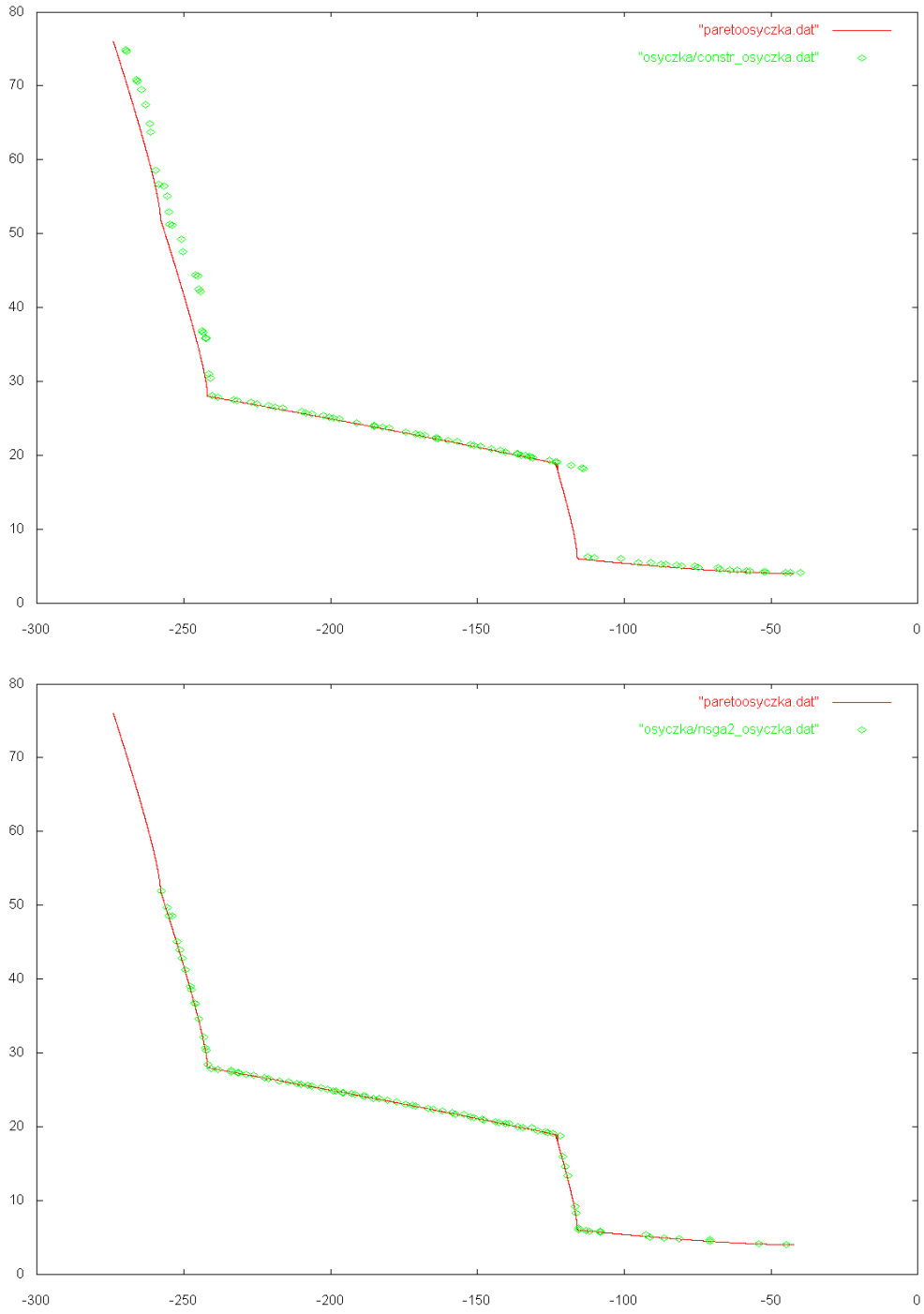

Figura 6.21: Frentes de Pareto generados por MyDE (arriba), NSGA-II (abajo) para la segunda función de prueba (MOPC2).

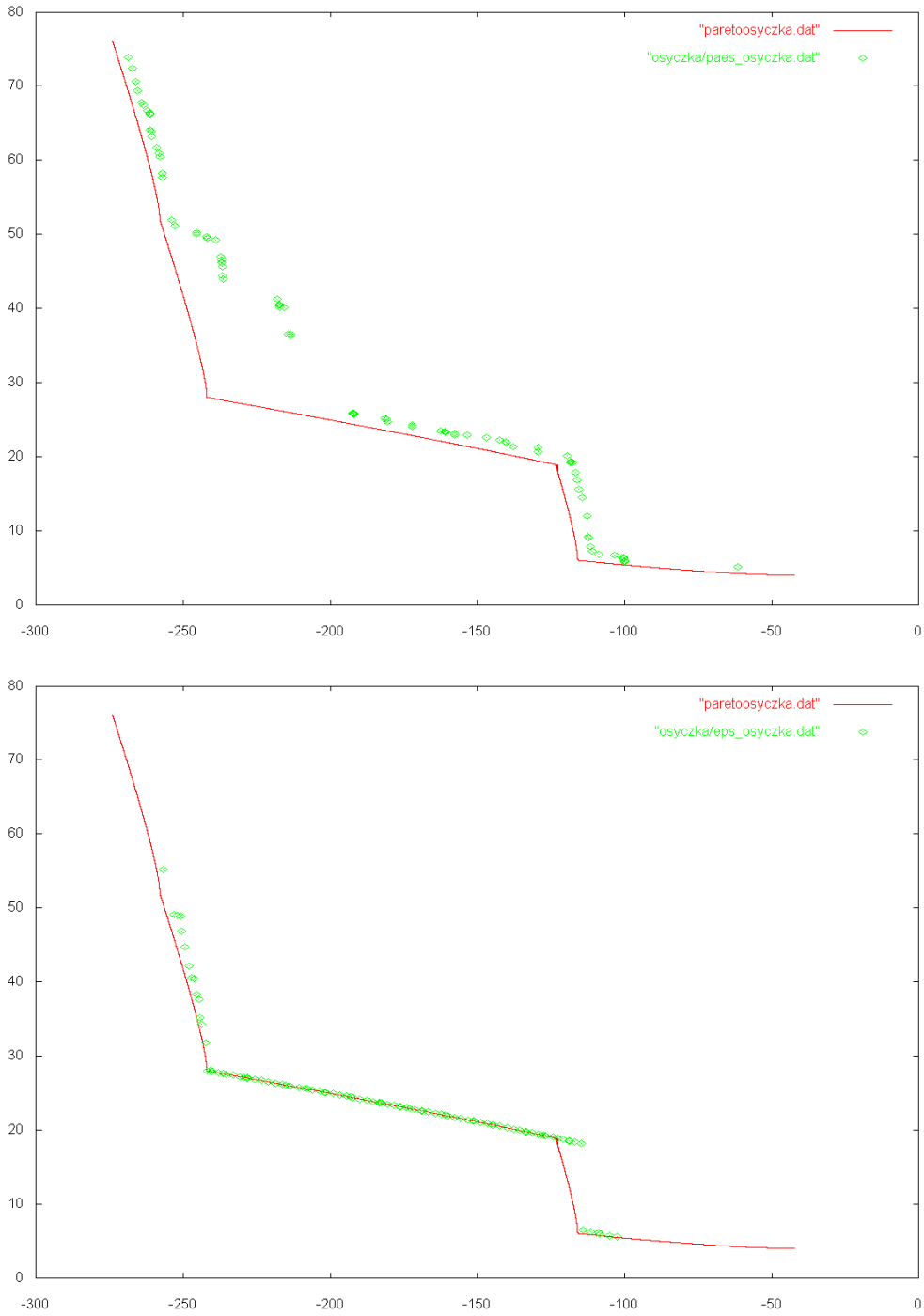

Figura 6.22: Frentes de Pareto generados PAES (arriba), ϵ-MyDE (abajo) para la segunda función de prueba (MOPC2).

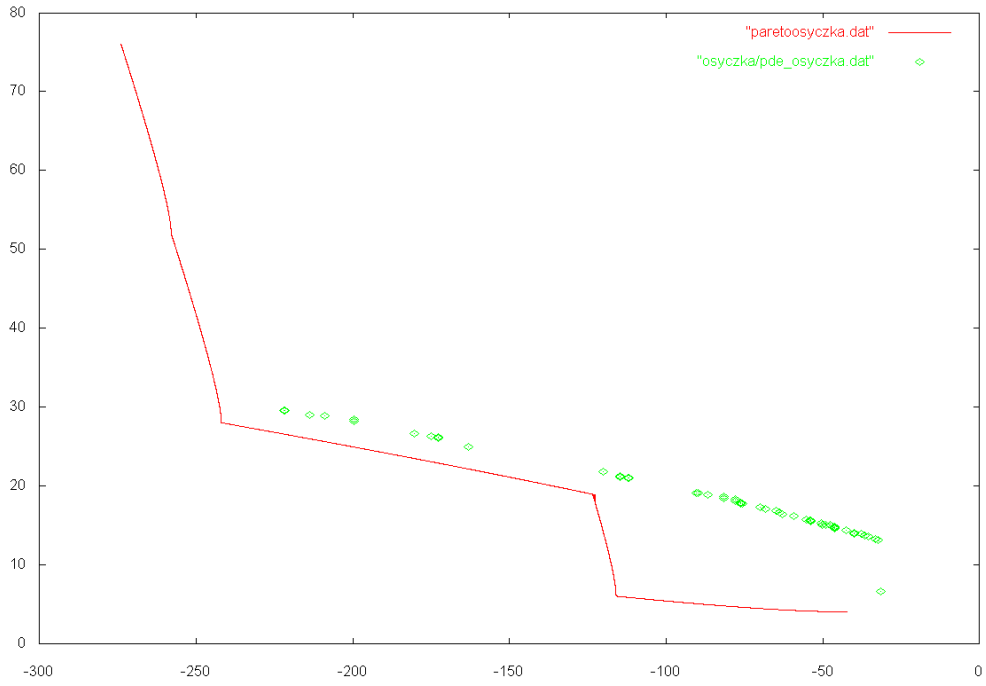

Figura 6.23: Frente de Pareto generado PDE para la segunda función de prueba (MOPC2).

Métrica - Algoritmo		media	mejor	peor	desv. Estándar
TE	MyDE	1	1	1	0
	NSGA-II	**0.935**	**0.78**	**1**	0.05558303
	PAES	1	1	1	**0**
	e-MyDE	1	1	1	**0**
	PDE	1	1	1	**0**
DG	MyDE	0.25091575	0.0649469	0.695286	0.18584031
	NSGA-II	0.48532375	**0.0204533**	1.56373	0.38820144
	PAES	0.89150985	0.176606	2.34472	0.66331084
	e-MyDE	**0.20735882**	0.0342463	**0.530559**	**0.13298887**
	PDE	2.3785781	0.762205	7.01374	1.75263415
D	MyDE	1.26613605	0.581057	1.91242	0.32236935
	NSGA-II	1.03451205	0.337736	1.44289	0.24461866
	PAES	3.04989795	0.839409	8.05777	2.18155976
	e-MyDE	1.03451205	0.337736	1.44289	0.24461866
	PDE	**1.5032201**	**0.125007**	**4.93991**	**1.31212847**

C	MyDE	NSGA-II	PAES	ϵ-MyDE	PDE	Dominan
MyDE		0.462376	0.829619	0.405999	0.962632	0.6651565
NSGA-II	0.958416		0.964834	0.830209	0.990519	**0.9359945**
PAES	0.221287	0.311881		0.166042	0.918015	0.40430625
ϵ-MyDE	0.866832	0.534653	0.831105		0.976576	0.8022915
PDE	0.144554	0.325743	0.707776	0.106052		0.32103125
Son Dominados	0.54777225	0.40866325	0.8333335	**0.3770755**	0.9619355	

Tabla 6.9: Resultados de Tasa de Error (TE), Distancia Generacional (DG), Distribución (D) y Cobertura (C) para el segundo problema, (MOPC2).

Métrica - Algoritmo		media	mejor	peor	desv. Estándar
TE	MyDE	0.9825	0.87	1	0.03507511
	NSGA-II	**0.7825**	**0.06**	**1**	0.27466487
	PAES	0.8985	0.59	1	0.11739609
	e-MyDE	1	1	1	**0**
	PDE	1	1	1	**0**
DG	MyDE	1.1571035	1.01423	1.60108	0.16035526
	NSGA-II	10.3952045	1.04607	32.8294	11.7194163
	PAES	1.6526235	1.03134	3.95321	0.79937593
	e-MyDE	**0.8778645**	**0.794507**	**0.971234**	**0.04478237**
	PDE	9.71233435	0.952927	34.7247	9.4525265
D	MyDE	14.198476	3.3256	91.228	20.4043087
	NSGA-II	**6.631766**	3.20602	14.1835	**3.79596756**
	PAES	24.6430905	5.76301	58.5302	10.7948238
	e-MyDE	54.6245215	2.46762	110.915	29.5975387
	PDE	6.8780055	**1.92143**	**13.9599**	3.8825971

C	MyDE	NSGA-II	PAES	ϵ-MyDE	PDE	Dominan
MyDE		0.658911	0.735546	0.180549	0.80396	0.5947415
NSGA-II	0.0980684		0.647059	0.0226777	0.79703	0.39120878
PAES	0.254086	0.767822		0.0274749	0.800495	0.46246948
ϵ-MyDE	0.907875	0.599505	0.581197		0.821287	**0.727466**
PDE	0.767707	0.578218	0.499246	0.56389		0.60226525
Son Dominados	0.5069341	0.651114	0.615762	**0.1986479**	0.805693	

Tabla 6.10: Resultados de Tasa de Error (TE), Distancia Generacional (DG), Distribución (D) y Cobertura (C) para el tercer problema, (MOPC3).

se acerca, sí llega a cubrirlo, y el segundo, continuamente se queda cerca del frente, aunque nunca lo cubre. En distribución, PDE es el que mejores resultados obtiene, aunque se ve en la gráfica que su desempeño es muy malo en lo que a convergencia se refiere pues no se acerca al frente verdadero.

En cuanto a la métrica de cobertura (C) en la que se comparan directamente los algoritmos entre sí, NSGA-II es al que menos dominan en promedio los demás algoritmos, aunque ϵ-MyDE es el que en promedio cubre más de los demás algoritmos. Esto es debido a que ϵ-MyDE obtiene resultados aceptables en todas sus ejecuciones (es decir, más consistente).

6.5.3. MOPC3

Las gráficas correspondientes al MOPC3 se muestran en las figuras 6.24, 6.25 y 6.26 y el resultado estadístico de la aplicación de las métricas se muestra en la tabla 6.10.

En este problema se observa que el verdadero frente tiene dos regiones,

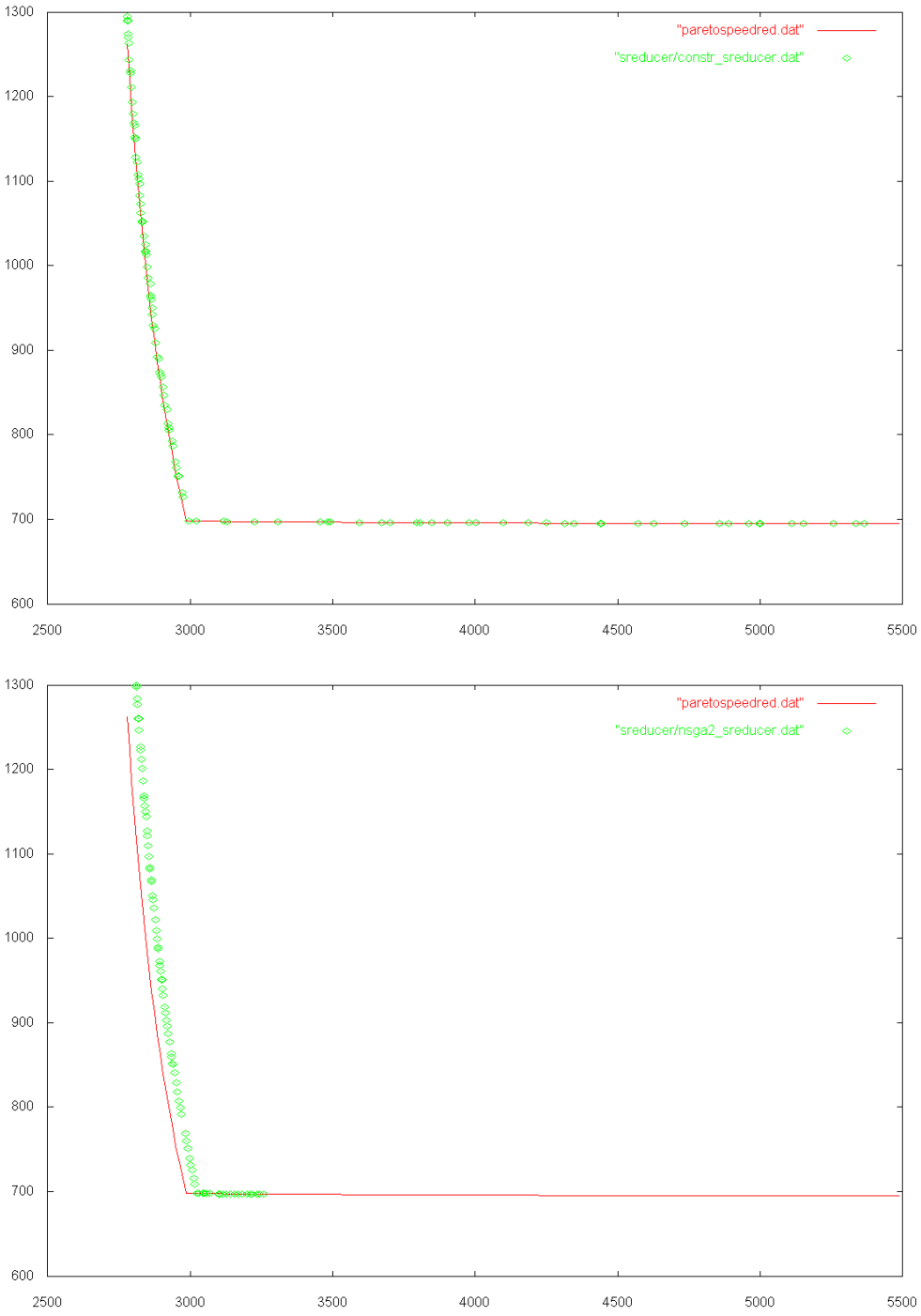

Figura 6.24: Frentes de Pareto generados por MyDE (arriba), NSGA-II (abajo) para la tercer función de prueba (MOPC3).

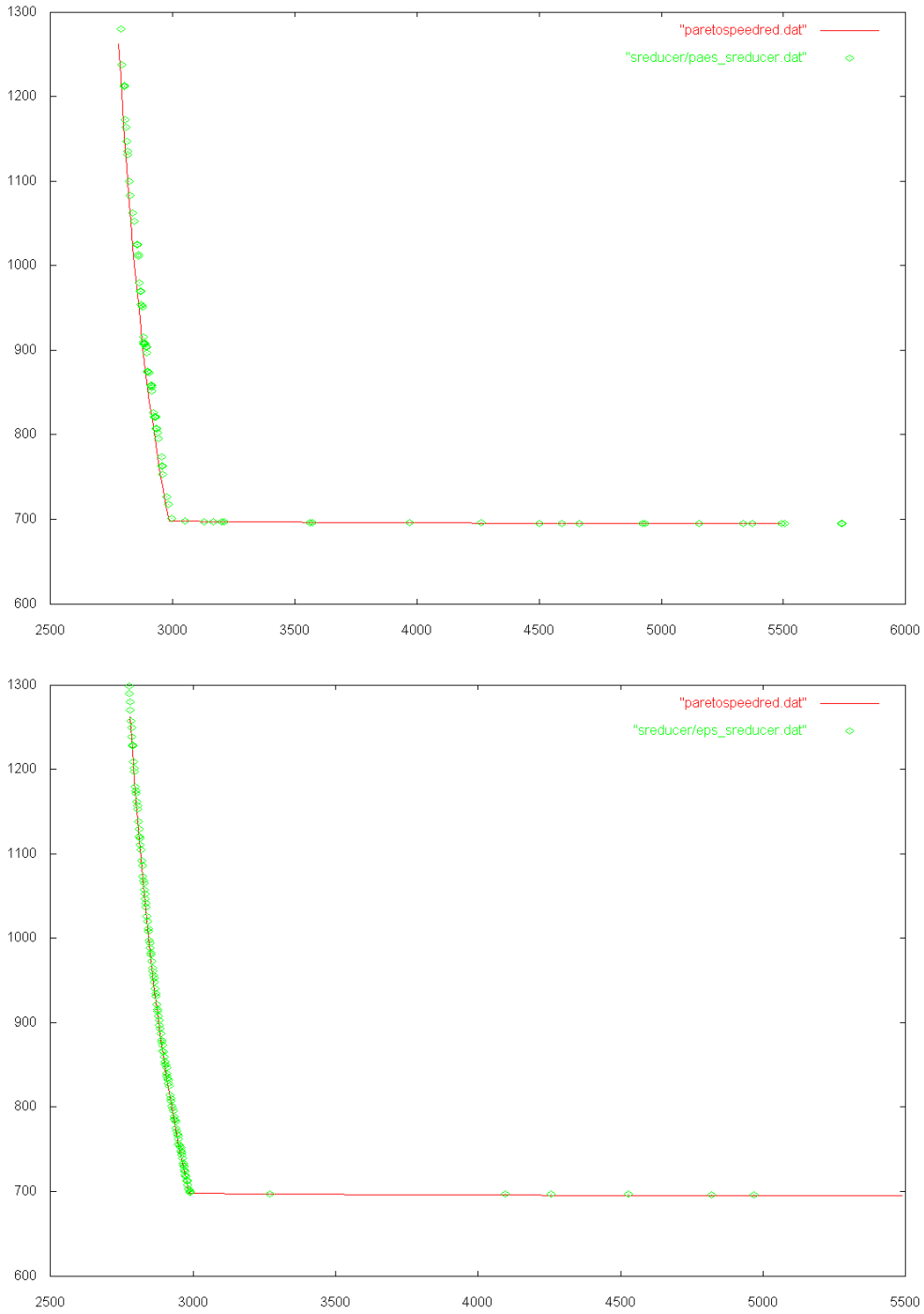

Figura 6.25: Frentes de Pareto generados PAES (arriba), ϵ-MyDE (abajo) para la tercer función de prueba (MOPC3).

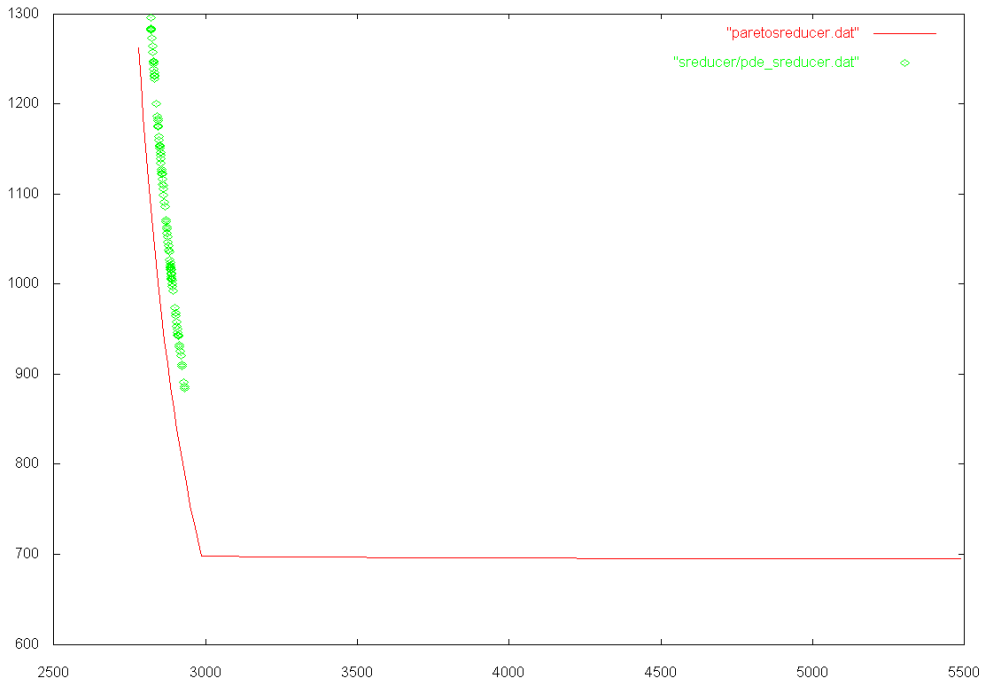

Figura 6.26: Frente de Pareto generado PDE para la tercer función de prueba (MOPC3).

una vertical y una difícil de alcanzar que es la parte baja (y muy horizontal), por lo que muchos algoritmos no llegan a cubrirla. MyDE es el que produce la mejor aproxamación, pues cubre ambas regiones del frente con una buena distribución. NSGA-II sólo alcanza la región vertical y no se acerca del todo mostrando buena distribución. PAES sí cubre ambas regiones pero su distribución es mala. ϵ-MyDE se acerca mucho al verdadero frente en la región vertical y en la horizontal casi no obtiene soluciones (esto se debe a la forma en que funciona la dominancia-ϵ). PDE es el que peor se ve, pues no se acerca al verdadero frente ni cubre las regiones del mismo.

De forma cuantitativa, se utilizan las métricas y se observa que NSGA-II es el mejor en TE y ϵ-MyDE en DG. Esto significa que el primero es el que mejor cubre el frente verdadero, y el segundo continuamente se queda cerca del frente (sin llegar a cubrirlo). En distribución, NSGA-II y PDE son los que mejores resultados obtienen.

En cuanto a la métrica de cobertura (C) en la que se comparan directamente los algoritmos entre sí, ϵ-MyDE es el que mejores resultados obtiene, pues es al que menos dominan en promedio y el que más domina en promedio a los demás algoritmos.

6.5.4. MOPC4

Las gráficas correspondientes al MOPC4 se muestran en las figuras 6.27, 6.28 y 6.29 y el resultado estadístico de la aplicación de las métricas se muestra en la tabla 6.11.

Las gráficas nos ayudan a evaluar el desempeño de los algoritmos de forma visual. Se ve por ejemplo, que todos los algoritmos se acercaron al frente de Pareto verdadero. MyDE logra cubrir todas las regiones del frente. NSGA-II se queda siempre en una sola región del frente, y aunque llega a cubrirlo sólo lo hace en esa zona. PAES también se acerca, aunque la distribución que se obtiene no es muy buena. ϵ-MyDE es el que mejor se ve, también se acerca y tiene una muy buena distribución. PDE también llega al frente verdadero pero no se ve una buena distribución.

De forma cuantitativa, se utilizan las métricas y se observa que NSGA-II es el que mejores resultados obtuvo en TE, DG y en distribución. Esto es debido a que siempre se quedó en una sola región la cual genera muy buenas soluciones en esa sola región pero no cubre todo el frente.

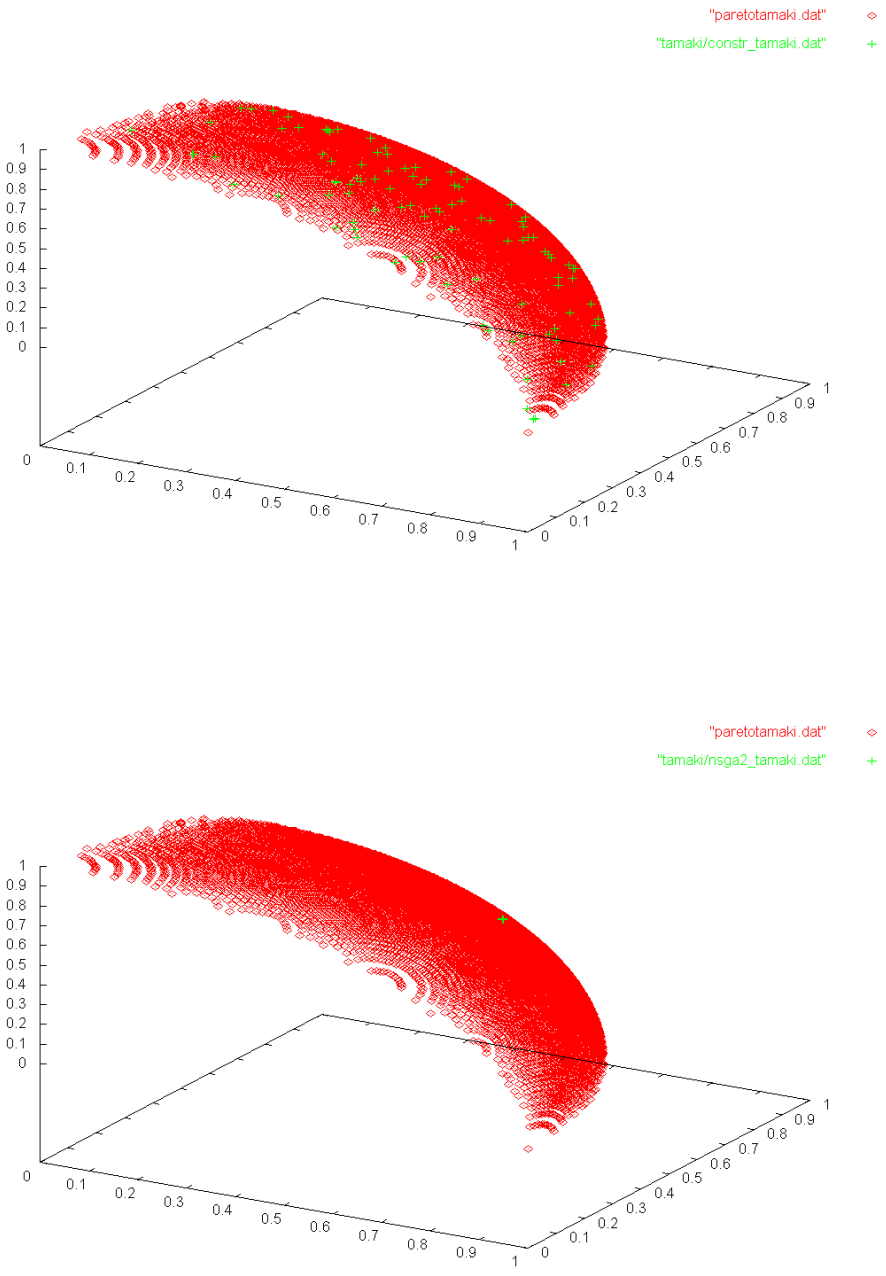

Figura 6.27: Frentes de Pareto generados por MyDE (arriba), NSGA-II (abajo) para la cuarta función de prueba (MOPC4).

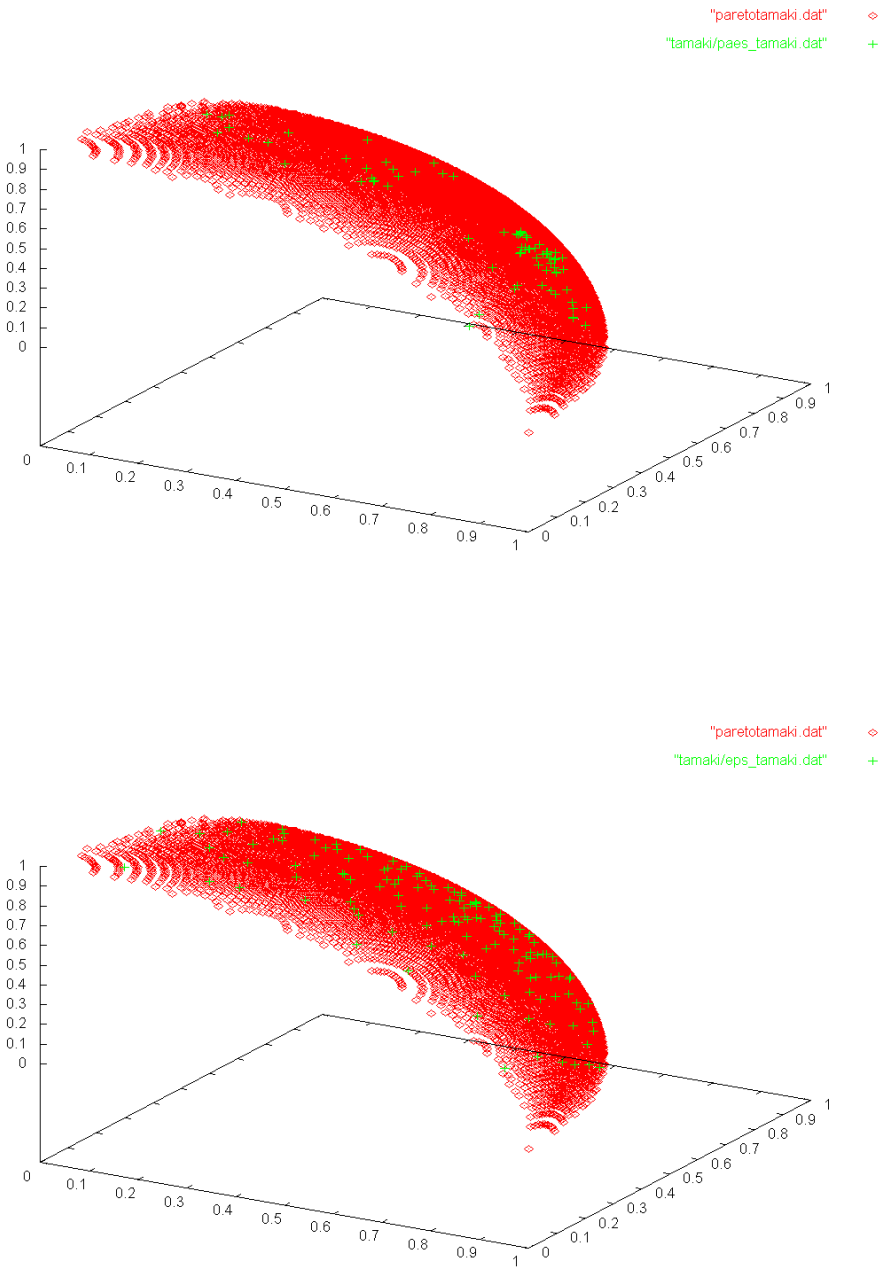

Figura 6.28: Frentes de Pareto generados PAES (arriba), ϵ-MyDE (abajo) para la cuarta función de prueba (MOPC4).

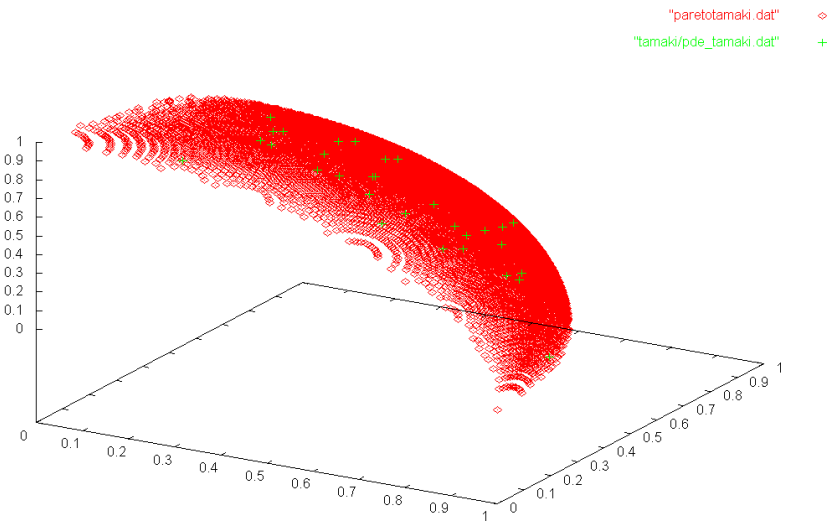

Figura 6.29: Frente de Pareto generado PDE para la cuarta función de prueba (MOPC4).

Métrica - Algoritmo		media	mejor	peor	desv. Estándar
TE	MyDE	0.704	0.62	0.79	0.04615762
	NSGA-II	**0**	**0**	**0**	**0**
	PAES	0.8115	0.6	0.93	0.09149489
	e-MyDE	0.53778185	0.478261	0.592593	0.03600738
	PDE	0.93140125	0.814815	1	0.0536189
DG	MyDE	0.00150789	0.00128463	0.00182941	0.00013476
	NSGA-II	**0.00035469**	**0.00023439**	**0.00050671**	7.5268E-05
	PAES	0.00248317	0.0014505	0.00352238	0.00050972
	e-MyDE	0.00087087	0.00077159	0.00103772	**7.1952E-05**
	PDE	0.00804467	0.00586278	0.011823	0.00163697
D	MyDE	0.05380946	0.0459221	0.0600146	0.00370726
	NSGA-II	**4.8229E-05**	**1.4082E-06**	**0.00021099**	**5.7565E-05**
	PAES	0.06564428	0.0457067	0.0856483	0.00876494
	e-MyDE	0.04267574	0.0381671	0.0493826	0.0031751
	PDE	0.09345934	0.0654136	0.14195	0.02128756

C	MyDE	NSGA-II	PAES	ϵ-MyDE	PDE	Dominan
MyDE		0	0.385149	0.0943177	0.644893	0.28108993
NSGA-II	0.00643564		0.0554455	0.00579822	0.118765	0.04661109
PAES	0.0990099	0		0.0626208	0.485748	0.16184468
ϵ-MyDE	0.293564	0	0.492079		0.735154	**0.38019925**
PDE	0.059901	0	0.20396	0.0204871		0.07108703
Son Dominados	0.11472764	**0**	0.28415838	0.04580596	0.49614	

Tabla 6.11: Resultados de Tasa de Error (TE), Distancia Generacional (DG), Distribución (D) y Cobertura (C) para el cuarto problema, (MOPC4).

Métrica - Algoritmo		media	mejor	peor	desv. Estándar
TE	MyDE	0.213	0.12	0.36	0.06610199
	NSGA-II	0.858	0.42	0.99	0.17398125
	PAES	0.86	0.76	0.96	0.06349472
	e-MyDE	**0.07928732**	**0.0377358**	**0.121212**	**0.02432303**
	PDE	0.645	0.53	0.99	0.14858986
DG	MyDE	0.11284635	0.00663767	0.314193	0.08894717
	NSGA-II	0.13713818	0.0120898	0.251471	0.07971614
	PAES	0.3471447	0.00637102	0.616793	0.15438835
	e-MyDE	**0.03116279**	**0.00516507**	**0.0875361**	**0.02858158**
	PDE	0.2253863	0.182241	0.319493	0.03682927
D	MyDE	0.1546079	0.0633468	0.538421	0.13287217
	NSGA-II	**0.06280001**	**0.0437959**	**0.108578**	**0.01558645**
	PAES	0.66808426	0.0503822	6.30262	1.34660086
	e-MyDE	0.3375712	0.116799	0.730226	0.19230002
	PDE	0.15048147	0.0557389	0.527688	0.13080187

C	MyDE	NSGA-II	PAES	ϵ-MyDE	PDE	Dominan
MyDE		0.935149	0.920792	0.132112	0.912376	**0.72510725**
NSGA-II	0.382557		0.930198	0.107522	0.817327	0.559401
PAES	0.490089	0.868812		0.675988	0.655941	0.6727075
ϵ-MyDE	0.615956	0.820297	0.707921		0.659406	0.700895
PDE	0.393459	0.805446	0.784653	0.108004		0.5228905
Son Dominados	0.47051525	0.857426	0.835891	**0.2559065**	0.7612625	

Tabla 6.12: Resultados de Tasa de Error (TE), Distancia Generacional (DG), Distribución (D) y Cobertura (C) para el quinto problema, (MOPC5).

En cuanto a la métrica de cobertura (C) en la que se comparan directamente los algoritmos entre sí, NSGA-II es al que menos dominan en promedio los demás algoritmos (pues siempre obtuvo soluciones no dominadas con respecto a todos los demás), aunque ϵ-MyDE es el que en promedio cubre más soluciones de los demás algoritmos. Esto es porque las soluciones de NSGA-II siempre fueron no dominadas con respecto a todos los algoritmos. ϵ-MyDE cubre todas las regiones del frente, por lo que tiene mayor capacidad de superar las soluciones de los otros algoritmos.

6.5.5. MOPC5

Las gráficas correspondientes al MOPC5 se muestran en las figuras 6.30, 6.31 y 6.32 y el resultado estadístico de la aplicación de las métricas se muestra en la tabla 6.12.

Es importante mencionar que el verdadero frente fue generado por enumeración. Sin embargo, debido a la alta dimensionalidad del problema, las divisiones por variable no fueron muchas, ya que tomaría mucho

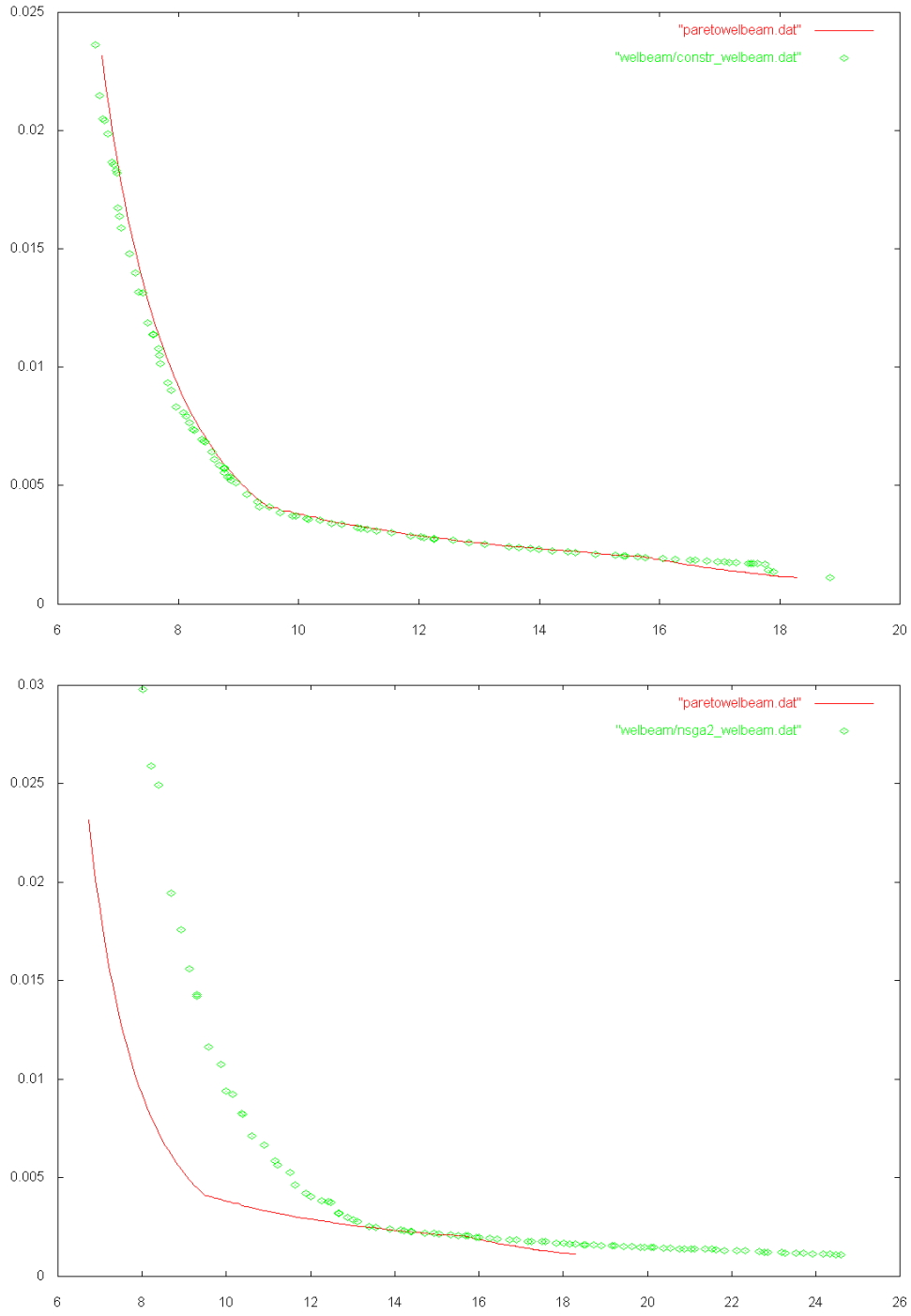

Figura 6.30: Frentes de Pareto generados por MyDE (arriba), NSGA-II (abajo) para la quinta función de prueba (MOPC5).

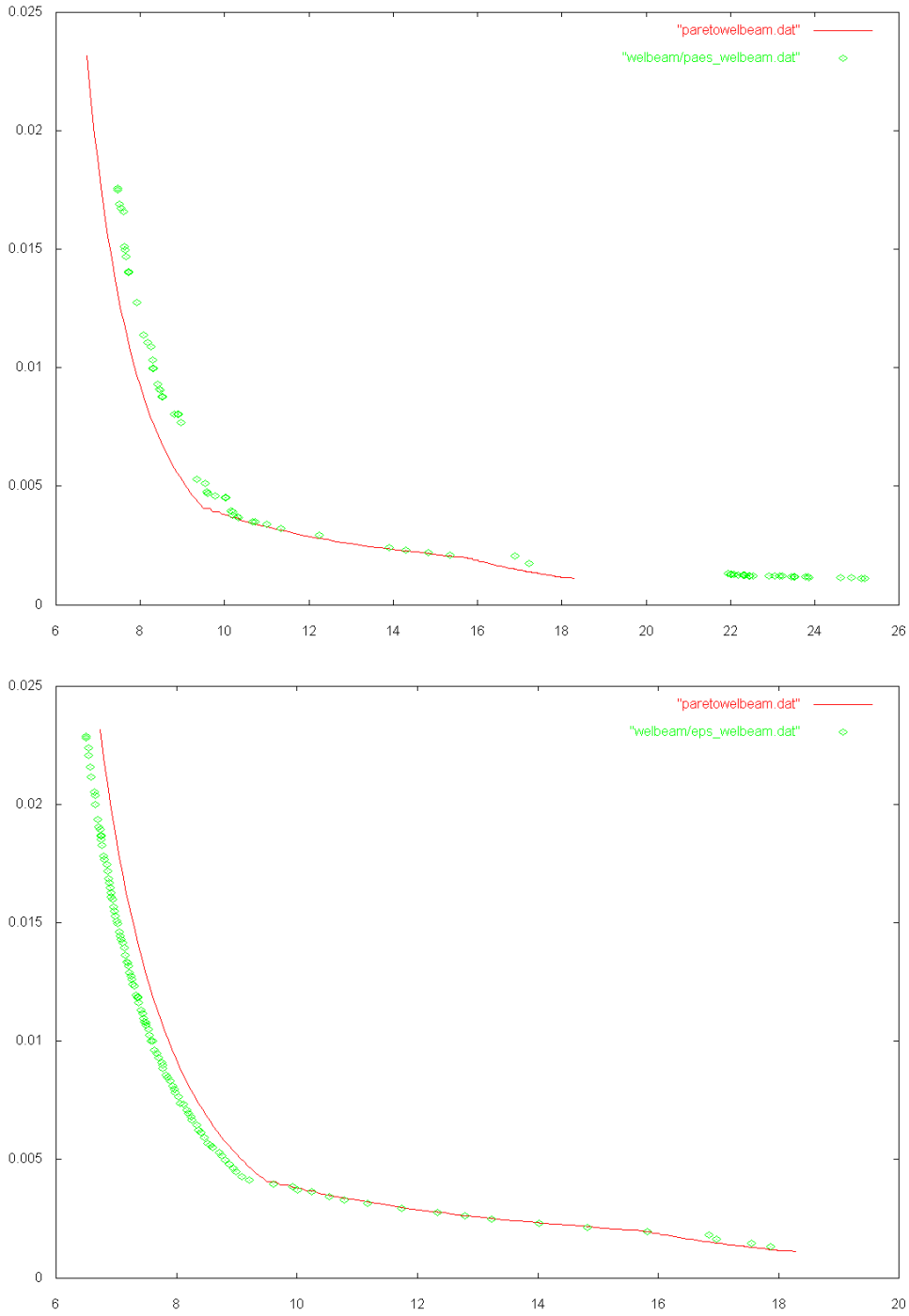

Figura 6.31: Frentes de Pareto generados PAES (arriba), ϵ-MyDE (abajo) para la quinta función de prueba (MOPC5).

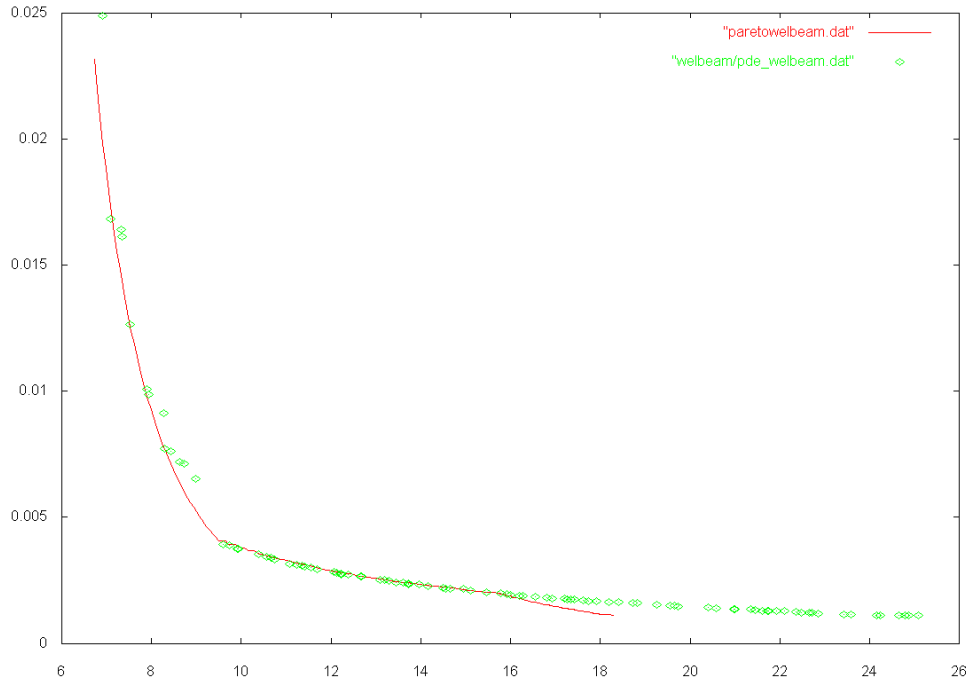

Figura 6.32: Frente de Pareto generado PDE para la quinta función de prueba (MOPC5).

tiempo encontrar el frente de Pareto verdadero. Por ende, el frente de Pareto que se proporciona no tiene una buena precisión. Además, de que en la segunda función objetivo de este mismo problema, los valores del frente son muy pequeños (van de 0.001 a 0.02), por lo que la gráfica hace notar que se encontró un mejor frente en algunos algoritmos.

Las gráficas nos ayudan a evaluar el desempeño de los algoritmos con respecto a todas las regiones del frente. MyDE cubre todas las regiones del frente y presenta una buena distribución dejando algunos huecos en el frente. NSGA-II se queda atrapado en algunas ocasiones en un frente falso, aunque su distribución es buena. PAES también se acerca sin llegar a cubrirlo, además de que la distribución no es muy buena. ϵ-MyDE es el que mejor se ve, cubriendo el frente en todas sus regiones y mostrando una buena distribución, aunque en la gráfica se nota que en una región no existen tantas soluciones pero este problema es debido a la dominancia-ϵ. PDE también llega al frente verdadero, pero su distribución no es buena, además de que hay regiones a las que llega con pocas o ninguna solución .

De forma cuantitativa, se utilizan las métricas y se observa que ϵ-MyDE es el que mejores resultado obtuvo en TE y en DG. Y en Distribución, el NSGA-II obtiene los mejores resultados.

En cuanto a la métrica de cobertura (C) en la que se comparan directamente los algoritmos entre sí, tanto MyDE como ϵ-MyDE son los que mejores resultados presentan, cubriendo a los demás algoritmos en su mayoría.

6.6. Conclusiones sobre los resultados en problemas con restricciones

En base a los resultados que MyDE obtiene en comparación con los demás algoritmos en estas funciones con restricciones, el desempeño de este algoritmo es muy competitivo. Aunque en el problema MOPC2 no cubre el frente verdadero, sí se acerca mucho en todas las regiones, y en las funciones restantes se acerca a los frentes verdaderos hasta cubrirlos en algunos casos. ϵ-MyDE, a diferencia del MyDE, se comportó mejor en la mayoría de los casos, pues en todos los problemas siempre fue el que mejores resultados obtuvo en la métrica de cobertura (C) y mantuvo un buen desempeño en las métricas restantes, además de que en las gráficas de los frentes se

observa que siempre alcanza a cubrir las regiones del frente de Pareto verdadero. Nótese sin embargo que el mismo concepto de dominancia-ϵ hace que en los frentes que tienen una pendiente muy vertical o muy horizontal no existan muchas soluciones factibles y, como consecuencia se forma un frente, con algunos huecos.

Capítulo 7

Conclusiones

Las técnicas de computación evolutiva han mostrado la capacidad de resolver una gran variedad de problemas difíciles. De entre los diferentes algoritmos evolutivos existentes, la evolución diferencial ha resultado una técnica muy efectiva para optimización en espacios continuos.

En esta tesis se utilizó la evolución diferencial para resolver problemas multiobjetivo con y sin restricciones. La idea de utilizar la evolución diferencial fue debido a su alta velocidad de convergencia demostrada en optimización global. Sin embargo, dicha velocidad de convergencia puede causar una presión de selección muy alta, lo cual puede producir convergencia prematura. Para controlar la presión de selección se propuso un esquema que mantiene diversidad variando la forma de selección a cada cierto número de iteraciones.

En el problemas multiobjetivo sin restricciones el algoritmo propuesto mostró resultados que mejoran en su mayoría a los obtenidos con técnicas evolutivas representantivas del estado del arte como el PAES, PDE y NSGA-II.

Para problemas multiobjetivo con restricciones no existían propuestas previas, por lo que se implementó un esquema relativamente simple de manejo de restricciones. Este mismo esquema se manejo en todos los algoritmos, con el fin de evaluarlos en igualdad de condiciones. En base a los resultados, se observa que ϵ-MyDE es el que mejores resultados obtiene en todas las funciones con restricciones.

Así entonces, se puede observar que ambas versiones demuestran ser muy competitivas, haciendose notar que la velocidad de convergencia marca una importante diferencia en lo referente a la obtención de buenas so-

luciones.

7.1. Trabajo futuro

En cuanto al trabajo futuro, se puede mejorar el mecanismo para manejo de restricciones, de forma que se encuentren más soluciones dentro de la zona factible más rápido y explotar dichas soluciones dentro de la frontera de la factibilidad para acercarse mejor al verdadero frente de Pareto.

Un problema que se presenta con la evolución diferencial es la convergencia prematura, la cual se logró controlar en el esquema propuesto mediante la selección de los padres para formar al hijo. Sin embargo, se pueden buscar formas más eficientes de explotar las soluciones sin perder de vista su diversidad.

En cuanto a la distribución de soluciones a lo largo del frente utilizando la dominancia-ϵ y su relación con el número de individuos no dominados, se podría encontrar una mejor forma de auto adaptar el valor de ϵ cuando se fija el número de soluciones a un parámetro específico por parte del usuario. Esto permitiría calcular de forma precisa el valor del vector-ϵ, sin perder la buena distribución ajustándose al número de soluciones requeridas. Además, esto permitiría la pérdida de los extremos del frente de Pareto.

También se puede lograr la auto-adaptación de varios parámetros, como el número de generaciones que se ejecutan, pues en ocasiones el algoritmo converge muy rápido al frente de Pareto y sería importante que el algoritmo detectara que no ha habido más mejoras y se detuviera automáticamente. Esto reduciría el número de evaluaciones de las funciones objetivo y facilitaría más el uso del algoritmo.

Bibliografía

[1] Hussein A. Abbass. The Self-Adaptive Pareto Differential Evolution Algorithm. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 831–836, Piscataway, New Jersey, May 2002. IEEE Service Center.

[2] Hussein A. Abbass and Ruhul Sarker. The Pareto Differential Evolution Algorithm. *International Journal on Artificial Intelligence Tools*, 11(4):531–552, 2002.

[3] B.V. Babu and M. Mathew Leenus Jehan. Differential Evolution for Multi-Objective Optimization. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 4, pages 2696–2703, Canberra, Australia, December 2003. IEEE Press.

[4] Thomas A. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[5] Colin B.Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, Great Britain, 1993.

[6] Carlos A. Coello Coello and Arturo Hernández Aguirre. Design of Combinational Logic Circuits through an Evolutionary Multiobjective Optimization Approach. *Artificial Intelligence for Engineering, Design, Analysis and Manufacture*, 16(1):39–53, January 2002.

[7] J.L. Cohon and D.H. Marks. A review and evaluation of multiobjective programming techniques. *Water Resources Research*, 11(2):208–220, 1975.

[8] David W. Corne, Joshua D. Knowles, and Martin J. Oates. The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 839–848, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[9] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In Schaffer J.D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, Washington, D.C., 1989. Morgan Kauffman Publishers.

[10] Kalyanmoy Deb. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7(3):205–230, Fall 1999.

[11] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[12] Kalyanmoy Deb, Manikanth Mohan, and Shikhar Mishra. Towards a Quick Computation of Well-Spread Pareto-Optimal Solutions. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 222–236, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.

[13] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Multi-Objective Optimization Test Problems. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 825–830, Piscataway, New Jersey, May 2002. IEEE Service Center.

[14] David B. Fogel, editor. *Evolutionary Computation. The Fossil Record. Selected Readings on the History of Evolutionary Algorithms*, New York, 1998. The Institute of Electrical and Electronic Engineers.

[15] Lawrence J. Fogel. Artificial intelligence through simulated evolution. *John Wiley*, 1966.

[16] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[17] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, Massachusetts, 1998.

[18] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[19] D.M. Himmelblau. Decomposition of large-scale problems. New York, North-Holland, 1973.

[20] A. Hoffman. *Arguments on Evolution: A Paleontologist's Perspective*. Oxford University Press, New York, 1989.

[21] John H. Holland. Outline for a logical theory of adaptative systems. *Journal of the Association for Computing Machinery*, (9):297–314, 1962.

[22] John H. Holland. *Evolutionstrategie: Optimierung technisher nach Prinzipien der biologischen Evolution*. Fromman-Holzboog, Stuttgart, Alemania, 1973.

[23] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[24] Golinski J. Optimal synthesis problems solved by means oj nonmlinear programming and random methods. *Journal of Mechanisms*, 5:287 – 309, 1970.

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

[26] Hajime Kita, Yasuyuki Yabumoto, Naoki Mori, and Yoshikazu Nishikawa. Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature—PPSN IV*, Lecture Notes in Computer Science, pages 504–512, Berlin, Germany, September 1996. Springer-Verlag.

[27] Frank Kursawe. A Variant of Evolution Strategies for Vector Optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science Vol. 496*, pages 193–197, Berlin, Germany, October 1991. Springer-Verlag.

[28] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining Convergence and Diversity in Evolutionary Multiobjective Optimization. *Evolutionary Computation*, 10(3):263–282, Fall 2002.

[29] Joanna Lis and A. E. Eiben. A Multi-Sexual Genetic Algorithm for Multiobjective Optimization. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Computation*, pages 59–64, Nagoya, Japan, 1996. IEEE.

[30] A. Callahan M. Maxfield and L.J. Fogel, editors. *Artificial Intelligence through a Simulation of Evolution*, Washington D.C., 1965. Biophysics and Cybernetics Systems: Proceedings of the Second Cybernetics Sciences, Spartan Books.

[31] Efrén Mezura-Montes and Carlos A. Coello Coello. A Simple Evolution Strategy to Solve Constrained Optimization Problems. In Erick Cantú-Paz, James A. Foster, Kalyanmoy Deb, Lawrence David Davis, Rajkumar Roy, Una-May O' Reilly, Hans-Georg Beyer, Russell Standish, Graham Kendall, Stewart Wilson, Mark Harman, Joachim Wegener, Dipankar Dasgupta, Mitch A. Potter, Alan C. Schultz, Kathryn A. Dowsland, Natasha Jonoska, and Julian Miller, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2003)*, pages 640–641, Heidelberg, Germany, July 2003.

Chicago, Illinois, Springer Verlag. Lecture Notes in Computer Science Vol. 2723.

[32] Andrzej Osyczka and Sourav Kundu. A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization*, 10:94–99, 1995.

[33] V. Pareto. *Cours D' Economie Politique*, volume I and II. F. Rouge, Lausanne, 1896.

[34] K. Price and R. Storn. Sitio web de la evolu ción diferencial. http://www.icsi.berkeley.edu/storn/code.html, 2004.

[35] K. M. Ragsdell and D. T. Phillips. Optimal design of a class of welded structures using geometric programming. *Journal of Engineering for Industry Series B*, B(98):1021–1025, 1975.

[36] J. David Schaffer and John J. Grefenstette. Multiobjective Learning via Genetic Algorithms. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 593–595, Los Angeles, California, 1985. AAAI.

[37] Marc Schoenauer and Zbigniew Michalewicz. Evolutionary Computation at the Edge of Feasibility. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the Fourth Conference on Parallel Problem Solving from Nature (PPSN IV)*, pages 245–254, Heidelberg, Germany, September 1996. Berlin, Germany, Springer-Verlag.

[38] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.

[39] Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient adaptative scheme for global optimization over continuous spaces. Technical Report TR-95- 12, International Computer Science, Berkeley, California, March 1995. .

[40] Rainer Storn and Kenneth Price. Differential evolution - a fast and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, (11):341–359, 1997.

[41] Hisashi Tamaki, Hajime Kita, and Shigenobu Kobayashi. Multi-Objective Optimization by Genetic Algorithms : A Review. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Computation (ICEC'96)*, pages 517–522, Nagoya, Japan, 1996. IEEE.

[42] Manuel Valenzuela-Rendón and Eduardo Uresti-Charre. A Non-Generational Genetic Algorithm for Multiobjective Optimization. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 658–665, San Mateo, California, July 1997. Michigan State University, Morgan Kaufmann Publishers.

[43] Feng Xue, Arthur C. Sanderson, and Robert J. Graves. Pareto-based Multi-Objective Differential Evolution. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 2, pages 862–869, Canberra, Australia, December 2003. IEEE Press.

[44] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2002.

[45] Eckart Zitzler and Lothar Thiele. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 1998.