Centro de Investigación y de Estudios
Avanzados del Instituto Politécnico Nacional
Departamento de Computación

Métodos para reducir evaluaciones en algoritmos evolutivos multi-objetivo, basados en aproximación de funciones

Tesis que presenta

Víctor Antonio Serrano Hernández

para obtener el Grado de

Maestro en Ciencias

en la Especialidad de

Ingeniería Eléctrica

Director de la Tesis:

Dr. Carlos A. Coello Coello

México, D. F. Noviembre 2007

Dedicado a mis padres, quienes encontraron la forma óptima de impulsarme a conseguir mis anhelos.

Agradecimientos

Agradezco a mis padres y a mi hermanita, por apoyarme siempre a seguir mis sueños.

A mi tía Agus y a mi abuelita, por su inagotable paciencia durante el tiempo que he vivido con ellas. Gracias por cuidarme, consentirme y quererme tanto.

A toda mi familia, mis tíos, mis primos, en especial a mi prima Carmen quien es sobre todo, una amiga. Nada es difícil con el apoyo de la familia.

A Daniella, por ser parte de mis sueños y metas. Gracias por tu apoyo incondicional, tu alegría llena mi vida y me motiva a ser mejor. No existe mejor aliciente que el hacerme feliz a tu lado.

A don Benjamín, doña Silvia y a David, sin ustedes no hubiese podido terminar este proyecto, gracias por hacerme sentir como parte de su familia.

A Jessica, por escucharme y aconsejarme. Su opinión fue decisiva para embarcarme en esta aventura.

A mis demás amigos David, Octavio, Mili, Rubén, Fabián, Mario, Gustavo, Mony y Julio. Ustedes viven mis logros y derrotas en cada momento, gracias por brindarme su amistad.

A mis compañeros de generación, la convivencia con ustedes fue una experiencia única. Gracias por su apoyo y amistad, en especial a mi amigo Eduardo, Elizabeth, Cuauhtémoc y Marco quienes siempre encontraban un espacio en sus actividades para escucharme.

Al doctor Carlos Coello, su ejemplo es mi objetivo a seguir. Gracias por su guía en este trabajo.

A mis sinodales, Dr. Debrup Chakraborty y Dr. Luis Gerardo de la Fraga, por la revisión de mi tesis y lograr con ello un mejor trabajo.

A Sofi por apoyarnos en cada momento y ayudarnos de cualquier forma posible.

Al CINVESTAV por permitirme formar parte de esta gran institución.

Al CONACYT por el apoyo económico brindado, sin el cual no hubiese podido llevar a cabo este proyecto.

Resumen

Al buscar solucionar un problema de optimización, existen múltiples herramientas matemáticas que nos ayudan a lograr este objetivo. No obstante, en el mundo real existen problemas de optimización para los cuales es difícil, o incluso imposible, aplicar métodos de programación matemática con éxito. En estos problemas, las heurísticas se vuelven una alternativa viable.

Los algoritmos evolutivos han sido ampliamente utilizados en optimización multi-objetivo debido a su naturaleza basada en la evolución de una población (o conjunto de soluciones), que permite la generación de diversos elementos del conjunto de óptimos de Pareto en una sola ejecución. Sin embargo, en un gran número de problemas de la vida real, las evaluaciones de las funciones objetivo a optimizar son muy costosas, en el sentido económico y/o computacional, lo cual puede volver impráctico el uso de algún algoritmo evolutivo.

En esta tesis se propone una forma de reducir el número de evaluaciones de la función objetivo en el NSGA-II, que es un algoritmo evolutivo multi-objetivo del estado del arte, mediante la implementación de métodos de aproximación de funciones, para aprovechar así los beneficios ofrecidos por los algoritmos evolutivos reduciendo su principal desventaja.

Mediante el uso de métodos de aproximación de funciones es posible construir un modelo similar al real, con la ventaja de poder acceder a él las veces que sean necesarias sin los inconvenientes relacionados con la función real. Al implementar dichos métodos al NSGA-II, fue posible obtener valores óptimos (o cercanos a éstos) en problemas de prueba estándar sin necesidad de realizar un número excesivo de evaluaciones a la función objetivo real.

Abstract

In the task of searching solutions for an optimization problem, there are many mathematical tools which help us to achieve this objective. Nevertheless, in the real world there exist optimization problems for which it is difficult or even impossible to apply mathematical programming methods successfully. In this type of problems, heuristics become a viable alternative.

Evolutionary algorithms have been widely used in multi-objective optimization due to their population (or solutions set)-based nature which allows them to generate several Pareto optimal elements in a single run. Nevertheless, in a large amount of problems in real life, the evaluation of the functions to be optimized are costly in an economic and/or a computational sense, and may turn impractical the use of an evolutionary algorithm.

In this thesis, we propose a method to reduce the number of the objective functions evaluations in the NSGA-II, which is a multi-objective evolutionary algorithm representative of the state-of-the-art hybridized with an implementation of functions aproximation methods. The aim is to exploit the good characteristics of evolutionary algorithms while overcoming their main disadvantage.

The use of functions approximation methods makes possible to build a model similar to the real one, with the advantage of being able to access it the times required by the problem without the disadvantages of evaluating the real function. When this type of methods were implemented into the NSGA-II, it was possible to obtain optimal values (or near optimal) in standard test problems without the need of performing a large number of evaluations to the real (and presumably costly) function.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

INDICE DE TABLAS

LISTA DE ALGORITMOS

CAPÍTULO 1

Introducción

El área de optimización comprende múltiples disciplinas en su aplicación, incluyendo los sistemas industriales, la economía y la solución de problemas sociales entre otros. Sin embargo, algunos de estos problemas presentan dificultad para realizar evaluaciones del problema, las cuales son necesarias en cualquier método de optimización. Los algoritmos evolutivos cuentan con la ventaja de prescindir de una descripción matemática del problema y pueden ser utilizados como cajas negras donde, como entrada se ofrecen las evaluaciones del problema requeridas por el algoritmo y como salida se dan soluciones "aceptables". Otra característica de los algoritmos evolutivos es la necesidad de requerir un número considerable de evaluaciones de la función objetivo. En el mundo real, existen problemas cuyas evaluaciones requieren de un tiempo considerable de cómputo, el cual extrapolado significa múltiples evaluaciones y resulta prohibitivo para encontrar solución al problema, por lo cual, en estos casos los algoritmos evolutivos no son aplicables.

Existen múltiples métodos de aproximación de funciones, los cuales requieren de algunos vectores o puntos de entrenamiento para crear lo que se llamará meta-modelo, el cual, una vez entrenado, es capaz de producir valores cercanos a la respuesta ofrecida por la función que aproximan, tomando como entrada un punto dado.

Para explotar las ventajas que ofrecen los algoritmos evolutivos, es necesario reducir al mínimo las evaluaciones directas de la función objetivo, las cuales son costosas computacionalmente hablando. Para lograr esta condición, en esta tesis se propone utilizar métodos de aproximación de funciones como herramienta para dirigir la búsqueda en un algoritmo evolutivo y de esta forma realizar el menor número posible de evaluaciones costosas.

El método kriging está catalogado como un buen predictor [15, pág. 106]. Las Redes de Funciones Base Radiales (RFBR) tienen un tiempo de entrenamiento reducido y producen aproximaciones de buena calidad [38, pág. 285]. Por lo anterior, en este documento realizamos experimentos sobre variantes de los dos métodos antes mencionados.

A continuación, a manera de introducción al tema de algoritmos evolutivos, en la siguiente sección se hace una breve descripción del tema.

1.1. Computación evolutiva

Existen problemas de optimización que se encuentran definidos de forma exacta y precisa por una expresión matemática. Para estos casos existen múltiples métodos de solución disponibles. Sin embargo, cuando nos enfrentamos a problemas altamente no lineales, con ruido y con espacios de búsqueda accidentados, se requiere de otro tipo de algoritmos que puedan sortear estas características, las cuales evidentemente incrementan la dificultad del problema. También se busca evitar el quedar atrapados en soluciones que representen óptimos locales (algo que suele ocurrir con los algoritmos de programación matemática). En este último tipo de problemas se ha utilizado con éxito la computación evolutiva.

"La computación evolutiva es el estudio de sistemas computacionales que utilizan ideas y obtienen su inspiración de la evolución y adaptación natural. El objetivo principal de la computación evolutiva es desarrollar sistemas cada vez más eficientes y robustos para resolver problemas complejos del mundo real [82]".

Para comprender el funcionamiento general de la computación evolutiva, es necesario primero entender las teorías evolutivas que la motivaron, por lo que a continuación se proporcionará una revisión rápida de los preceptos Neodarwinistas, que engloban a la mayoría de las ideas tomadas de la evolución natural que se utilizan en la computación evolutiva.

1.1.1. Neodarwinismo

Existen diversas teorías de la evolución natural, pero de entre ellas, la más aceptada en la actualidad es la síntesis evolutiva moderna o neodarwinismo. El neodarwinismo mezcla la teoría de la evolución de las especies mediante la selección natural de Charles Darwin, con la teoría genética de Gregor Mendel como base de la herencia biológica y el seleccionismo de August Weismann, considerando además a la mutación genética aleatoria como fuente de variación.

Los principios de la teoría neodarwinista establecida en los años 30 y 40's enuncian que la variación genética de las poblaciones surge por azar mediante la mutación (causada por errores en la réplica del ácido desoxirribonucleico ó ADN) y la recombinación (la mezcla de los cromosomas homólogos durante la meiosis). La evolución consiste básicamente en los cambios de la frecuencia de los alelos entre las generaciones, como resultado de la deriva genética, el flujo genético y la selección natural. La especiación ocurre gradualmente cuando las poblaciones se encuentran aisladas reproductivamente.

La *deriva genética* es una fuerza evolutiva que actúa junto con la selección natural cambiando las características de las especies en el tiempo. Se trata de un cambio aleatorio en la frecuencia de alelos de una generación a otra. Normalmente se da una

pérdida de los alelos menos frecuentes, resultando en una disminución en la diversidad genotípica de la población.

El *flujo genético* (también conocido como migración) es la transferencia de genes de una población a otra. La migración hacia o desde una población puede ser responsable de importantes cambios en las frecuencias del acervo genético (el número de individuos con un rasgo particular). Hay un número de factores que afectan el ritmo del flujo genético entre poblaciones diferentes. Uno de los factores más significativos es la movilidad. Una mayor movilidad tiende a incrementar el potencial migratorio de un individuo.

El concepto clásico de *selección natural* afirma que las condiciones de un medio ambiente favorecen o dificultan la supervivencia o reproducción de los organismos vivos o individuos. La supervivencia de un individuo dependerá de sus peculiaridades, por lo que existe una selección implícita de aquellos individuos que tengan características favorables en el ambiente en el que se encuentren. Este concepto fue concebido por Darwin en un intento por explicar la evolución biológica. Se deriva de dos premisas: en la primera se afirma que entre los descendientes de un organismo hay una variación aleatoria, no determinista, que es, en parte, heredable. La segunda enuncia que esta variabilidad puede dar lugar a diferencias de supervivencia y de éxito reproductor, haciendo que algunas características de nueva aparición se puedan extender en la población. La acumulación de estos cambios a lo largo de las generaciones produciría todos los fenómenos evolutivos.

El neodarwinismo afirma que la historia de la mayoría de los seres vivos puede ser descrita con base en sólo un puñado de procesos estadísticos actuando sobre y dentro de las poblaciones de las especies [25]. Dichos procesos son la reproducción, la mutación, la competencia y la selección.

La *reproducción* es un proceso biológico que permite la producción de nuevos organismos, siendo una característica común de todas las formas de vida conocidas. Las dos modalidades básicas se agrupan en dos tipos, que reciben los nombres de asexual o vegetativa y de sexual o generativa. En la reproducción asexual un único organismo es capaz de originar otros individuos nuevos, que son copias del mismo desde el punto de vista genético. La reproducción sexual requiere la intervención de dos individuos, siendo, generalmente, de sexos diferentes. Los descendientes producidos como resultado de este proceso biológico, serán fruto de la combinación del ADN de ambos progenitores y, por tanto, serán genéticamente distintos a ellos.

La *mutación* es un término utilizado en genética y biología y se define como una alteración o cambio en la información genética (genotipo) de un ser vivo y que, por lo tanto, va a producir un cambio de características, que se presenta súbita y espontáneamente, y que se puede transmitir o heredar a la descendencia. La unidad genética capaz de mutar es el gen, el cual es la unidad de información hereditaria que forma parte del ADN. En los seres multicelulares, las mutaciones sólo pueden ser heredadas cuando afectan a las células reproductivas.

La *competencia* es la interacción que tiene lugar en una comunidad, entre individuos de especies diferentes, que coinciden por el turno que ocupan en la circulación

de energía y nutrientes o de obtención de recursos, cuando existe una demanda activa de un recurso en común que puede ser limitado. También puede establecerse la competencia entre dos poblaciones cuando escasean factores de tipo abiótico.

La reproducción sexual conlleva a la *selección* sexual, que es el proceso utilizado por los machos o hembras de una especie para seleccionar a su pareja. Este proceso frecuentemente involucra demostraciones prolongadas de vigor entre miembros de las especies, generalmente los machos. La competencia entre machos ha sido vista como una fuente importante de selección sexual desde Darwin en 1859, quien concluyó que el destino de los individuos abatidos no era precisamente la muerte, sino el quedar condenados a tener poca o ninguna descendencia. Una muerte genética tiene el mismo impacto que la muerte fisiológica para purgar un defecto genético.

En resumen, la teoría del neodarwinismo dicta que el individuo es el objetivo de la selección. Sin embargo, las especies, al menos la población en reproducción, actúan como la reserva genética del comportamiento aprendido. Los individuos se reproducen lo suficiente para ocupar los recursos de espacio disponibles. Los errores durante la réplica genética dan como resultado la variación fenotípica. Más adelante, debido a la naturaleza del programa genético, esas variaciones fenotípicas son heredables. La competencia es debido al inevitable exceso de población y conlleva a una selección estocástica en contra de los individuos que se encuentran menos adaptados al ambiente actual.

La evolución darwiniana no es más que la consecuencia inevitable de sistemas de reproducción-información que compiten entre sí, operando en un espacio finito dentro de un universo entrópico positivo.

Partiendo del panorama anterior, se facilitará la comprensión de los sistemas computacionales de optimización basados en la teoría evolutiva, así como la asimilación de conceptos utilizados dentro de estas técnicas.

1.1.2. Paradigmas de la computación evolutiva

En computación evolutiva se cuenta con diversas ramas y en años recientes se ha utilizado el término *algoritmos evolutivos* para englobar los principales tipos de técnicas contenidas en la computación evolutiva. Dichos algoritmos son las estrategias evolutivas, la programación evolutiva y los algoritmos genéticos, los cuales serán descritos brevemente a continuación.

Algoritmos evolutivos

Los algoritmos evolutivos se caracterizan por estar basados en el uso de una población de soluciones, así como la existencia de comunicación e intercambio de información entre los individuos de la misma. Dicho intercambio de información es resultado de los procesos de selección y/o recombinación efectuados entre los individuos.

La población, en el contexto de algoritmos evolutivos y en computación evolutiva en general, se refiere a un conjunto de posibles soluciones del problema que se busca

resolver. Se busca que dicha población (de soluciones) evolucione, es decir, que las soluciones mejoren en cada generación.

El algoritmo 1.1 muestra el funcionamiento general de un algoritmo evolutivo. Los operadores de búsqueda son llamados operadores genéticos y son los responsables de crear nuevos individuos a partir de los ya existentes. Un algoritmo difiere de otro dependiendo de la representación adoptada por los individuos y de los esquemas que sean utilizados para implementar la evaluación de aptitud y los operadores de selección y recombinación.

Algoritmo 1.1: Algoritmo evolutivo

Entrada: Parámetros del AE
Salida: Solución al problema
1 Inicializar $i = 0$;
2 Generar de forma aleatoria la **población** $P(i)$;
3 **repeat**
4 Evaluar la aptitud de cada individuo en $P(i)$;
5 Seleccionar a los padres de $P(i)$ basado en su aptitud;
6 Aplicar los **operadores de recombinación** a los padres y generar la población $P(i+1)$;
7 **until** *Convergencia de la población o se alcanza el tiempo máximo* ;

Estrategias evolutivas

Las estrategias evolutivas fueron propuestas de forma independiente por Rechenberg [67] y Schwefel [70], en Alemania, en colaboración con otro estudiante egresado, Peter Bienert, en 1965. Inicialmente se enfocaban a la optimización de problemas en mecánica de fluidos y problemas de dispositivos de hardware, pero después se enfocaron hacia algoritmos de optimización de funciones. La propuesta original de la estrategia evolutiva no contaba con población, pero más adelante, en 1981, fue introducida por Schwefel [71].

En este tipo de algoritmos, la representación de los individuos es muy similar a la representación natural de un problema y no enfatiza la representación genética de los individuos.

Las estrategias evolutivas generalmente usan un esquema de selección determinístico, mutación Gaussiana y recombinación discreta o intermedia.

Existen dos esquemas principales de selección determinística que son $(\mu + \lambda)$ y (μ, λ) donde μ es el tamaño de la población y λ el número de hijos generados a partir de μ padres. Los μ individuos con mejor aptitud de $\mu+\lambda$ candidatos son seleccionados para formar la siguiente generación. En las estrategias evolutivas (μ, λ) se seleccionan únicamente los μ individuos más aptos de entre los hijos λ para formar la siguiente generación. Por tanto, se requiere que $\lambda \geq \mu$.

En las estrategias evolutivas, la mutación es frecuentemente implementada agregando un número aleatorio Gaussiano a un padre. Si tenemos un padre (individuo)

$\vec{x} = (x_1, x_2, \ldots, x_n)$, entonces un descendiente será generado por mutación de la siguiente manera:

$$x'_i = x_i + N_i(0, \sigma_i) \tag{1.1}$$

donde $N_i(0, \sigma_i)$ es el valor de una variable aleatoria con distribución normal con media 0 y desviación estándar σ_i. Se generan de forma independiente n valores aleatorios (uno para cada variable del problema).

La desviación estándar σ_i es un parámetro importante en la mutación Gaussiana y el valor de este parámetro afecta de forma importante el desempeño de la estrategia evolutiva. Sin embargo el valor óptimo de este parámetro depende del problema y la dimensionalidad del mismo.

Schwefel propuso incluir σ_i como parte del individuo con el fin de mejorarlo automáticamente. A este procedimiento se le conoce como *auto-adaptación*.

En las estrategias evolutivas, la recombinación se efectúa principalmente de dos formas: recombinación discreta e intermedia. Suponiendo los vectores $\vec{x} = (x_1, x_2, \ldots, x_n)$ y $\vec{y} = (y_1, y_2, \ldots, y_n)$, la recombinación discreta genera los descendientes $\vec{x'} = (x'_1, x'_2, \ldots, x'_n)$ y $\vec{y'} = (y'_1, y'_2, \ldots, y'_n)$ de acuerdo a la siguiente ecuación:

$$x'_i = \begin{cases} x_i & \text{con probabilidad } p \\ y_i & \text{de otra forma} \end{cases} \tag{1.2}$$

$\vec{y'}$ será el complemento de $\vec{x'}$.

La recombinación intermedia se basa en promedios. Si se tienen los padres $\vec{x} = (x_1, x_2, \ldots, x_n)$ y $\vec{y} = (y_1, y_2, \ldots, y_n)$, los descendientes $\vec{x'} = (x'_1, x'_2, \ldots, x'_n)$ y $\vec{y'} = (y'_1, y'_2, \ldots, y'_n)$ se generan mediante la ecuación:

$$x' = x_i + \alpha(y_i - x_i) \tag{1.3}$$

donde $\alpha \in (0, 1)$, y representa un parámetro de ponderación. Generalmente se utiliza $\alpha = 0.5$. También se puede generar de manera aleatoria. $\vec{y'}$ se genera de forma similar.

Programación evolutiva

En una serie de estudios realizados a partir de 1962 [26, 28, 27, 29, 30, 31], Lawrence J. Fogel propuso usar una simulación de la evolución en una población de soluciones que contendían entre sí con el fin de desarrollar inteligencia artificial y exploró las posibilidades de lo que llamó *programación evolutiva*. La aplicación inicial de la programación evolutiva fue en el ramo de la inteligencia artificial, donde buscaba evolucionar máquinas de estados finitos.

A finales de los 80's, la programación evolutiva fue aplicada a problemas combinatorios y problemas de optimización numérica. Se reemplazaron las máquinas de estados finitos por representaciones basadas en el problema a resolver. Las operaciones de variación eran hechas de tal forma que se mantenía una estrecha relación de comportamiento entre padres y descendencia.

Las diferencias más importantes con respecto a las estrategias evolutivas son los operadores de recombinación y selección. La programación evolutiva no usa recombinación o cruza, y en cambio utiliza una competencia probabilística como mecanismo de selección. La programación evolutiva, cuando se aplica en optimización numérica, utiliza vectores de números reales como individuos, mutación Gaussiana y autoadaptación, de manera similar a las estrategias evolutivas.

Algoritmos genéticos

Surgieron al menos tres propuestas similares y de forma independiente en el transcurso de aproximadamente una década. Este método fue propuesto inicialmente por Fraser en 1957 [33], Bremermann utiliza un método similar en 1962 [4] y John Holland [2, 68] también realiza publicaciones del mismo en 1962.

Los algoritmos genéticos difieren con las estrategias evolutivas y la programación evolutiva en términos de la representación de individuos y los operadores de variación. Los algoritmos genéticos realizan la codificación genética de las soluciones potenciales en cromosomas. Luego, propagan copias de estos individuos basándose en un criterio externo de aptitud y por medio de mutación y recombinación se generan nuevos individuos para cada generación siguiente. Esto equivale a transformar el problema original de un espacio a otro. Por lo anterior, la representación jugará un papel crucial en el éxito o fracaso del algoritmo genético. Un punto crucial en la aplicación de algoritmos genéticos para resolver un problema es cómo encontrar la representación que ayude a que se realice de forma eficiente la búsqueda de la solución del problema.

Un algoritmo genético simple es aquél que utiliza representación binaria, un punto de cruza y mutación mediante intercambio de bits. La representación binaria significa que cada individuo será representado por un número determinado de bits, 0 ó 1. La selección se realiza de forma aleatoria asignando una probabilidad proporcional a la aptitud de cada individuo. La cruza de un punto consiste en que dadas dos cadenas binarias, x y y, de longitud n, se define un punto de cruza en el rango $[1, n-1]$ de manera uniforme y aleatoria, digamos r. De esta manera, el primer descendiente consistirá de los primeros r bits de x y de los últimos $n-r$ bits de y. El segundo descendiente consistirá de los primeros r bits de y y de los últimos $n-r$ bits de x. En la mutación mediante intercambio de bits, cada bit de un individuo tiene cierta probabilidad de ser intercambiado, es decir, si es 0, se cambia a 1 o viceversa. El algoritmo 1.2 representa a un algoritmo genético simple.

Algoritmo 1.2: Algoritmo genético

Entrada: Parámetros del AG
Salida: Solución al problema

```
1 Inicializar i = 0;
2 Generar de forma aleatoria la población P(i);
3 repeat
4     Evaluar la aptitud de cada individuo en P(i);
5     Seleccionar a los padres de P(i) con base en su aptitud ϵ;
6     Aplicar el operador de cruza a los padres seleccionados;
7     Aplicar mutación a los individuos obtenidos de la cruza;
8     Reemplazar a los padres por la descendencia para producir la generación
      P(i + 1);
9 until Alcanzar el criterio de terminación del algoritmo ;
```

1.2. Organización de la tesis

La organización de esta tesis se describe brevemente a continuación.

Capítulo 1: Introducción.

En este capítulo se da un bosquejo del problema que se aborda en la tesis y de las herramientas utilizadas para solucionarlo. Se proporciona una breve reseña de los orígenes de la computación evolutiva, sus características y sus principales paradigmas. También se presenta un panorama general del contenido de esta tesis.

Capítulo 2: Optimización multi-objetivo.

Se introducen los conceptos básicos de la optimización multi-objetivo y se describen algunos de los algoritmos evolutivos multi-objetivo más populares en la literatura especializada. También se mencionan y bosquejan los principales algoritmos utilizados en el área y se describe a detalle el algoritmo evolutivo multi-objetivo utilizado como base en el presente trabajo.

Capítulo 3: Métodos de aproximación de funciones.

Se describen los métodos de aproximación de funciones utilizados en este trabajo, asi como sus métodos de entrenamiento adoptados y su uso como predictores.

Capítulo 4: Optimización asistida por métodos de aproximación de funciones.

En este capítulo se realiza la fusión de los conceptos analizados en los capítulos previos. Se detallan los procedimientos utilizados para sacar el mayor provecho de ca-

da una de las herramientas, se muestra a detalle el algoritmo propuesto así como los resultados de la comparación hecha entre el algoritmo propuesto y uno de los mejores algoritmos del estado del arte en el área: el NSGA-II.

Capítulo 5: Conclusiones y trabajo a futuro.

Se presentan las conclusiones a las que llegamos con los resultados obtenidos. Se ofrecen también ideas sobre investigaciones en el tema, desarrolladas en el tiempo de elaboración de la tesis y que, debido a limitantes obvias de tiempo, no se desarrollaron.

Apéndice A: Funciones de prueba.

Se describen las funciones utilizadas en las comparaciones hechas entre el algoritmo propuesto y el del estado del arte.

Apéndice B: Convergencia del AGMOCAF en las funciones de prueba.

Se muestran las gráficas que prueban la convergencia del algoritmo propuesto en las funciones de prueba adoptadas.

CAPÍTULO 2

Optimización multi-objetivo

El término **optimización**, según Kalyanmoy Deb [18, pág. 1], es la acción de encontrar una o más soluciones factibles que corresponden a valores extremos de uno o más objetivos. Cuando un problema de optimización involucra solamente una función objetivo, a la tarea de encontrar la solución óptima se le llama optimización mono-objetivo. En el caso de que el problema de optimización involucre más de una función objetivo, a la tarea de encontrar una o más soluciones óptimas se le conoce como optimización multi-objetivo.

En el mundo real, los problemas de optimización requieren por naturaleza varios objetivos. El definir varios objetivos en un problema, generalmente ofrece una mejor idea de la tarea que se busca cumplir. En un conjunto de soluciones se pueden tener escenarios en conflicto entre diferentes objetivos, es decir, al mejorar uno de los objetivos, los demás objetivos se ven afectados de forma intrínseca en el problema. Por lo anterior, la solución a un problema multi-objetivo no es única, sino un conjunto de posibles soluciones dentro de las cuales no es posible considerar una solución mejor que la otra. Más adelante se explicarán algunos conceptos básicos en el área de optimización multi-objetivo para comprender el tipo de problema abordado y el correcto uso de las herramientas existentes para su solución.

2.1. Conceptos básicos

2.1.1. Optimización multi-objetivo

La optimización multi-objetivo se puede definir como:

"Encontrar un vector de variables de decisión que satisfaga las restricciones dadas y optimice un vector de funciones cuyos elementos representan las funciones objetivo. Esas funciones forman una descripción matemática de los criterios a optimizarse y generalmente se encuentran en conflicto entre sí. Por lo tanto, el término *optimizar* significa encontrar las soluciones que darían valores aceptables para todas las

funciones objetivo" [59].

Formalmente, en un problema multi-objetivo se busca el vector de variables de decisión $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ que optimice a:

$$\mathbf{F}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})], \qquad f_i : \mathbb{R}^n \to \mathbb{R} \tag{2.1}$$

sujeto a:
$g_i(\vec{x}) < 0, \qquad i = 1, 2, \ldots, m;$
$h_j(\vec{x}) = 0, \qquad j = 1, 2, \ldots, p$

donde:
k es el número de funciones objetivo,
n es el número de variables de decisión,
m es el número de restricciones de desigualdad,
p es el número de restricciones de igualdad,

2.1.2. Optimalidad de Pareto

En problemas multi-objetivo, cuando los objetivos se encuentran en conflicto entre sí, se buscan normalmente soluciones compromiso en vez de una única solución. Por lo tanto, la noción de *óptimo* es diferente en estos casos. La noción de óptimo más comúnmente adoptada es la propuesta por Francis Ysidro Edgeworth [23] y más tarde generalizada por Vilfredo Pareto [60]. Dicha noción es comúnmente conocida bajo el término de *optimalidad de Pareto*.

Se dice que un vector de variables de decisión:
$\vec{x}^* \in \mathcal{F}$ es un *óptimo de Pareto* existente en la región factible $\mathcal{F}$ si (asumiendo minimización)

$$f_i(\vec{x}^*) \leq f_i(\vec{x}), \qquad \forall i \in [1, 2, \ldots, k] \quad \text{y} \tag{2.2}$$
$$f_j(\vec{x}^*) < f_j(\vec{x}), \qquad \exists j \in [1, 2, \ldots, k] \tag{2.3}$$

En palabras, $\vec{x}^*$ es un óptimo de Pareto si no existe otro vector factible de variables de decisión $\vec{x} \in \mathcal{F}$ que mejore algún criterio sin causar el deterioro simultáneo de al menos otro criterio. Sin embargo, este concepto casi siempre ofrece no una, sino varias soluciones llamadas *conjunto de óptimos de Pareto*. Los vectores $\vec{x}^*$ correspondientes a las soluciones incluidas en el conjunto de óptimos de Pareto son llamados *no dominados*. La imagen del conjunto de óptimos de Pareto bajo las funciones objetivo es llamada *frente de Pareto*.

En la figura 2.1 se grafican los objetivos f_1 vs. f_2. Se puede apreciar que del conjunto de soluciones, los puntos p_1 y p_2 tienen el mínimo valor de f_1, sin embargo, p_1 y p_3 tienen el mínimo valor para f_2. De acuerdo a las ecuaciones (2.2) y (2.3), se puede observar que el único óptimo de Pareto del conjunto representado es p_1.

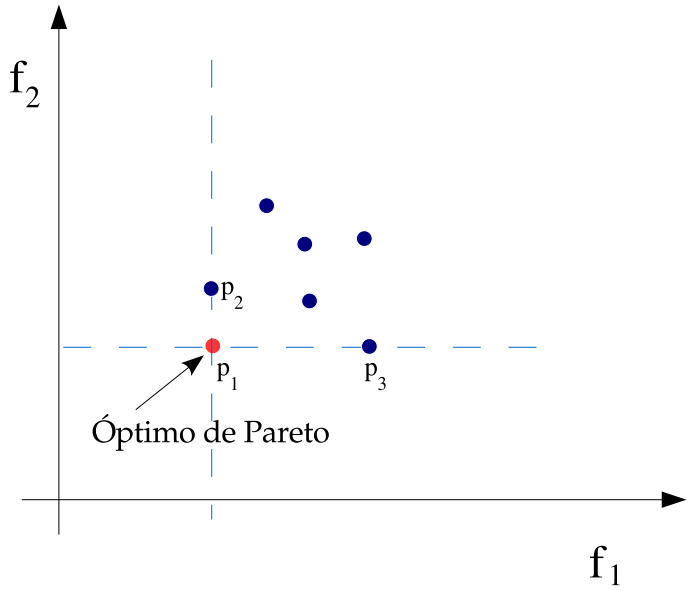

Figura 2.1: Representación de un óptimo de Pareto.

2.1.3. Dominancia de Pareto

En algoritmos multi-objetivo se utiliza frecuentemente el concepto de dominancia de Pareto al compararse dos soluciones y decidir si una domina a otra o no.

Una solución $\vec{x}_a$ se dice que domina a otra $\vec{x}_b$ si se cumplen las condiciones siguientes (para el caso de minimización) :

1. La solución $\vec{x}_a$ no es peor que $\vec{x}_b$ en todos los objetivos, o:

$$f_i(\vec{x}_a) \leq f_j(\vec{x}_b), \quad \forall i \in [1, 2, \ldots, k] \tag{2.4}$$

2. La solución $\vec{x}_a$ es estrictamente mejor que $\vec{x}_b$ en al menos un objetivo, o:

$$f_i(\vec{x}_a) < f_j(\vec{x}_b), \quad \exists i \in [1, 2, \ldots, k] \tag{2.5}$$

Si alguna de las condiciones (2.4) o (2.5) son violadas, la solución $\vec{x}_a$ no domina a la solución $\vec{x}_b$. La forma matemática de representar que $\vec{x}_a$ domine a $\vec{x}_b$ en este trabajo será mediante el operador $\prec$ de la siguiente manera:

$$\vec{x}_a \prec \vec{x}_b \tag{2.6}$$

En la figura 2.2 se muestra un conjunto de soluciones y se hace una distinción entre los elementos que son dominados y los que no lo son. Como puede apreciarse, el punto p_1 tiene el menor valor para f_1 y p_5 tiene el menor valor con respecto a f_2. Como ninguna otra solución podrá mejorarlos, éstos son elementos no dominados. Ahora, para los puntos p_2 y p_6 observamos que tienen el mismo valor en f_1; sin embargo, p_2 tiene un valor menor en f_2, por lo tanto domina a p_6. Para p_3 observamos que no existe una solución que tenga valores menores en ambos ejes.

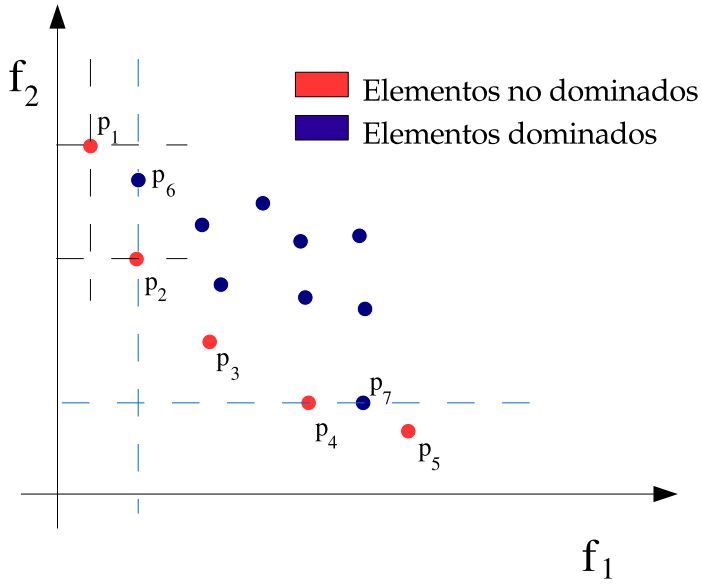

Figura 2.2: Elementos dominados y no dominados en un conjunto.

2.1.4. Frente de Pareto

Para una función multi-objetivo $F(\vec{x})$ dada y un conjunto de óptimos de Pareto Ω, el frente de Pareto $\mathcal{FP}^*$ se define como:

$$\mathcal{FP}^* = \{\mathbf{F}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \cdots, f_k(\vec{x})) \,|\vec{x} \in \mathcal{F}\} \tag{2.7}$$

2.1.5. Categorías de funciones objetivo

En problemas multi-objetivo, las funciones objetivo pueden ser categorizadas como *totalmente en conflicto, sin conflicto entre sí o parcialmente en conflicto*. En un conjunto Φ de soluciones dado, un vector de funciones objetivo $\mathbf{F} = \{f_1, f_2, \ldots, f_k\}$ se dice que está *totalmente en conflicto* si no existen dos soluciones $\vec{x}_a$ y $\vec{x}_b$ en un conjunto Φ tal que $(\mathbf{F}(\vec{x}_a) \prec \mathbf{F}(\vec{x}_b)) \vee (\mathbf{F}(\vec{x}_b) \prec \mathbf{F}(\vec{x}_a))$. Esta clase de problemas multi-objetivo no requieren ser optimizados debido a que el conjunto Φ ya representa el conjunto global de óptimos de Pareto.

Por otra parte, las funciones objetivo se dicen estar *sin conflicto entre sí* cuando cualquier par de soluciones $\vec{x}_a$ y $\vec{x}_b$ en un conjunto Φ satisfacen $(\mathbf{F}(\vec{x}_a) \prec \mathbf{F}(\vec{x}_b)) \vee (\mathbf{F}(\vec{x}_b) \prec \mathbf{F}(\vec{x}_a))$. Esta clase de problemas multi-objetivo pueden ser convertidos fácilmente en mono-objetivo ya sea considerando sólo uno de los objetivos o combinando los objetivos en una función escalar. En este caso, cualquier mejora en uno de los objetivos que lo componen implica la mejora de los otros objetivos y viceversa. El tamaño del conjunto de óptimos de Pareto es igual a uno en este tipo de problemas.

En el tercer caso, un vector de funciones objetivo $\mathbf{F} = \{f_1, f_2, \ldots, f_k\}$ se dice tener funciones objetivo *parcialmente en conflicto* si existen dos conjuntos de soluciones $\mathbf{P}_a$

y $\mathbf{P}_b$, los cuales cuentan con al menos un elemento cada uno $\vec{x}_a \in \mathbf{P}_a, \quad \vec{x}_b \in \mathbf{P}_b$ que cumplen con $(\mathbf{F}(\vec{x}_a) \prec \mathbf{F}(\vec{x}_b)) \vee (\mathbf{F}(\vec{x}_b) \prec \mathbf{F}(\vec{x}_a))$. En esta categoría se encuentran muchos problemas del mundo real, donde se busca al conjunto de óptimos de Pareto que representan los compromisos entre los objetivos en conflicto [76, pág. 3].

2.1.6. Objetivo de la optimización multi-objetivo

Cuando se tiene un problema con dos o más objetivos en conflicto, generalmente el conjunto de soluciones óptimas contienen más de un elemento, por lo tanto, la tarea de elegir una solución entre las otras se vuelve una tarea difícil si no se cuenta con información adicional del problema. En estos casos, todas las soluciones que representan un óptimo de Pareto tienen la misma importancia, por lo que es necesario encontrar tantas soluciones como sea posible, que sean óptimos de Pareto. Por lo tanto, se puede llegar a la conclusión de que existen dos propósitos en la optimización multi-objetivo:

1. Encontrar un conjunto de soluciones lo más cercanas posible al conjunto de óptimos de Pareto.

2. Encontrar un conjunto de soluciones lo más diversas posible.

El primer punto es obligatorio en cualquier tarea de optimización debido a que las soluciones carecen de utilidad si no pertenecen al conjunto de óptimos de Pareto. Únicamente cuando las soluciones convergen al conjunto de óptimos de Pareto (o suficientemente cerca a ellas) es cuando nos interesa examinar su diversidad.

En la figura 2.3 se clasifican las soluciones como deseables y no deseables tomando en cuenta qué tan cercanas se encuentran al frente de Pareto verdadero del problema de optimización.

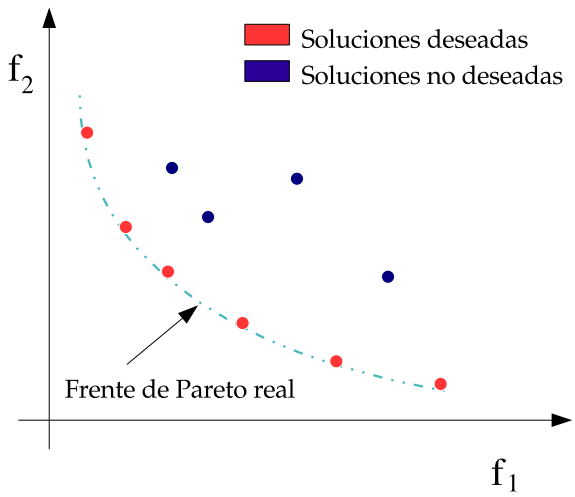

Figura 2.3: Clasificación de soluciones de acuerdo a su cercanía al frente de Pareto.

Por otro lado, el segundo punto es una característica deseada de manera particular para problemas multi-objetivo. Se desea obtener soluciones que además de pertenecer al conjunto de óptimos de Pareto, deben también encontrarse distribuidas lo más uniformemente posible. De esta forma se puede asegurar que se tiene un buen conjunto de soluciones.

En la figura 2.4 se observan dos conjuntos de soluciones; las que se encuentran mejor distribuidas son preferidas sobre aquellas que se encuentran muy cercanas entre sí, inclusive cuando ambos conjuntos de soluciones están sobre el frente de Pareto.

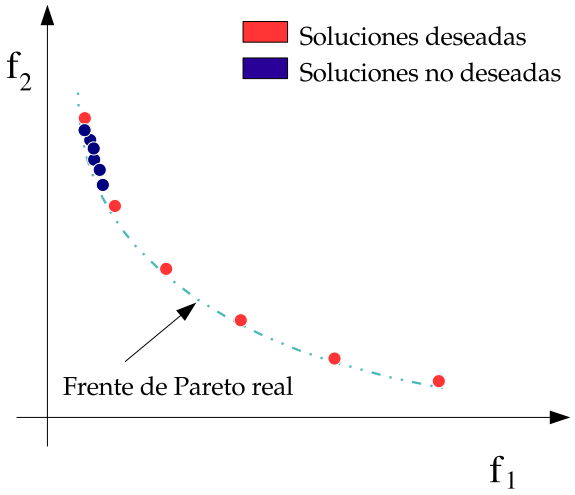

Figura 2.4: Clasificación de soluciones de acuerdo a su distribución en el frende de Pareto.

2.1.7. Espacios de búsqueda

En optimización multi-objetivo se involucran dos espacios de búsqueda, a diferencia de la optimización mono-objetivo donde solamente existe uno, que es el espacio de las variables de decisión. En optimización multi-objetivo existe, además del espacio de las variables de decisión, otro llamado espacio de los objetivos o criterios [18, págs. 24-25]. Aunque los espacios de las variables de decisión y el de los objetivos se encuentran relacionados por un mapeo directo entre ellos, este mapeo no es lineal en la mayoría de los casos y las propiedades de los dos espacios no son similares. Por ejemplo, la proximidad entre dos soluciones en un espacio no significa que exista una proximidad similar en el otro espacio. La meta de diversidad en los problemas de optimización multi-objetivo puede realizarse en el espacio de variables de decisión o en el de objetivos dependiendo del problema; sin embargo, en la mayoría de los casos se realiza en el espacio de los objetivos.

2.2. Algoritmos multi-objetivo

Existen métodos clásicos utilizados para solucionar problemas de optimización multi-objetivo, como lo son el método de las sumas ponderadas, el método de restricciones-ϵ, el método de las métricas ponderadas, el método de Benson, el método de la función de utilidad, entre otros [18, págs. 47-75]. Sin embargo, los métodos clásicos cuentan con la desventaja de que generan una sola solución por cada ejecución algorítmica. Además, algunos de estos algoritmos no pueden generar porciones no convexas del frente de Pareto (p.ej., las sumas ponderadas lineales [18, pág. 73]). Los algoritmos clásicos requieren además de cierto conocimiento previo del problema para poder establecer los parámetros que el algoritmo requiere. La mayoría de los algoritmos clásicos sugieren una forma de transformar problemas multi-objetivo a mono-objetivo.

También existen las técnicas de optimización estocástica, como el recocido simulado, búsqueda tabú, optimización mediante colonia de hormigas, etc., las cuales pueden utilizarse para resolver problemas de optimización. Sin embargo, debido a la forma en que trabajan dichos algoritmos, las soluciones que generan no necesariamente son las mejores posibles y, en general, su optimalidad no puede garantizarse.

Dentro de las técnicas de optimización estocástica se encuentran los algoritmos evolutivos, que se caracterizan por contar con una población de posibles soluciones y un proceso de reproducción, el cual permite la combinación de soluciones existentes para generar nuevas soluciones. Los algoritmos evolutivos cuentan con una técnica de selección (que imita a la selección natural), la cual determina qué individuos de la población en turno participan en la nueva población. Este proceso permite encontrar varios miembros del conjunto de óptimos de Pareto en una sola ejecución del algoritmo, en lugar de tener que realizar una serie de ejecuciones por separado, como ocurre con los algoritmos clásicos y con técnicas estocásticas no poblacionales (p.ej., el recocido simulado).

2.2.1. Algoritmos evolutivos multi-objetivo

La primer implementación de lo que hoy llamamos algoritmos evolutivos multi-objetivo (AEMOs), fue realizada por Schaffer con su algoritmo genético de evaluación vectorial (VEGA), introducido a mediados de los años 80 y que buscaba resolver problemas de aprendizaje de máquina [69]. Desde entonces se han propuesto una amplia variedad de algoritmos en la literatura [11, 8, 10], los cuales, de forma general, los podemos dividir en:

- Funciones de agregación
- Métodos poblacionales
- Métodos basados en jerarquización de Pareto

Funciones de agregación

Las *funciones de agregación* son quizá el método más simple para manejar múltiples objetivos y consiste en combinar todos los objetivos en uno solo usando operaciones aritméticas tales como la suma o multiplicación. Estas técnicas son conocidas como "funciones de agregación" porque combinan todos los objetivos del problema en uno solo. Las funciones de agregación pueden ser lineales o no lineales y ambos tipos han sido utilizados con éxito relativo en múltiples ocasiones. Sin embargo, han sido poco apreciadas por investigadores de optimización evolutiva multi-objetivo debido a sus limitantes (no pueden generar porciones no convexas del frente de Pareto). Sin embargo, es importante hacer notar que las funciones de agregación no lineales no necesariamente presentan esta limitante [11]. De hecho, incluso funciones de agregación lineal pueden ser definidas de tal forma que puedan generar frentes de Pareto cóncavos [40]. A pesar de esto, este tipo de métodos tienen poca popularidad entre los investigadores de optimización evolutiva multi-objetivo.

Métodos poblacionales

En este caso, con el fin de diversificar la búsqueda en un algoritmo evolutivo, se utiliza la población del mismo para explorar distintos tipos de soluciones, aunque el concepto de dominancia de Pareto no se encuentra directamente incorporado en el proceso de selección. Un ejemplo clásico de este tipo de métodos es VEGA [69]. VEGA consiste básicamente de un algoritmo genético simple con un mecanismo de selección modificado. En cada generación, se genera un número de sub-poblaciones aplicando selección proporcional de acuerdo a la función objetivo en turno. De esta forma, para problemas con k objetivos, k sub-poblaciones de tamaño M/k cada una, son generadas (asumiendo un tamaño de población total de M individuos). Estas sub-poblaciones son mezcladas entre sí para obtener una nueva población de tamaño M, en la cual el algoritmo genético aplica los operadores de cruza y mutación.

VEGA tiene varios problemas, de los cuales, el más serio es que su esquema de selección es opuesto al concepto de dominancia de Pareto. Por ejemplo, si un individuo codifica una buena solución compromiso para todos los objetivos, pero no es la mejor en ninguno de ellos, ésta es descartada. Evidentemente, dicho individuo debería de ser preservado porque codifica una solución Pareto-óptima. Schaffer sugirió algunas heurísticas para lidiar con este problema; sin embargo, el hecho de que la dominancia de Pareto no se encuentra directamente incorporada en el proceso de selección del algoritmo continúa siendo una desventaja.

Un aspecto interesante de VEGA es que continúa siendo utilizado por algunos investigadores principalmente porque es apropiado para problemas en los que necesitemos lidiar con un gran número de objetivos [12].

Métodos basados en jerarquización de Pareto

Los *métodos basados en jerarquización de Pareto* toman como base la principal desventaja de VEGA. Su funcionamiento consiste en un esquema de selección ba-

sado en el concepto de optimalidad de Pareto. Este tipo de método fue propuesto originalmente por Goldberg [16], quien también indicó que el ruido estocástico haría inservible este tipo de algoritmo, a menos que se adoptara algún mecanismo especial para bloquear la convergencia a una solución única. Goldberg sugirió el método de nichos o el de compartición de aptitud para mantener la diversidad y evitar la convergencia del algoritmo genético a una sola solución.

Los métodos basados en jerarquización de Pareto pueden dividirse históricamente en dos generaciones. La primera generación se caracteriza por el uso de compartición de aptitud y nichos combinados con un mecanismo de selección basado en optimalidad de Pareto. Los algoritmos más representativos de esta primera generación son los siguientes:

1. *Non-dominated Sorting Genetic Algorithm (NSGA)*: Este algoritmo fue propuesto por Srinivas y Deb [73]. El método se basa en varias capas de clasificación de individuos como sugiere Goldberg [35]. Antes de realizar la selección, la población se ordena de acuerdo a su no-dominancia, es decir, todos los individuos no-dominados son clasificados en una categoría (con un valor de aptitud proporcional al tamaño de población, para ofrecer la misma posibilidad de reproducción para dichos individuos). Con el fin de mantener la diversidad de la población, estos individuos ya clasificados son mezclados con sus valores de aptitud. Luego, este grupo de individuos clasificados se ignoran de la población total para obtener la otra capa de individuos no-dominados. El proceso continúa hasta que todos los individuos de la población hayan sido clasificados. El mecanismo de selección utilizado es el método del sobrante estocástico. Debido a que los individuos del primer frente son los que obtienen el valor máximo de aptitud, éstos se copian en mayor proporción comparados con los demás individuos de la población. Esto permite guiar la búsqueda hacia regiones no dominadas y produce la convergencia de la población hacia dichas regiones. La mezcla ayuda a distribuir la población sobre el frente de Pareto del problema.

2. *Niched-Pareto Genetic Algorithm (NPGA)*: Fue propuesto por Horn et al. [39]. El NPGA usa un esquema de selección por torneo basado en la dominancia de Pareto. La idea básica del algoritmo es la siguiente: Dos individuos se seleccionan de forma aleatoria y se comparan contra un subconjunto de la población total (típicamente alrededor del 10 % de la población). Si uno de ellos es dominado (por los individuos seleccionados aleatoriamente de la población) y el otro no, gana este último. Cuando ambos competidores son dominados o no-dominados, el resultado del torneo se decide mediante compartición de aptitud (es decir, el individuo que resida en la región menos poblada del espacio de búsqueda es el ganador).

3. *Multiobjective Genetic Algorithm (MOGA)*: Fue propuesto por Fonseca y Fleming [32]. En MOGA, la clasificación de un individuo determinado corresponde al número de individuos en la población actual que lo dominan. A todos los

individuos que son no-dominados se les asigna el valor de 1 en la clasificación. La asignación de aptitud se realiza de la siguiente manera:

a) Se ordena la población de acuerdo a su clasificación.

b) Se asigna la aptitud a los individuos mediante la interpolación del mejor (valor de 1) al peor (valor de $n \leq M$) de acuerdo a una función, generalmente lineal, aunque no necesariamente.

c) Se realiza un promedio de aptitud con los individuos de la misma clasificación, de forma que todos ellos se muestreen con la misma proporción. Este procedimiento mantiene constante la aptitud de la población global y a su vez mantiene una presión de selección apropiada, definida por la función utilizada.

La segunda generación de AEMOs nació con la introducción de la noción de elitismo. En el contexto de optimización multi-objetivo, generalmente se utiliza el término *elitismo* para hacer referencia al uso de una población externa (también llamada población secundaria) con el fin de retener los individuos no-dominados. Por lo que, el uso de dicho archivo externo, despierta varias inquietudes, como son:

- ¿Cómo interactúa el archivo externo con la población principal?
- ¿Qué se debe hacer cuando el archivo externo se llene?
- ¿Se debe imponer un criterio adicional para agregar elementos al archivo en lugar de sólo utilizar la dominancia de Pareto?

Nótese que el elitismo también puede ser introducido mediante el uso de selección $(\mu + \lambda)$ en la cual los padres compiten con los hijos y aquellos que son no-dominados se seleccionan para la próxima generación.

Los AEMOs más representativos de la segunda generación son los siguientes:

1. *Strength Pareto Evolutionary Algorithm (SPEA)*: Este algoritmo fue introducido por Zitzler y Thiele [84]. Este método fue concebido como una forma de integrar distintos AEMOs. SPEA usa un archivo que contiene soluciones no-dominadas previamente encontradas (el llamado conjunto externo no-dominado). En cada generación, individuos no-dominados se copian a un conjunto externo. Para cada individuo en este conjunto externo, se calcula un valor de *fuerza*. Este valor es similar al valor de clasificación del MOGA, ya que es proporcional al número de soluciones a las que un determinado individuo domina.

   En SPEA, el valor de aptitud para cada miembro de la población en curso se calcula de acuerdo a las fuerzas de todas las soluciones externas no-dominadas que dominan a dicho individuo. Además, una técnica de agrupamiento llamada "método de vinculación promedio" [55] se usa para mantener la diversidad.

2. *Strength Pareto Evolutionary Algorithm 2 (SPEA2)*: Este algoritmo tiene tres principales diferencias respecto a su predecesor [83]:

a) Incorpora una estrategia de asignación de aptitud de grano fino, la cual toma en cuenta, para cada individuo, el número de individuos que domina y el número de individuos por los cuales es dominado.

b) Usa la técnica de estimación de densidad del vecino más cercano, que guía la búsqueda de forma más eficiente.

c) Tiene un método mejorado de truncamiento del archivo externo que garantiza la preservación de soluciones de frontera.

3. *Pareto Archived Evolution Strategy (PAES)*: Este algoritmo fue introducido por Knowles y Corne [42]. PAES consiste de una estrategia evolutiva del tipo $(1+1)$ (o sea, un solo padre que genera un solo hijo) en combinación con un archivo histórico que guarda algunas de las soluciones no-dominadas encontradas previamente. Este archivo es usado como conjunto de referencia y cada individuo mutado es comparado contra dicho conjunto. Un aspecto interesante del algoritmo es el proceso para mantener diversidad que consiste en un procedimiento de densidad de población que divide el espacio de objetivos de forma recursiva. Cada solución es localizada en una determinada celda de una malla basándose en los valores de sus objetivos (que se usan como coordenadas o "localización geográfica"). Un mapa de esta malla se mantiene, indicando el número de soluciones que residen en cada celda de la malla. Debido a que el procedimiento es adaptativo, no se requieren parámetros extra (a excepción del número de divisiones de la malla, las cuales se efectúan en el espacio de las funciones objetivo).

4. *Non-dominated Sorting Genetic Algorithm II (NSGA-II)*: Deb et al. [20] propusieron una versión mejorada del NSGA [73], llamada NSGA-II, que es más eficiente (computacionalmente hablando), usa elitismo y un operador de comparación de densidad de población que mantiene la diversidad sin requerir otro parámetro adicional. El NSGA-II no utiliza un archivo externo como los algoritmos previos, sino que combina los mejores padres con los mejores hijos obtenidos (es decir, usa una selección $(\mu + \lambda)$).

5. *Niched Pareto Genetic Algorithm 2 (NPGA 2)*: Erickson et al. [24] propusieron una versión corregida del NPGA [39] llamada NPGA 2. Este algoritmo utiliza jerarquización de Pareto pero mantiene la selección por torneo (utilizando la aptitud para decidir la selección en caso de empate). En este caso no se utiliza un archivo externo y el mecanismo de elitismo es similar al adoptado en el NSGA-II. El conteo de nichos en el NPGA 2 es calculado usando individuos en la siguiente generación parcialmente llena en lugar de usar la generación actual. A este mecaniso se le llaman repartición de aptitud continuamente actualizada y fue propuesto por Oei et al. [58].

6. *Pareto Envelope-based Selection Algorithm (PESA)*: Este algoritmo fue propuesto por Corne et al. [14]. El método utiliza una población interna pequeña y una población externa (o secundaria) más grande. PESA usa la misma división de

hiper-malla del espacio del fenotipo (o sea, la función objetivo) adoptada por PAES para mantener diversidad. Su mecanismo de selección se basa en la métrica de densidad de población usada por la hiper-malla previamente mencionada. Esta misma medida de densidad de población es usada para decidir qué solución introducir en la población externa (el archivo de vectores no-dominados encontrados a lo largo del proceso evolutivo). Por lo mismo, el archivo externo juega un papel crucial en el algoritmo ya que determina no sólo el esquema de diversidad, sino que también la selección realizada por el método. También existe una versión corregida de este algoritmo, llamada PESA-II [13]. Este algoritmo es idéntico al PESA, excepto por el hecho de que utiliza una selección basada en regiones. En esta selección, la unidad de selección es un hipercubo en vez de un individuo. El proceso consiste en seleccionar (usando cualquiera de las técnicas tradicionales de selección) un hipercubo y después elegir al azar a cualquier individuo dentro del hipercubo. La principal motivación de este algoritmo es reducir el costo computacional asociado con los algoritmos evolutivos multi-objetivo tradicionales (es decir, aquellos basados en clasificación de Pareto).

7. *Micro-Genetic Algorithm*: Este método fue introducido por Coello Coello y Toscano Pulido [9, 6]. Un micro-algoritmo genético es un algoritmo genético con una población muy pequeña (no más de 5 individuos) y un proceso de reinicialización. La forma en que funciona el micro-algoritmo genético es la siguiente: Primero se genera una población aleatoria. Esta población alimenta a la memoria de población, que es dividida en dos partes: una porción reemplazable y otra no-reemplazable. La porción no-reemplazable de la memoria de población no cambia durante toda la ejecución del algoritmo y su función es proveer la diversidad requerida por el algoritmo. En contraste, la porción reemplazable experimenta cambios después de cada ciclo del micro-algoritmo genético.

   La población del micro-algoritmo genético al principio de cada uno de sus ciclos se toma de ambas porciones de la memoria de población, de forma que exista una mezcla de individuos generados aleatoriamente (la porción no-reemplazable) y de individuos evolucionados (la porción reemplazable). Durante cada ciclo, el micro-algoritmo genético aplica los operadores genéticos convencionales. Después de que el micro-algoritmo genético termina un ciclo, dos vectores no-dominados se eligen de la población final y se comparan con las soluciones de la memoria externa (esta memoria está inicialmente vacía). Si alguno de ellos (o ambos) permanecen no-dominados después de compararlos con los vectores de esta memoria externa, entonces se agregarán a ésta. Este es el registro histórico de los vectores no-dominados. Todos los vectores dominados contenidos en la memoria externa se eliminan.

Una vez revisadas las características de los principales algoritmos para resolver problemas multi-objetivo, resta decidir cuál es el adecuado para ser utilizado como base para nuestro estudio, cuyo objetivo es reducir el número de evaluaciones a la

función objetivo.

Para reducir las evaluaciones de la función objetivo, la idea básica es optimizar un modelo aproximado de la función objetivo verdadera. Luego, se usa un número reducido de las mejores soluciones obtenidas para realizar algunas evaluaciones de la función verdadera. Con los puntos evaluados directamente se busca actualizar y volver más preciso el modelo aproximado. El proceso convergerá en el momento en que la función aproximada sea precisa en su predicción y se evalúen los óptimos de la función objetivo real.

Con base en la descripción anterior, las características que se demandan del algoritmo evolutivo multi-objetivo son las siguientes:

1. *Tener baja complejidad computacional*: El algoritmo híbrido final requerirá de varias ejecuciones del motor de búsqueda que se elija, por lo que se busca un algoritmo con una complejidad razonablemente baja.

2. *Ser un algoritmo de la segunda generación*: Además de buscar soluciones basadas en el frente de Pareto, el usar elitismo es una característica deseable para evitar desechar soluciones que nos sean útiles a lo largo de la búsqueda, por lo que se prefiere este tipo de algoritmos sobre los de primera generación.

3. *Ofrecer soluciones óptimas y bien distribuidas*: En nuestra propuesta se utilizarán nuevas soluciones para evaluarlas directamente con las funciones objetivo reales. Estas evaluaciones servirán para mejorar el modelo aproximado. Sin embargo, para realizar una mejor aproximación, es necesario que las soluciones se encuentren lo más dispersas posibles.

4. *Utilizar operadores de selección modulares*: El algoritmo híbrido contará con un archivo externo de soluciones evaluadas directamente Sin embargo, para ingresar a este archivo, se buscará adoptar el esquema de selección utilizado por el algoritmo multi-objetivo elegido.

Por lo que, al realizar una evaluación de las características de los principales algoritmos evolutivos multi-objetivo, se seleccionó el NSGA-II, el cual cuenta con una complejidad $O(mN^2)$ con m objetivos y un tamaño de población N. Es un algoritmo que usa una selección basada en jerarquización de Pareto y lleva implícito el elitismo; por lo tanto, es un algoritmo de segunda generación adecuado para el fin que perseguimos. De acuerdo a sus características de selección y asignación de aptitud, el NSGA-II encuentra soluciones óptimas y bien distribuidas de manera eficiente y eficaz.

Es por las razones anteriores que el NSGA-II será el algoritmo base utilizado en este trabajo. Por lo tanto, se dará una descripción más detallada de su funcionamiento en la sección siguiente.

2.3. NSGA-II

Este algoritmo fue propuesto por Deb et al. en el año 2000 [20]. A diferencia de otros métodos, donde sólo se utiliza una estrategia de preservación elitista, el NSGA-II también hace uso de un mecanismo explícito de preservación de diversidad. Este algoritmo tiene poca similitud con su predecesor (el NSGA [73]), sin embargo los autores mantuvieron el nombre para resaltar su origen.

En el NSGA-II, la población de hijos Q_t es creada a partir de la población padre P_t. Sin embargo, en lugar de encontrar únicamente el frente no dominado de Q_t, combina ambas poblaciones (padres e hijos), para formar R_t de tamaño $2N$. Se utiliza un ordenamiento basado en no dominancia para clasificar la población R_t. Esto requiere más esfuerzo comparado con realizar únicamente el ordenamiento de la población Q_t. Sin embargo, permite una verificación global de elementos no-dominados en las poblaciones de padres e hijos. El proceso anterior se puede apreciar en la figura 2.5.

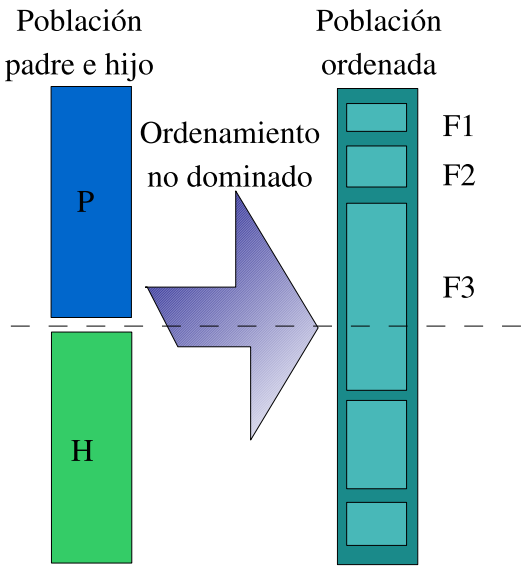

Figura 2.5: Ordenamiento no dominado en el NSGA-II.

Una vez terminado el ordenamiento no-dominado, la nueva población es llenada con soluciones de diferentes frentes no-dominados, uno a la vez. El llenado comienza con el mejor frente no-dominado y continúa con las soluciones del segundo frente no-dominado, y así sucesivamente. Como la población de R_t es de tamaño $2N$, no todos los frentes podrán ser incorporados en la nueva población de tamaño N, por lo que todos los frentes que no puedan agregarse a la nueva población serán borrados. Cuando el último frente permitido sea considerado, puede ocurrir que existan más soluciones en el último frente que el total de localidades disponibles en la nueva población. En este caso, en lugar de descartar de forma arbitraria a los miembros sobrantes del último frente, se utiliza una estrategia de nichos para escoger los miembros del último frente que formarán parte de la nueva población; éstos serán los que residan en las regiones menos pobladas en dicho frente.

Este proceso se ejemplifica en la figura 2.6, donde se aprecia que el frente que no puede agregarse completo a la siguiente población de tamaño N, es ordenado utilizando el operador de distancia de densidad de población y de esa forma, los elementos que tengan mayor distancia son agregados a la siguiente población y los demás son desechados.

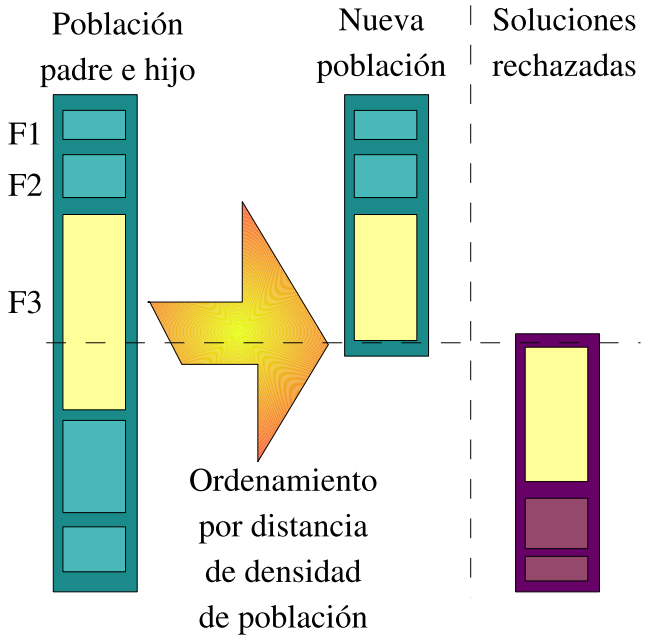

Figura 2.6: Uso del operador distancia de densidad de población en el NSGA-II.

La estrategia antes mencionada no afecta notablemente el desempeño del algoritmo al principio de la evolución, ya que en esta etapa, existen varios frentes en la población combinada R_t. Es probable que varios frentes se encuentren incluidos en la nueva población antes de sumar N soluciones. En este caso no importa tanto qué elementos sean utilizados para completar la población. Sin embargo, durante las últimas etapas de la simulación, es probable que la mayor parte de las soluciones en la población se encuentren en los mejores frentes no-dominados. Es también probable que en la población combinada R_t de tamaño $2N$, el número de soluciones en el primer frente no-dominado exceda N. Es en este caso cuando el algoritmo antes descrito asegura que se elegirá un conjunto diverso de soluciones de los elementos no-dominados del conjunto anterior.

El funcionamiento general del algoritmo es el siguiente: Primero se crea una población inicial P_0 de forma aleatoria. La población es ordenada en diferentes niveles de no-dominancia. A cada solución se le asigna un valor de aptitud igual a su nivel de no-dominancia (el mejor nivel es 1). De esta forma, se busca obtener una minimización en dicho valor. Se crea una población de hijos Q_0 de tamaño N mediante los operadores de selección por torneo binario, recombinación y mutación. Después de la primera generación, el procedimiento cambia. El procedimiento de elitismo para

$t \geq 1$ en una generación en particular se muestra en el algoritmo 2.1.

Algoritmo 2.1: Elitismo

Entrada: P_t, Q_t
Salida: P_{t+1}

```
1  R_t = P_t ∪ Q_t;
2  F = ordenamiento-basado-en-no-dominancia,   F = (F_1, F_2, ...);
3  i = 1;  while |P_{t+1}| < N do
4      Asignar-distancia-de-densidad-de-población(F_i);
5      P_{t+1} = P_{t+1} ∪ F_i; i = i + 1;
6  end
7  Ordena(P_{t+1}, ≥_n);
8  P_{t+1} = P_{t+1}[0 : N];
9  Q_{t+1} = genera-nueva-población (P_{t+1});
10 t = t + 1;
```

Primero se crea una población combinada $R_t = P_t \cup Q_t$ de tamaño $2N$. La población R_t se ordena conforme a la no dominancia de sus elementos. La nueva población P_{t+1} se forma al agregar soluciones comenzando con las del primer frente, después las del segundo y así sucesivamente hasta que el tamaño de la población exceda N. A continuación se ordena la población P_{t+1} de acuerdo al operador de comparación de población, después de lo cual, se mantienen únicamente los primero N individuos. De esta forma se obtiene la población P_{t+1} de tamaño N. A esta población se le aplicarán los operadores de selección, cruza y mutación para crear una nueva población Q_{t+1} de tamaño N.

2.3.1. Operador de comparación de población

El operador de comparación de población $<_n$ compara dos soluciones y retorna al ganador del torneo. Este operador presupone que cada solución tiene dos atributos:

1. Una clasificación de no-dominación r_i en la población.

2. Una medida de la distancia de densidad de población d_i.

La distancia de densidad de población d_i de una solución i es la medida del espacio de búsqueda alrededor de i que no es ocupado por ninguna otra solución en la población. Basado en estos dos atributos, se puede definir el operador de selección por torneo poblacional como: *Una solución i ganará un torneo contra otra solución j si cualquiera de las siguientes condiciones son verdaderas:*

1. *Si la solución i tiene un mejor rango o clasificación, es decir,* $r_i < r_j$.

2. *Si tienen el mismo rango pero la solución i tiene una mejor distancia de densidad de población que la solución j, es decir,* $r_i = r_j$ *y* $d_i > d_j$.

La primera condición asegura que las soluciones seleccionadas caigan en un mejor frente no-dominado. La segunda condición resuelve el empate entre ambas soluciones al encontrarse en el mismo frente, utilizando la distancia de población local. La que resida en un área menos poblada (con mayor densidad de población d_i) ganará. La distancia de densidad de población d_i en el NSGA-II se realiza con complejidad $O(MN \log N)$. El peor de los casos se presenta cuando todas las soluciones se encuentran en un mismo frente.

2.3.2. Distancia de densidad de población

Para obtener un estimado de la densidad de las soluciones que rodean a una solución i en particular dentro de la población, se toma la distancia promedio de dos soluciones en ambos lados de la solución i a lo largo de cada objetivo. Esta cantidad d_i es proporcional al perímetro del cuboide formado por las soluciones más cercanas a i como vértices; a esta cantidad se le llama *distancia de densidad de población*. El algoritmo 2.2 se utiliza para calcular la distancia de densidad de población en cada punto en el conjunto $\mathcal{F}$.

Algoritmo 2.2: Distancia de densidad de población

Entrada: $\mathcal{I}$
Salida: asigna distancias a los elementos de $\mathcal{I}$

```
1  l = |I|
2  foreach i do
3      I[i]_distancia = 0
4  end
5  foreach m do objetivo
6      I = ordena(I, m)
7      I[1]_distancia = I[l]_distancia = ∞
8      for i = 2 to i = (l − 1) do
9          I[i]_distancia = I[i]_distancia + (I[i + 1] · m − I[i − 1] · m)
10     end
11 end
```

En la notación del algoritmo 2.2, $\mathcal{I}[i] \cdot m$ representa el m-ésimo valor de la función objetivo del individuo i en el conjunto $\mathcal{I}$.

La figura 2.7 muestra gráficamente la distancia de densidad de población para una solución i con dos objetivos f_1 y f_2. Se ilustra el cuboide formado por las soluciones $(i-1)$ e $(i+1)$, que sirven como vértices. De esta manera, cuando las soluciones se encuentran próximas entre sí, el valor de la distancia de densidad de población será pequeño. Al preferirse las soluciones cuyo valor asignado es mayor, se hace una selección de los vectores más dispersos.

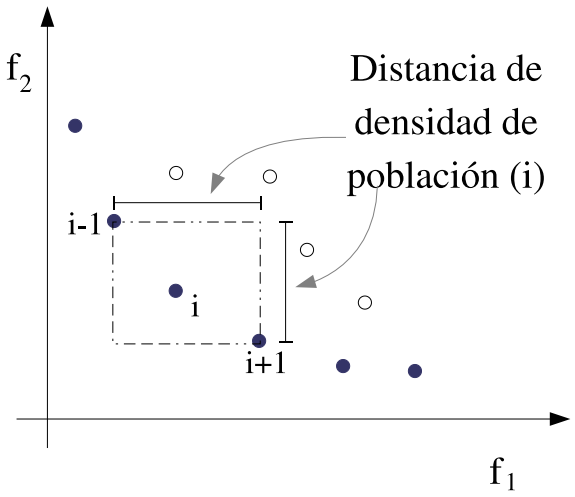

Figura 2.7: Cálculo de la distancia de densidad de población.

Esta métrica denota la mitad del perímetro del cuboide que tiene como vértices a las soluciones más cercanas al punto en cuestión. Nótese que para cualquier solución i las mismas soluciones $(i+1)$ e $(i-1)$ no necesitan ser vecinos en todos los objetivos, particularmente para $M \geq 3$.

CAPÍTULO 3

Métodos de aproximación de funciones

En este trabajo se busca combinar las propiedades de interpolación de las técnicas de aproximación de funciones con los algoritmos evolutivos buscando predecir el resultado de soluciones no evaluadas para guiar la búsqueda del algoritmo evolutivo sin la necesidad de realizar un número excesivo de evaluaciones de la función objetivo.

La teoría de interpolación se ocupa de encontrar una función obtenida mediante una tabla de otra función [74, pág 1]. De esta segunda se tienen los valores $f(x_0), f(x_1), \ldots, f(x_n)$ para los valores correspondientes $x_0, x_1, \ldots, x_n$. Con los valores mencionados podemos formar la tabla 3.1.

x_0	$f(x_0)$
x_1	$f(x_1)$
$\vdots$	$\vdots$
x_n	$f(x_n)$

Tabla 3.1: Datos para interpolación.

La tarea que se persigue es encontrar, aunque sea de forma aproximada, el valor de $f(x)$ para un argumento no contenido en la tabla.

Cabe mencionar que el problema de interpolación, en el sentido estricto de la palabra, no puede resolverse conociendo únicamente los datos de la tabla comentada, sino que es indispensable tener al menos una idea general de las características de la función. Por lo tanto, se debe seleccionar la técnica de aproximación cuyas características sean apropiadas para el problema abordado. Como en la mayoría de los casos se desconocen tales características, se proponen distintas técnicas de aproximación de funciones para poder seleccionar alguna, con el fin de obtener las mejores soluciones en el problema multi-objetivo seleccionado.

Para entender de forma general el funcionamiento de los métodos de aproximación de funciones, en este capítulo se hará una breve revisión del funcionamiento, carac-

terísticas y bondades de dos técnicas de interpolación: las funciones base radiales y el método kriging.

3.1. Redes de funciones base radiales

Este tipo de técnica es una red neuronal artificial con características específicas. Fue motivada por la respuesta de sintonía local mostrada en neuronas biológicas. Las funciones base radiales han sido utilizadas para interpolación [52, 66], estimación de densidad de probabilidad [62, 22, 72] y aproximación de funciones multivariables [64]. El modelo presentado a continuación se conoce como redes de funciones base radiales (RFBR).

El término función base radial se puede explicar dividiéndolo en dos partes. Por función radial debe entenderse una función

$$g : \mathbb{R}^d \to \mathbb{R} : (x_1, x_2, \ldots, x_d) \mapsto \phi(\|x_1, x_2, \ldots, x_d\|_2) \tag{3.1}$$

para alguna función $\phi : \mathbb{R} \to \mathbb{R}$. Es decir, el valor de la función g en un punto $\vec{x} = (x_1, x_2, \ldots, x_d)$ sólo depende de la distancia Euclidiana de $\vec{x}$ al origen:

$$\|\vec{x}\|_2 = \sqrt{\sum_{i=1}^{d} x_i^2} \tag{3.2}$$

Lo anterior explica el término *radial*.

Ahora, para explicar la parte de función base, supongamos que se tienen ciertos puntos definidos (llamados centros) $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$. Ahora consideremos la siguiente combinación lineal de la función g centrada en los puntos $\vec{x}_i$:

$$f : \mathbb{R}^d \to \mathbb{R} : \vec{x} \mapsto \sum_{i=1}^{n} \alpha_i g(\vec{x} - \vec{x}_i) = \sum_{i=1}^{n} \alpha_i \phi(\|\vec{x} - \vec{x}_i\|) \tag{3.3}$$

donde $\|\vec{x} - \vec{x}_i\|$ es la distancia Euclidiana entre los puntos $\vec{x}$ y $\vec{x}_i$. De esta forma se tiene una función compuesta f que se encuentra extendida, en un espacio dimensional finito, por las funciones base:

$$g_i : \vec{x} \mapsto g(\|\vec{x} - \vec{x}_i\|) \tag{3.4}$$

La característica más importante que distingue a las RFBR de modelos que utilizan las funciones base radiales, es su naturaleza adaptativa, que generalmente les permite utilizar un número relativamente pequeño de unidades localmente sintonizadas (funciones base radiales). Las RFBR fueron propuestas de forma independiente por Broomhead y Lowe [5], Lee y Kil [75], Niranjan y Fallside en 1988 [57] así como Moody y Darken en 1989 [54, 53]. Esquemas similares fueron propuestos por Hanson y Burr [36], Lapedes y Farver en 1987 [45], Casdagli en 1989 [7], Poggio y Girosi en 1990 [63], entre otros. En seguida se describe la arquitectura de las RFBR y su algoritmo asociado de entrenamiento.

Las RFBR tienen una estructura de alimentación hacia adelante que consiste en una única capa oculta de J unidades localmente sintonizadas que se encuentran completamente interconectadas a una capa de salida de L unidades lineales, como se muestra en la figura 3.1.

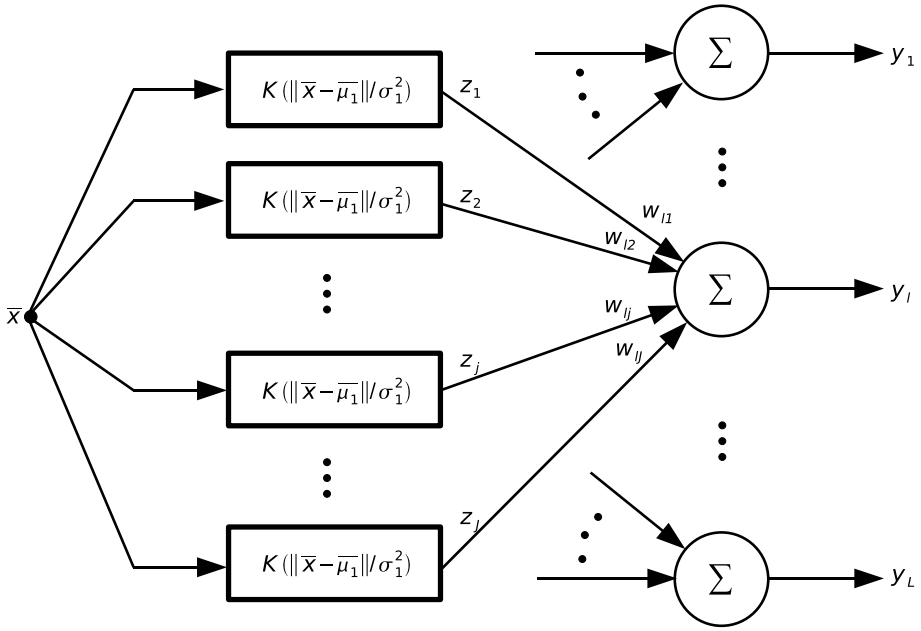

Figura 3.1: Red de funciones base radiales.

Todas las células ocultas reciben simultáneamente los valores reales del vector n-dimensional de entrada $\vec{x}$. Nótese la ausencia de pesos (pesos = 1 para todas las entradas) de la capa oculta a diferencia de las redes neuronales artificiales. Esto se debe a que las salidas de las neuronas en la capa oculta no se calculan usando el mecanismo de activación de sumas de pesos y sigmoidal, sino que, cada salida oculta z_j se obtiene calculando la cercanía de la entrada $\vec{x}$ a un parámetro vectorial n-dimensional llamado $\vec{\mu}_j$ asociado con la j-ésima célula oculta. En dicha célula, la respuesta característica está dada por la ecuación mostrada a continuación:

$$z_j(\vec{x}) = K\left(\frac{\|\vec{x}-\vec{\mu}_j\|}{\sigma_j^2}\right) \tag{3.5}$$

donde K es una función estrictamente positiva simétrica radialmente con un máximo en su centro $\vec{\mu}_j$ y que decrece rápidamente a cero al alejarse del centro. El parámetro σ_j es el ancho del campo receptivo en el espacio de entrada para la célula j. Esto implica que z_j tenga un valor apreciable únicamente cuando la distancia $\|\vec{x}-\vec{\mu}_j\|$ sea menor al ancho σ_j. Dado un vector de entrada $\vec{x}$, la salida de la RFBR será el vector L-dimensional $\vec{y}$, cuyo l-ésimo componente está dado por:

$$y_l(\vec{x}) = \sum_{j=1}^{J} w_{lj} z_j(\vec{x}) \tag{3.6}$$

El uso de las RFBR es adecuado para aproximación de funciones continuas $f : \mathbb{R}^n \rightarrow \mathbb{R}^L$, donde n es suficientemente pequeña. De acuerdo a las ecuaciones 3.5 y 3.6, las RFBR pueden verse como una aproximación de la función deseada $f(\vec{x})$ por superposición de funciones base no ortogonales con forma de campana. El grado de precisión puede controlarse por tres parámetros: el número de funciones base utilizadas, su localización y su anchura. Puede mostrarse que las RFBR son aproximadores universales [65, 37, 3, 61].

En las RFBR se utiliza comúnmente una función base Gaussiana (ecuación (3.7)) para las células ocultas.

$$z_j(\vec{x}) = \exp\left(-\frac{\|\vec{x} - \vec{\mu}_j\|^2}{2\sigma_j^2}\right) \tag{3.7}$$

donde σ_j y $\vec{\mu}_j$ son la desviación estándar y la media de la j-ésima unidad receptiva, respectivamente.

En la figura 3.2 se muestra una aproximación de dicha función base.

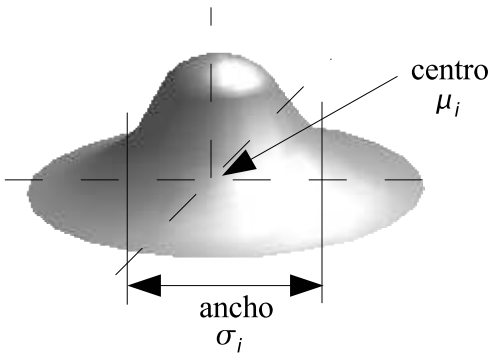

Figura 3.2: Función base radial Gaussiana y parámetros.

Otras posibles funciones base radiales son las mostradas en la tabla 3.2.

Lineal	r
Monomial	$\|r\|^{2m+1}$
Tiras de láminas delgadas	$\|r\|^{2m} \log \|r\|$
Multicuadrática	$\sqrt{1 + (\varepsilon r)^2}$
Cuadrática inversa	$\frac{1}{1 + (\varepsilon r)^2}$
Multicuadrática inversa	$\frac{1}{\sqrt{1 + (\varepsilon r)^2}}$
Gaussiana	$\exp^{-(\varepsilon r)^2}$

$r = (\|\vec{x} - \vec{\mu}_j\|)$

Tabla 3.2: Funciones base radiales comúnmente utilizadas.

3.1.1. Entrenamiento de las RFBR

Considérese un conjunto de entrenamiento de pares etiquetados como $x^i, d^i, \quad i = 1, \ldots, m$ que representan asociaciones de un mapeo determinado o muestras de una función multivariable continua. También considérese la función a minimizarse E que representa el criterio de error de suma de cuadrados, como el representado en la ecuación (3.8), el cual será minimizado para el conjunto de entrenamiento ofrecido.

$$E = \frac{1}{2}\sum_{i=1}^{m}(d^i - y^i)^2 \tag{3.8}$$

En otras palabras, se desea desarrollar un método de entrenamiento que minimice E actualizando de forma adaptativa los parámetros variables de la RFBR. Estos parámetros son los centros del campo receptivo (las medianas $\vec{\mu}_j$ de las unidades Gaussianas en la capa oculta), los anchos del campo receptivo (las desviaciones estándar σ_j) y los pesos de la capa de salida (w_{lj}).

Debido a la naturaleza diferenciable de las características de transferencia de las RFBR, se puede utilizar el método del gradiente descendente totalmente supervisado sobre E [54, 64]. En particular, los valores de $\vec{\mu}_j$, σ_j y w_{lj} se actualizan con las fórmulas 3.9, 3.10 y 3.11, respectivamente.

$$\Delta\vec{\mu}_j = -\rho_\mu \nabla_{\vec{\mu}_j} E \tag{3.9}$$

$$\Delta\sigma_j = -\rho_\sigma \frac{\partial E}{\partial \sigma_j} \tag{3.10}$$

$$\Delta w_{lj} = -\rho_w \frac{\partial E}{\partial w_{lj}} \tag{3.11}$$

donde ρ_μ, ρ_σ y ρ_w son constantes positivas pequeñas.

La desventaja de este método de entrenamiento es que el tiempo de entrenamiento es comparable con el entrenamiento de redes neuronales generales [79]. De utilizar el método de aprendizaje supervisado, no se garantiza el uso de la ventaja computacional de localidad. Una forma de rectificar este problema es utilizando un método que mantenga pequeños los valores de σ_j, además del uso de aprendizaje basado en gradiente descendente para los centros de las funciones base.

Es posible utilizar una estrategia de entrenamiento que separe el aprendizaje de la capa oculta del de la capa de salida en las RFBR debido a la naturaleza de campo receptivo local de las células ocultas. Esta estrategia ha mostrado ser efectiva en términos de velocidad de entrenamiento. No obstante, esta ventaja es contrarrestada por una reducida capacidad de generalización a menos que se posea un número elevado de funciones base. A continuación se describen métodos eficientes para localizar los centros y obtener los anchos de los campos receptivos. En lo que respecta a los pesos de la capa de salida, éstos pueden ser calculados mediante la ecuación (3.11), una vez que se han sintetizado las celdas ocultas. Este cálculo puede verse como la acción de encontrar los coeficientes de normalización apropiados de las funciones base. Es decir,

los pesos w_{lj} determinan la contribución de la j-ésima función base hacia la l-ésima salida de la RFBR.

Una forma de encontrar los centros y anchos apropiados de los campos receptivos sin propagar el error de salida hacia atrás en la red es poblar densamente el espacio de entrada con campos receptivos, como en la figura 3.3. Un método consiste en colocar los centros de acuerdo a una malla definida en el espacio de entrada [5]. Asumiendo una malla uniforme, con k divisiones a lo largo de cada dimensión en el espacio de entrada n-dimensional, esta malla requeriría k^n funciones base para cubrir el espacio de entrada, por lo que se volvería poco práctico para espacios de alta dimensionalidad. Una estrategia alternativa sería colocar k centros en un conjunto de k muestras de entrenamiento seleccionadas de manera aleatoria. Este método requiere de un gran número de campos receptivos para representar de forma adecuada la distribución de los vectores de entrada en un espacio altamente multidimensional, a menos que se tenga conocimiento previo sobre la localización de regiones del espacio de entrada que sean valiosas para la interpolación.

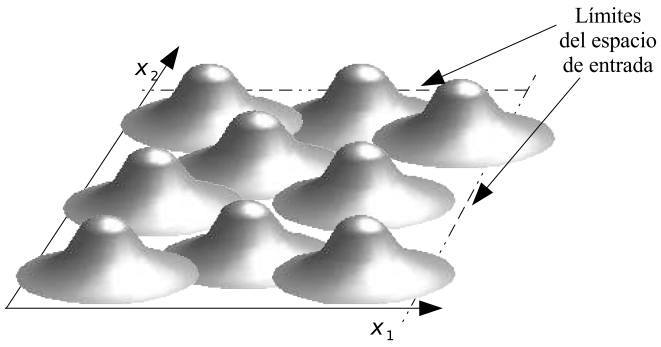

Figura 3.3: Campo de entrada saturado de campos receptivos.

Moody y Darken en 1989 [54], emplearon un aprendizaje no supervisado de los centros de campo receptivos $\vec{\mu}_j$ en el cual utilizaron un número relativamente pequeño de funciones base. Los centros adaptativos se entrenan para representar sólo las partes del espacio de entrada que se encuentra ricamente representado por grupos de datos. La estrategia adaptativa también ayuda a reducir el error de muestreo debido a que permite determinar los centros $\vec{\mu}$ de un gran número de muestras de entrenamiento. Se utiliza el algoritmo de agrupamiento k-medios [46, 1] para localizar el conjunto de k centros de funciones base que representarán un mínimo local del error E entre los vectores de entrenamiento y el centro $\vec{\mu}_j$ más próximo de los k centros. En el algoritmo de k-medios clásico, las k funciones base radiales que son asignadas inicialmente como centros $\vec{\mu}_j, \quad j = 1, 2, \ldots, k$ son tomadas de k vectores de entrenamiento de forma aleatoria. Los vectores sobrantes son asignados a la clase j del centro más cercano $\vec{\mu}_j$. Después, los centros son recalculados como el promedio de los vectores de entrenamiento de la misma clase. Se continúa con estos dos pasos hasta que todos los centros dejan de cambiar. Una versión incremental de este proceso se puede implementar sin

necesidad de almacenar los vectores pasados de entrenamiento ni la información de identificación de grupo de cada vector. En este proceso, en cada paso, se selecciona un vector de entrenamiento $\vec{x}$ de forma aleatoria y el centro $\vec{\mu}_j$ del campo receptivo más cercano se actualiza de acuerdo a:

$$\Delta\vec{\mu}_j = \rho(\vec{x} - \vec{\mu}_j) \tag{3.12}$$

donde ρ es una constante pequeña positiva. Generalmente, no existe ningún método formal para especificar el número de k células ocultas en una RFBR. Normalmente se utiliza la validación cruzada para decidir k.

Una vez que los centros del campo receptivo se han encontrado, debe buscarse su ancho σ_j para obtener una interpolación suave. En teoría, las RFBR con el mismo valor de σ_j en cada célula oculta, tienen la capacidad de aproximación universal [61]. Por lo tanto, se puede tomar un solo valor global de σ para todas las células ocultas. Se debe utilizar un valor relativamente pequeño (positivo) para este parámetro de ancho para preservar las características de respuesta local de las células ocultas. El valor σ debe encontrarse mediante validación cruzada para un conjunto particular de entrenamiento. Resultados empíricos de Moody y Darken en 1989 [54] sugieren que una buena estimación del parámetro global de ancho es el ancho promedio $\sigma = \langle\|\vec{\mu}_i - \vec{\mu}_j\|\rangle$, que representa un promedio global de todas las distancias Euclidianas entre el centro de cada unidad i y su vecino más cercano j. Otra heurística basada en cálculos locales consiste en obtener el valor de $\sigma_j = \alpha\|\vec{\mu}_i - \vec{\mu}_j\|$, donde $\vec{\mu}_i$ es el centro del vecino más cercano a la unidad j (generalmente $1.0 \leq \alpha \leq 1.5$).

Como se mencionó anteriormente, los pesos para la capa de salida w_{lj} pueden ser calculados adaptativamente utilizando la regla delta

$$\Delta w_{lj} = w_{lj}^n - w_{lj}^a = -\rho_w \frac{\partial E}{\partial w_{lj}} = \rho(d_l - y_l) f'_0(red_l) z_j \tag{3.13}$$

donde $red_l = \sum_{j=0}^{J} w_{lj} z_j$ es la suma ponderada para la unidad de salida l, $f_0(red) = \lambda\ red$ se establece como activación lineal en la salida. f'_0 es la derivada de f_0 con respecto a red y por último w^n y w^a representan los valores de los pesos nuevos y actuales, respectivamente.

Como se puede apreciar, el término $f'(red_l)$ puede omitirse por tratarse del caso de unidades lineales. La ecuación (3.13) conduce a la minimización del error E modificando los pesos de la capa de salida, para un valor suficientemente pequeño de ρ. De forma alternativa, para el caso de unidades de salida lineales, se puede formular el problema de calcular los pesos como un conjunto de ecuaciones lineales simultáneas y emplear el método de inversión para obtener la solución de minimizar E. Sin perder generalidad, considérese una RFBR de una salida y sea $\vec{w} = [w_1, w_2, \cdots, w_j]$ el vector de pesos de la unidad de salida. Para problemas de interpolación estricta, se desea que la función de interpolación encontrada se encuentre restringida a mapear de forma precisa los puntos de prueba $\vec{x}^i$ en su objetivo asociado d^i para $i = 1, 2, \ldots, m$. Se sabe que un polinomio de orden finito $r = m - 1$ es capaz de realizar una interpolación estricta en m muestras $\vec{x}^i, d^i$, asumiendo distintos vectores $\vec{x}^i \in \mathbb{R}^n$. Un

resultado similar se tiene en las RFBR; por lo tanto, existe una clase de funciones base radiales que garantizan que una RFBR con m funciones sea capaz de interpolar de forma estricta m puntos de prueba. Por lo anterior, en este caso puede omitirse la búsqueda de centros $\vec{\mu}_j$ al igualarlos a los puntos que se tienen de prueba $\vec{\mu}_j = \vec{x}^j$ para $j = 1, 2, \ldots, m$. Con esto se forma una matriz $\mathbf{Z}$ como la representada en la ecuación (3.14), la cual es llamada *matriz de interpolación.*

$$\mathbf{Z}_{ji} = K\left(\frac{\|\vec{x}^i - \vec{x}^j\|}{\sigma_j^2}\right) \qquad i = 1, 2, \ldots, m; \quad j = 1, 2, \ldots, m \tag{3.14}$$

Debe notarse que el parámetro σ_j aún debe ser encontrado, ya que éste afecta la calidad de la interpolación de la red [38, pág. 292].

De lo anterior, se puede asegurar una solución $\vec{w}^*$ si $\mathbf{Z}$ es una matriz no singular. Los valores de $\vec{w}^*$ pueden obtenerse con la ecuación:

$$\vec{w}^* = (\mathbf{Z}^T)^{-1}\vec{d} \tag{3.15}$$

donde $\vec{d} = [d^1, d^2, \ldots, d^m]^T$.

Nótese que, al utilizar todos los puntos de prueba, la matriz $\mathbf{Z}$ es simétrica, por lo tanto $\mathbf{Z} = \mathbf{Z}^T$. Aunque en teoría, la ecuación (3.15) asegura siempre una solución al problema estricto de interpolación, en la práctica, el cálculo directo de $\mathbf{Z}^{-1}$ puede volverse mal condicionado cuando $\mathbf{Z}$ se encuentra cercana a ser singular.

3.2. Kriging

Kriging es un método de predicción espacial basado en la minimización del error cuadrático medio. Generalmente, el método kriging depende de las propiedades de segundo orden de la función aleatoria $Z(\cdot)$. Matheron [47] nombró el método de prediccion lineal espacial óptima como kriging en honor a D. G. Krige, un ingeniero sudafricano quien, en los años 50 desarrolló un método empírico para determinar la distribución de los yacimientos de oro basado en muestreos de los mismos en un territorio [44]. Sin embargo, la formulación de la predicción lineal espacial óptima no se obtuvo del trabajo de Krige [48]. En los trabajos de Wold en 1938 [81], Kolmogorov en 1941 [43] y Wiener en 1949 [80] se encuentran ecuaciones de predicción lineal óptima que reflejan la noción de dar más peso a las observaciones cercanas al punto de predicción.

Al mismo tiempo del desarrollo de la geoestadística en ingeniería de minas bajo las investigaciones de G. Matheron en Francia, las mismas ideas fueron desarrolladas en meteorología en la entonces Unión Soviética por L. S. Gandin [34]. La contribución original de estos autores (de manera simultánea) fue establecer la predicción lineal óptima (en términos de variogramas) en un conjunto espacial. Gandin nombró su estrategia *análisis objetivo* y utilizó el término *interpolación óptima* en lugar de kriging.

Para una comprensión global del método kriging, es necesario conocer el término variograma, por lo cual se procederá a explicarlo a continuación.

3.2.1. Variograma

Suponiendo

$$\text{var}(Z(\vec{s}_1) - Z(\vec{s}_2)) = 2\gamma(\vec{s}_1 - \vec{s}_2), \quad \forall \vec{s}_1, \vec{s}_2 \in D \subset \mathbb{R}^d \tag{3.16}$$

La cantidad $2\gamma(\cdot)$, que es función únicamente del incremento $\vec{s}_1 - \vec{s}_2$, fue llamada *variograma* y $\gamma(\cdot)$ *semivariograma* por Matheron en 1962 [49]. En la explicación del método kriging, la función 2γ (suponiendo que existe), será tratada como un parámetro del proceso aleatorio $Z(\cdot)$.

Los variogramas se pueden dividir en dos clases, los variogramas experimentales, que denotaremos como $\hat{\gamma}(\vec{h})$ y los variogramas modelo, que se describirán como $\gamma(\vec{h}; \vec{\theta})$. Los primeros se pueden generar mediante los datos $\vec{s}_1, \ldots, \vec{s}_n$ y su evaluación correspondiente en el proceso $Z(\vec{s}_i), \quad i = 1, \ldots, n$. Mientras que los variogramas modelo son familias de funciones que cumplen con ciertas propiedades [15, pág. 60].

El variograma experimental puede estimarse, basándose en el método de momentos [49, 50], con la siguiente expresión

$$2\hat{\gamma}(\vec{h}) \equiv \frac{1}{|N(\vec{h})|} \sum_{N(\vec{h})} \left(Z(\vec{s}_i - Z(\vec{s}_j))\right), \qquad \vec{h} \in \mathbb{R}^d \tag{3.17}$$

donde

$$N(\vec{h}) \equiv \{(\vec{s}_i, \vec{s}_j) : \vec{s}_i - \vec{s}_j = \vec{h}; i, j = 1, \ldots, n\} \tag{3.18}$$

y $|N(\vec{h})|$ es el número de pares distintos en $N(\vec{h})$. Nótese que $N(\vec{h}) \neq N(-\vec{h})$ aunque $2\hat{\gamma}(\vec{h}) = 2\hat{\gamma}(-\vec{h})$.

En este trabajo se utilizaron modelos de variogramas isotrópicos, es decir, que únicamente dependen de $\|\vec{s}_1 - \vec{s}_2\|$. Algunos de estos modelos son mostrados a continuación, limitándonos a aquellos que son aplicables en $\mathbb{R}^d$, para $d \geq 1$.

- *Modelo lineal*:

$$\gamma(\vec{h}; \vec{\theta}) = \begin{cases} 0 & \vec{h} = \vec{0} \\ c_0 + b_l \|\vec{h}\| & \vec{h} \neq \vec{0} \end{cases} \tag{3.19}$$

  $\vec{\theta} = (c_0, b_l)^T$, donde $c_0 \geq 0$ y $b_l \geq 0$.

- *Modelo exponencial (Gaussiano)*:

$$\gamma(\vec{h}; \vec{\theta}) = \begin{cases} 0 & \vec{h} = \vec{0} \\ c_0 + c_e\{1 - \exp(-\|\vec{h}\|/a_e)\} & \vec{h} \neq \vec{0} \end{cases} \tag{3.20}$$

  $\vec{\theta} = (c_0, c_e, a_e)^T$, donde $c_0 \geq 0$, $c_e \geq 0$ y $a_e \geq 0$.

- *Modelo racional cuadrático*:

$$\gamma(\vec{h}; \vec{\theta}) = \begin{cases} 0 & \vec{h} = \vec{0} \\ c_0 + c_r \|\vec{h}\|^2/(1 + \|\vec{h}\|^2/a_r) & \vec{h} \neq \vec{0} \end{cases} \tag{3.21}$$

  $\vec{\theta} = (c_0, c_r, a_r)^T$, donde $c_0 \geq 0$, $c_r \geq 0$ y $a_r \geq 0$.

- *Modelo potencia*:

$$\gamma(\vec{h};\vec{\theta}) = \begin{cases} 0 & \vec{h} = \vec{0} \\ c_0 + b_p\|\vec{h}\|^\lambda & \vec{h} \neq \vec{0} \end{cases} \tag{3.22}$$

  $\vec{\theta} = (c_0, b_p, \lambda)^T$, donde $c_0 \geq 0$, $b_p \geq 0$ y $0 \leq \lambda < 2$.

3.2.2. Kriging ordinario

Supongamos que tenemos una función aleatoria (o proceso aleatorio) $Z(\vec{s}) : \vec{s} \in D \subset \mathbb{R}^d$, para la cual se muestrearon n datos $Z(\vec{s}_1), \ldots, Z(\vec{s}_n)$. Estos datos son usados para realizar una inferencia en el proceso, es decir, para encontrar una función $g(Z(\vec{s}) : s \in D)$, o de manera abreviada $g(Z(\cdot))$, que es una predicción de la función $Z(\cdot)$.

Algunas veces no se está interesado en la función $Z(\cdot)$, sino en la versión "sin ruido" de dicha función. Supongamos que la función $Z(\cdot)$ se puede descomponer en:

$$Z(\vec{s}) = S(\vec{s}) + \epsilon(\vec{s}), \qquad \vec{s} \in D \tag{3.23}$$

donde $\epsilon(\cdot)$ es un error en el proceso de medición. En este caso, el interés radica en predecir una función $g(S(\cdot))$ de la función aleatoria $S(\cdot)$ sin ruido.

La función de este método es realizar inferencias en valores no observados del proceso aleatorio $Z(\cdot)$ o $S(\cdot)$ a partir de los datos en:

$$\mathbf{Z} \equiv (Z(\vec{s}_1), \ldots, Z(\vec{s}_n))^T$$

observados en lugares espaciales conocidos

$$\vec{s}_1, \ldots, \vec{s}_n$$

Denotemos el predictor genérico de $g(Z(\cdot))$ o $g(S(\cdot))$ por

$$p(\mathbf{Z}; g) \tag{3.24}$$

La elección de un buen predictor dependerá de la geometría y localización de la región de espacio donde se desea la predicción y si se requiere predecir el proceso Z o el proceso S.

El método kriging ordinario realiza una predicción espacial bajo las siguientes dos suposiciones:

- *De modelo*

$$Z(s) = \mu + \delta(\vec{s}), \qquad \vec{s} \in D, \quad \mu \in \mathbb{R} \tag{3.25}$$

  con μ desconocida.

- *De predictor*

$$p(\mathbf{Z}; B) = \sum_{i=1}^{n} \lambda_i Z(\vec{s}_i), \qquad \sum_{i=1}^{n} \lambda_i = 1 \tag{3.26}$$

La condición de que el predictor lineal sume 1 garantiza

$$E(p(\mathbf{Z}; B)) = \mu = E(Z(B)), \forall \mu \in \mathbb{R} \tag{3.27}$$
$$\tag{3.28}$$

donde

$$Z(B) \equiv \frac{\int_B Z(\vec{u}) d\vec{u}}{|B|} \tag{3.29}$$

y B es el promedio del proceso sobre un bloque B cuya localización y geometría son conocidos. El volumen d-dimensional de B es $|B|$.

Existe una versión de kriging llamada kriging simple, donde μ en la ecuación (3.25) es conocida y los coeficientes no están restringidos a tener que sumar 1.

Si tenemos que

$$g(Z(\cdot)) = Z(B) \equiv \begin{cases} \dfrac{\int_B Z(\vec{u}) d\vec{u}}{|B|} & |B| > 0, \\ \text{prom}\{Z(\vec{u}) : \vec{u} \in B\} & |B| = 0, \end{cases} \tag{3.30}$$

entonces el óptimo $p(\cdot; B)$ minimizará el error cuadrático medio de predicción

$$\sigma_e^2 \equiv E(Z(B) - p(\mathbf{Z}; B)) \tag{3.31}$$

sobre la clase de predictores lineales $\sum_{i=1}^n \lambda_i Z(\vec{s}_i)$ que satisfaga $\sum_{i=1}^n \lambda_i = 1$.

Predicción espacial óptima del proceso Z

Para el método kriging ordinario, la minimización de la ecuación (3.31) se lleva a cabo sobre $(\lambda_1, \ldots, \lambda_n)$, sujeto a $\sum_{i=1}^n \lambda_i = 1$, donde el modelo (3.25) cumple con el variograma

$$2\gamma(\vec{h}) = \text{var}(Z(\vec{s} + \vec{h}) - Z(\vec{s})), \qquad \vec{h} \in \mathbb{R}^d \tag{3.32}$$

La función $\gamma(\vec{h})$ es un variograma modelo como los mostrados en la sección 3.2.1, donde los valores de θ son encontrados al aproximar la función a la gráfica obtenida utilizando directamente los puntos muestreados en la ecuación (3.16).

Supongamos, por el momento, que $B = \{\vec{s}_0\}$ en la ecuación (3.30). Entonces, el proceso de minimizar

$$E\left(Z(\vec{s}_0) - \sum_{i=1}^n \lambda_i Z(\vec{s}_i)\right)^2 - 2m\left(\sum_{i=1}^n \lambda_i - 1\right) \tag{3.33}$$

con respecto a $\lambda_1, \ldots, \lambda_n$ y m (el parámetro m es un multiplicador de Lagrange) asegura que $\sum_{i=1}^n \lambda_i = 1$.

Ahora, la condición $\sum_{i=1}^{n} \lambda_i = 1$ implica que

$$\begin{aligned}\left(Z(\vec{s}_0) - \sum_{i=1}^{n} \lambda_i Z(\vec{s}_i)\right)^2 &= -\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i\lambda_j \left(Z(\vec{s}_i) - Z(\vec{s}_j)\right)^2/2 \\ &\quad +2\sum_{i=1}^{n} \lambda_i \left(Z(\vec{s}_i) - Z(\vec{s}_j)\right)^2/2 \end{aligned} \tag{3.34}$$

de forma que bajo el modelo 3.25, la ecuación (3.33) se transforma a

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i\lambda_j\gamma(\vec{s}_i - \vec{s}_j) + \sum_{i=1}^{n} \lambda_i\gamma(\vec{s}_0 - \vec{s}_i) - 2m\left(\sum_{i=1}^{n} \lambda_i - 1\right) \tag{3.35}$$

Después de derivar la ecuación (3.35) con respecto a $\lambda_1, \ldots, \lambda_n$ y m e igualando el resultado a cero, se puede ver que los parámetros óptimos satisfacen para:

$$-\sum_{i=1}^{n} \lambda_i\gamma(\vec{s}_i - \vec{s}_j) + \gamma(\vec{s}_0 - \vec{s}_i) - m = 0, \qquad i = 1, \ldots, n, \tag{3.36}$$

$$\sum_{i=1}^{n} \lambda_i = 1 \tag{3.37}$$

Por lo que, los valores óptimos de $\lambda_1, \ldots, \lambda_n$ pueden obtenerse con la ecuacion:

$$\vec{\lambda}_0 = \mathbf{\Gamma}_0^{-1}\vec{\gamma}_0 \tag{3.38}$$

donde

$$\vec{\lambda}_0 \equiv (\lambda_1, \ldots, \lambda_n, m)^T \tag{3.39}$$

$$\vec{\gamma}_0 \equiv (\gamma(\vec{s}_0 - \vec{s}_1), \ldots, \gamma(\vec{s}_0 - \vec{s}_1), 1)^T \tag{3.40}$$

$$\mathbf{\Gamma}_0 \equiv \begin{cases} \gamma(\vec{s}_i - \vec{s}_j) & i = 1, \ldots, n;\ j = 1, \ldots, n \\ 1 & i = n+1;\ j = 1, \ldots, n \\ 0 & i = n+1;\ j = n+1 \end{cases} \tag{3.41}$$

y $\mathbf{\Gamma}_0$ es una matriz simétrica $(n+1)\times(n+1)$. Se puede apreciar que la ecuación (3.38) se mantiene si $\gamma(\vec{h})$ se reemplaza por $\gamma(\vec{h})+c$, para cualquier $c \in \mathbb{R}$. Dicha sustitución es necesaria algunas veces para obtener una solución numéricamente estable a $\mathbf{\Gamma}_0\vec{\lambda}_0 = \vec{\gamma}_0$.

De la ecuación (3.38), los coeficientes $\vec{\lambda} \equiv (\lambda_1, \ldots, \lambda_n)^T$ están dados por

$$\vec{\lambda}^T = \left(\vec{\gamma} + \vec{1}\frac{(1 - \vec{1}^T\mathbf{\Gamma}^{-1}\vec{\gamma})}{\vec{1}^T\mathbf{\Gamma}^{-1}\vec{1}}\right)^T \mathbf{\Gamma}^{-1} \tag{3.42}$$

$$m = -\frac{1 - \vec{1}^T\mathbf{\Gamma}^{-1}\vec{\gamma}}{\vec{1}^T\mathbf{\Gamma}^{-1}\vec{1}} \tag{3.43}$$

donde $\vec{\gamma} \equiv (\gamma(\vec{s}_0 - \vec{s}_1), \ldots, \gamma(\vec{s}_0 - \vec{s}_n))^T$ y $\mathbf{\Gamma}$ es la matriz de $n \times n$ cuyo elemento (i, j) es $\gamma(\vec{s}_i - \vec{s}_j)$. Con estos datos se puede obtener el predictor óptimo $\hat{p}(\mathbf{Z}; \vec{s}_0)$ de la ecuación (3.26).

El error cuadrático medio de predicción minimizado (ver ecuación (3.31)) también es llamado *varianza kriging* y puede expresarse por

$$\sigma_k^2(\vec{s}_0) = \vec{\lambda}_0^T \vec{\gamma}_0 = \sum_{i=1}^{n} \lambda_i \gamma(\vec{s}_0 - \vec{s}_i) + m \tag{3.44}$$

$$= \vec{\gamma}^T \mathbf{\Gamma}^{-1} \vec{\gamma} - \frac{\vec{1}^T \mathbf{\Gamma}^{-1} \vec{\gamma} - 1}{\vec{1}^T \mathbf{\Gamma}^{-1} \vec{1}} \tag{3.45}$$

También se puede expresar como

$$\sigma_k^2(\vec{s}_0) = 2 \sum_{i=1}^{n} \lambda_i \gamma(\vec{s}_0 - \vec{s}_i) - \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j \gamma(\vec{s}_i - \vec{s}_j) \tag{3.46}$$

3.2.3. Ajuste de parámetros del método kriging

El objetivo de utilizar el método kriging como herramienta de interpolación es encontrar la función $g(Z(\cdot))$, que es una predicción de la función $Z(\cdot)$. Para la tarea anterior es necesario conocer los valores de los parámetros que componen el modelo de predicción kriging. Esto puede llevarse a cabo de distintas formas; sin embargo, a continuación se describirá el proceso utilizado en este trabajo.

1. Cálculo del variograma experimental $2\hat{\gamma}(\vec{h})$ mediante la ecuación (3.17). Se recomienda que existan al menos 30 valores $\vec{h}$ para realizar un buen variograma [15, pág. 70].

2. Ajuste de los parámetros $\vec{\theta}$ de un variograma modelo seleccionado basándose en los datos del variograma experimental, mediante la aproximación del variograma modelo al variograma experimental. Al realizar dicha aproximación, el variograma experimental debe ser "recortado" a la mitad, es decir, que la diferencia máxima a utilizarse será a lo más la mitad de la diferencia máxima posible y además se deben utilizar las diferencias para las cuales $|N(\vec{h}_j)| > 30$ [41, pág. 194].

3. Obtención de pesos del predictor $\vec{\lambda}$. Se obtienen mediante la ecuación (3.42).

4. Evaluación del vector no conocido $\vec{s}_0$ mediante la ecuación (3.26) utilizando los pesos calculados en el paso anterior. Debe notarse que el vector $\vec{\gamma}$ utilizado en la ecuación (3.42) obtiene valores distintos para cada evaluación del vector $\vec{s}_0$.

3.3. Comparación de métodos de aproximación de funciones

En la sección anterior se describieron dos familias de métodos de aproximación de funciones, las RFBR y el método de kriging con sus diferentes variogramas. En esta sección se mostrarán experimentos de aproximación de funciones utilizando algunas de las variantes mencionadas adoptando superficies en tres dimensiones, con el fin de apreciar visualmente su aproximación.

Con la finalidad de apreciar la interpolación obtenida por los métodos previamente mencionados, se utilizó la función:

$$y(\vec{x}) = 2\sin(x_1) + 2\cos(\sqrt{x_2}), \qquad \vec{x} = [x_1, x_2] \tag{3.47}$$

La interpolación de la función se hizo en la superficie $\mathcal{S}$ delimitada por $0 \leq x_1 \leq 10$ y $0 \leq x_1 \leq 10$. Se obtuvieron 30 puntos en dicha región de forma aleatoria. Los puntos resultantes pueden ser observados en la figura 3.4.

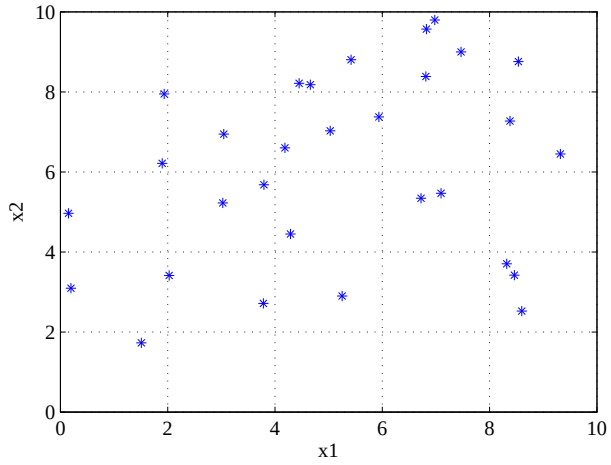

Figura 3.4: Distribución de puntos previamente evaluados

La evaluación de $\mathcal{S}$ en la función 3.47 se muestra en la figura 3.5. Se indican en rojo los 30 puntos previamente evaluados que servirán como base para la interpolación.

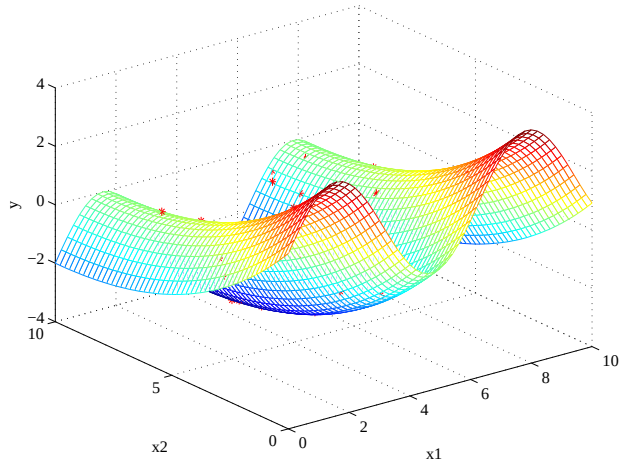

Figura 3.5: Función 3.47 y puntos previamente evaluados.

Utilizando las RFBR como herramienta de interpolación, utiilizamos las funciones base mostradas anteriormente en la tabla 3.2. A continuación se muestra en la figura 3.6 la aproximación de la función de prueba 3.47 utilizando RFBR y una función base lineal. A la izquierda se muestra la interpolación y a la derecha la diferencia entre la función original y la aproximada.

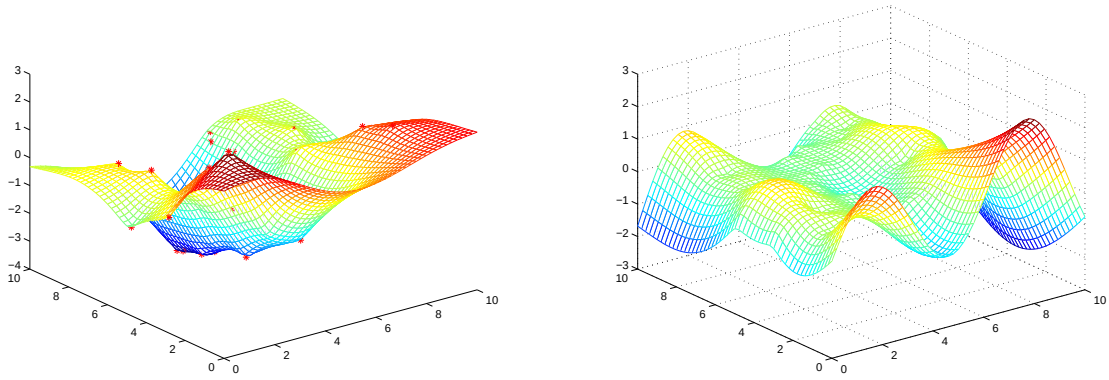

Figura 3.6: Aproximación con una RFBR y una función base lineal.

Siguiendo con la RFBR, se realizó la interpolación de la función prueba, pero esta vez se utilizó una función base monomial con $m = 1$. El resultado se muestra en la figura 3.7. Al utilizar la función base monomial se aprecia una interpolación más suave con respecto a la anterior (lineal) y el error también se reduce.

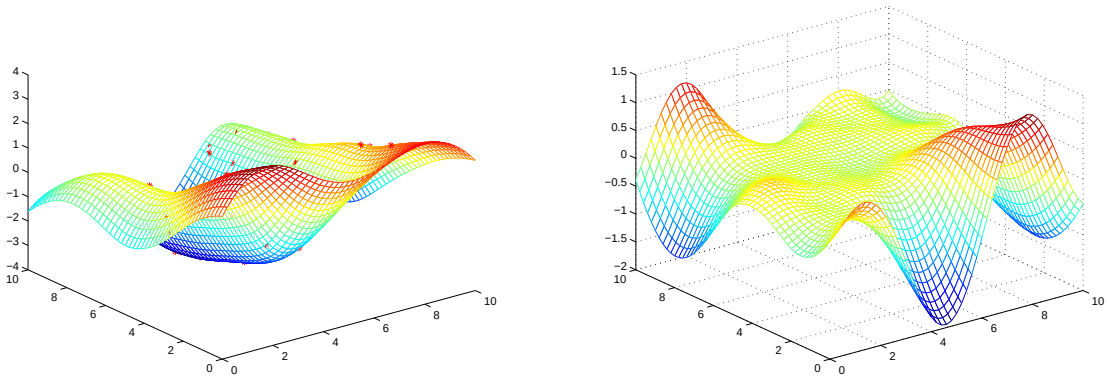

Figura 3.7: Aproximación con una RFBR y una función base monomial con $m = 1$.

En la figura 3.8 se muestra la interpolación realizada con la RFBR y funciones base del tipo de tiras de láminas delgadas con un valor de $m = 1$. Se puede apreciar que la aproximación es más precisa en los lugares cercanos a los puntos de prueba, pero se muestra más errática en lugares con poca información, es decir, con menos puntos circundantes previamente evaluados.

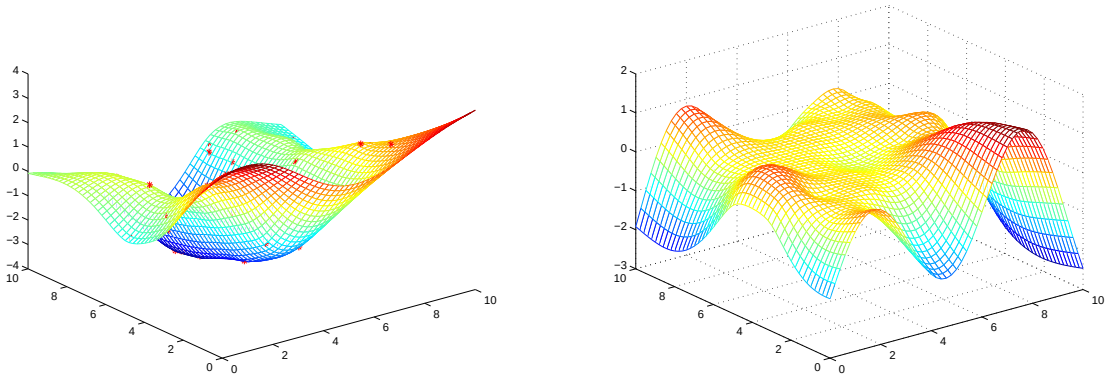

Figura 3.8: Aproximación con una RFBR y una función base de tiras de láminas delgadas con $m = 1$.

A continuación se muestra la aproximación realizada con la función base multi-cuadrática. En la figura 3.9 se utilizó una RFBR con funciones base multi-cuadráticas con un valor $\epsilon = 0.8$.

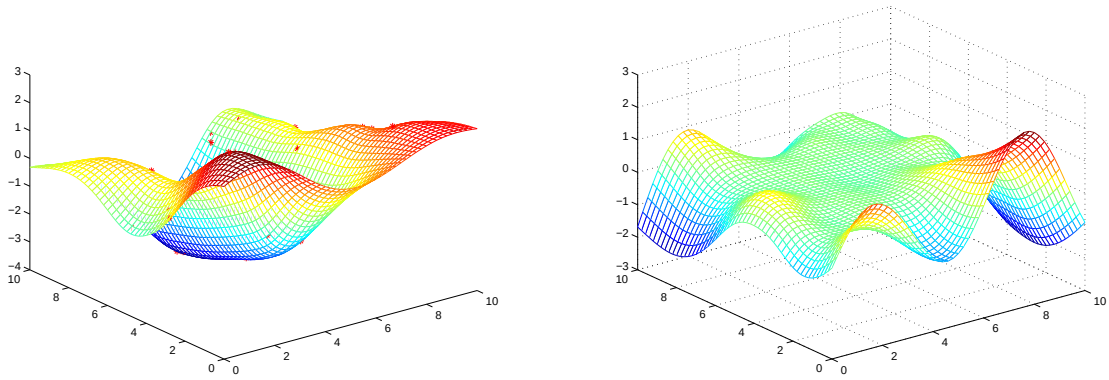

Figura 3.9: Aproximación con una RFBR y una función base multi-cuadrática con $\epsilon = 0.8$.

En la figura 3.10 se muestra el resultado de utilizar la función base cuadrática inversa con un valor de $\epsilon = 0.1$ en una RFBR. En esta aproximación se aprecia una mayor similitud con la función original. También se puede apreciar que la diferencia con la función original se acentúa en las zonas carentes de puntos previamente evaluados.

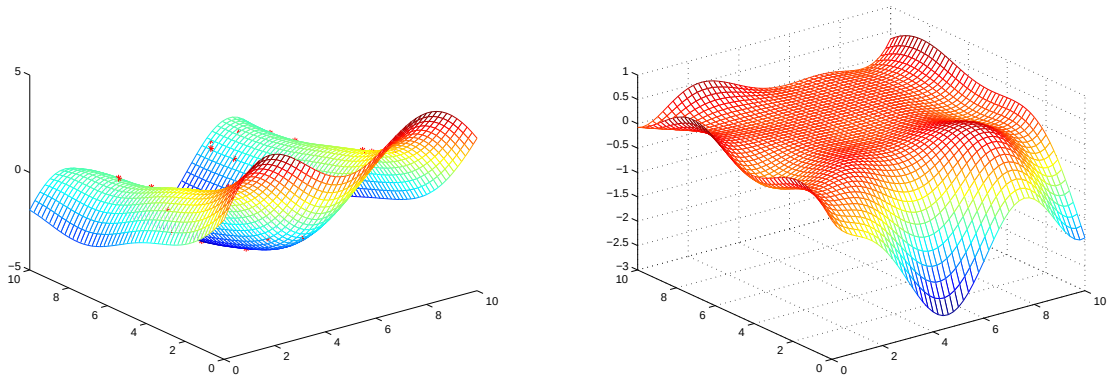

Figura 3.10: Aproximación con una RFBR y una función base cuadrática inversa con $\epsilon = 0.1$.

En la figura 3.11 se muestra el resultado de utilizar la función base multi-cuadrática inversa con un valor de $\epsilon = 0.1$ en una RFBR. Se tiene una gran similitud con la figura 3.10.

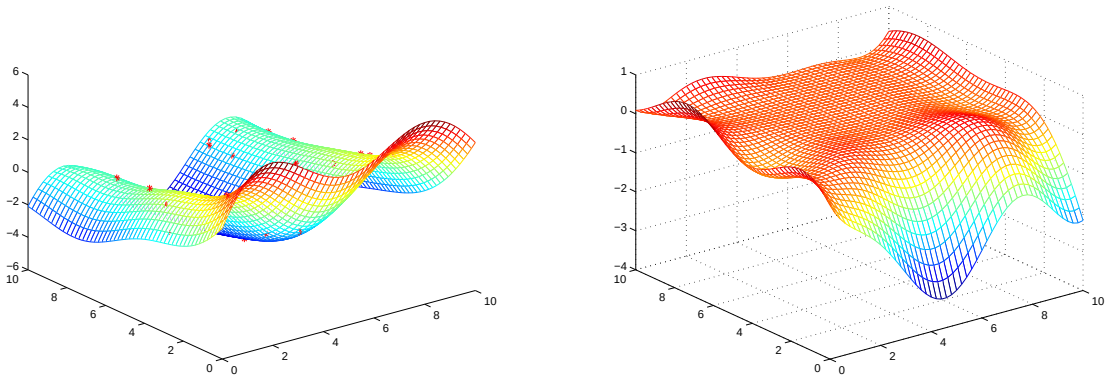

Figura 3.11: Aproximación con una RFBR y una función base multi-cuadrática inversa con $\epsilon = 0.1$.

Y, finalmente dentro de las RFBR, se realizó la interpolación mediante la función base Gaussiana, con un valor de $\epsilon = 0.3$. Los resultados se pueden apreciar en la figura 3.12. Los resultados también son aceptables, ya que difiere únicamente en zonas sin puntos pre-evaluados.

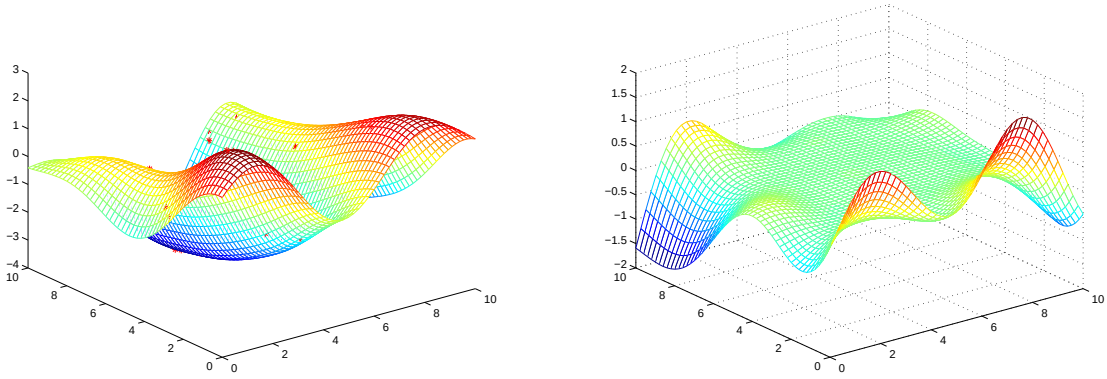

Figura 3.12: Aproximación con una RFBR y una función base Gaussiana con $\epsilon = 0.3$.

Las aproximaciones siguientes fueron realizadas con el método kriging utilizando los variogramas mencionados en la sección 3.2.1, a excepción del variograma modelo potencia, ya que en este caso, se obtienen resultados similares al variograma exponencial.

Con el método kriging, utilizando un variograma lineal, se obtuvo la aproximación mostrada en la figura 3.13.

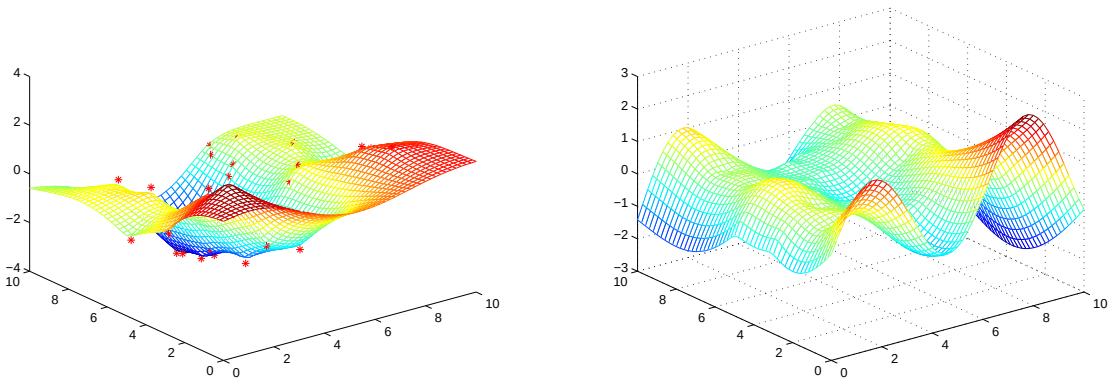

Figura 3.13: Aproximación con kriging y un variograma lineal.

Utilizando el variograma exponencial con el método kriging obtuvo la interpolación mostrada en la figura 3.14.

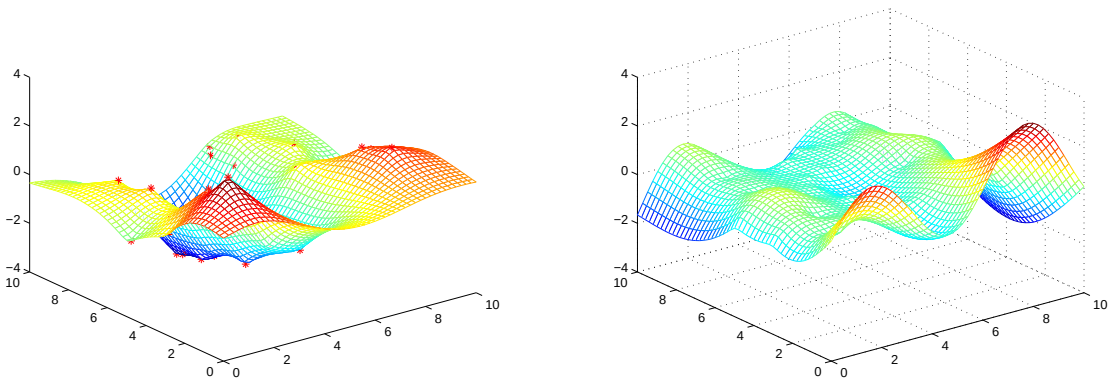

Figura 3.14: Aproximación con Kriging y un variograma exponencial.

Ahora, usando un variograma racional cuadrático se obtuvo la aproximación mostrada en la figura 3.15.

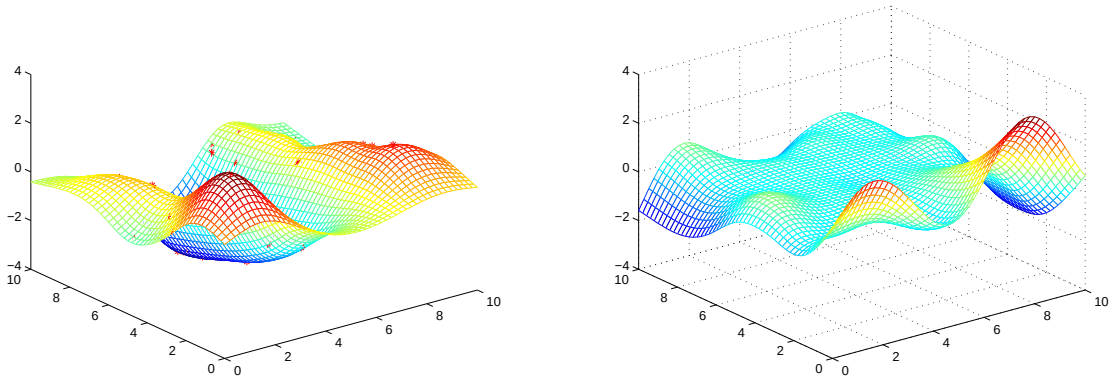

Figura 3.15: Aproximación con kriging y un variograma racional cuadrático.

Debido a que la apreciación visual no es suficiente para determinar qué método y variante son los más adecuados, se realizó una comparación numérica entre el valor real de la función $y(\vec{x})$ y su aproximación en el área $\mathcal{S}$. La comparación se realizó utilizando la fórmula 3.48, la cual nos dará un valor proporcional al error en la interpolación, utilizando evaluaciones distribuidas de forma homogénea en $\mathcal{S}$.

$$pv(\vec{x}) = |\sum_{i=1}^{n} y(\vec{x}_i) - \hat{y}(\vec{x}_i)| \tag{3.48}$$

donde $\hat{y}(\vec{x}_i)$ es el valor obtenido utilizando un modelo de aproximación de $y(\vec{x})$ evaluado en el punto $\vec{x}_i$.

En la tabla 3.3 se muestra la evaluación de la fórmula 3.48 con puntos distribuidos de forma homogénea en S y con una distancia de 0.2 entre sí, dando un total de $n = 2601$ puntos formados por una rejilla de 51×51 puntos en la superficie $x_1 - x_2$.

Función base	Lineal	1497.1
	Tiras de láminas delgadas	1195.4
	Monomial	772.54
	Gaussiana	486.69
	Multi-cuadrática	1034.2
	Cuadrática inversa	554.7
	Multi-cuadrática inversa	648.42
Variograma	Lineal	1646
	Exponencial	1598
	Racional cuadrático	1228

Tabla 3.3: Medida del error de aproximación de funciones de interpolación.

Con estos datos nos es más sencillo apreciar que los mejores resultados son obtenidos con las RFBR utilizando las funciones base Gaussiana, cuadrática inversa y

multi-cuadrática inversa. En el método kriging la mejor aproximación se realiza utilizando un variograma racional cuadrático. Sin embargo, la similitud dista mucho de ser buena comparada con RFBR. Además de la falta de precisión del método kriging, la complejidad computacional del método es mayor, ya que es necesario entrenar los pesos $\vec{\lambda}$, por lo que dicha implementación en la aproximación de funciones no es la más adecuada.

CAPÍTULO 4

Optimización asistida por métodos de aproximación de funciones

En algunos problemas de optimización, la evaluación directa de la función objetivo requiere de un tiempo elevado. Para este tipo de problemas se puede utilizar un modelo aproximado a la función objetivo real, también llamado meta-modelo. El evaluar el meta-modelo requiere menos tiempo que la evaluación de la función objetivo original, por lo que se puede utilizar esta propiedad para realizar la optimización con algoritmos evolutivos, los cuales tienen como característica general el requerir de poblaciones de posibles soluciones, las cuales deben ser evaluadas. Al utilizar un meta-modelo, se resuelve el inconveniente de utilizar múltiples veces la función objetivo costosa.

Quizá el método más directo de utilizar un meta-modelo sería creándolo con un número fijo de soluciones pre-evaluadas directamente en la función objetivo original [56]. Sin embargo, como hemos visto en el capítulo 3, un meta-modelo ofrece poca fiabilidad en regiones lejanas a los puntos evaluados de forma exacta, por lo tanto, si no se tienen suficientes muestras de las regiones donde se encuentran las soluciones óptimas, será poco probable que con dicho meta-modelo se puedan obtener los resultados deseados.

Otra forma de usar un meta-modelo sería utilizando la aproximación únicamente en las regiones de interés, es decir, en las regiones donde se encuentran los óptimos, pero como no se tiene esa información *a priori* (en cuyo caso se tendría resuelto el problema de optimización), se requiere tener alguna idea de las regiones que conviene "poblar" de puntos de prueba para mejorar el meta-modelo en estas regiones. La manera de obtener información sobre qué regiones evaluar puede ser utilizando el algoritmo evolutivo con un meta-modelo inicial, al obtener los resultados de la optimización que éste realiza. Con esto se tendrá también una dirección de búsqueda, con lo que, mediante evaluaciones directas a la función objetivo, se podrá mejorar la calidad del meta-modelo para continuar con la optimización. Realizando este proceso en forma sucesiva, eventualmente se puede encontrar el conjunto de óptimos de Pareto del problema original. Este último esquema es el que se utilizó en esta tesis.

El algoritmo híbrido propuesto fue desarrollado en lenguaje C. El código del NSGA-II, que se utilizó como base, se descargó de la página del KanGAL (*Kampur Genetic Algorithms Laboratory* [1]), el cual fue modificado en algunos módulos para poder adecuarlo al algoritmo híbrido. Los métodos de aproximación de funciones, el algoritmo de k-medias, así como el resto de los elementos del algoritmo híbrido fueron programados completamente. A continuación se detalla la manera en que se implementó el algoritmo propuesto.

4.1. Descripción del algoritmo propuesto

El algoritmo propuesto se puede explicar y entender mediante las siguientes etapas:

1. Generación aleatoria de una población inicial $\mathcal{P}$ distribuida mediante *hipercubos latinos*.

2. *Entrenamiento del meta-modelo* utilizando $\mathcal{P}$.

3. *Proceso de sub-optimización* sobre el meta-modelo.

4. Selección de los 20 mejores resultados del proceso de sub-optimización , se agregan éstos a $\mathcal{P}$ y evaluación de los mismos.

5. Creación de una *población con diversidad* y se agregan a $\mathcal{P}$ en el caso de no encontrar en el paso anterior al menos 5 individuos distintos a los contenidos en $\mathcal{P}$.

6. En caso de que $|\mathcal{P}|$ sobrepase los 400 individuos, se aplicará un *proceso de reducción de la población de entrenamiento* para agilizar el entrenamiento del meta-modelo.

El procedimiento antes descrito se repite a partir del paso 2 hasta lograr la convergencia del algoritmo (o alcanzar un número máximo de evaluaciones).

En el capítulo 3 se realizó el estudio de los modelos de aproximación de funciones y en la tabla 3.3 se mostró que los mejores resultados fueron para la RFBR con una función base monomial, Gaussiana, cuadrática inversa y multicuadrática inversa. Por lo tanto, éstos son los algoritmos que se utilizaron para crear el meta-modelo, ofreciendo al usuario del algoritmo propuesto la posibilidad de seleccionar alguno de ellos al ejecutar el algoritmo. De tal forma, el algoritmo propuesto consiste de un híbrido entre un AEMO de segunda generación (el NSGA-II [20] en este caso) y un meta-modelo.

En los problemas de prueba utilizados (las funciones ZDT [85]) se conocen de antemano las soluciones óptimas, con lo cual se pudo mostrar que el algoritmo híbrido

[1] En la página web: *http://www.iitk.ac.in/kangal/index.shtml*

propuesto, en la mayoría de las funciones, converge con un menor número de evaluaciones a la función objetivo que el algoritmo NSGA-II original. Dichos resultados se mostrarán más adelante (en la sección 4.3 en la página 60 de análisis de resultados).

A continuación se hace una explicación más detallada de la forma en que se llevó a cabo la implementación de los procesos antes mencionados.

4.1.1. Población aleatoria bien distribuida

Como en un principio no se conoce cuál es la región donde es necesario realizar evaluaciones para detallar dicha zona, se debe crear un meta-modelo que cuente con un conjunto $\mathcal{P}$ de puntos pre-evaluados bien distribuidos sobre el espacio de búsqueda. En este caso, se utiliza el algoritmo de *hipercubos latinos* como generador aleatorio de una población bien distribuida.

En la población inicial es necesario tener puntos bien distribuidos en el espacio de búsqueda con el fin de realizar un modelo adecuado y poder guiar de forma correcta la optimización realizada por el NSGA-II. Con este fin se utiliza un algoritmo de generación de puntos distribuidos. Koehler y Owen [51] describen tres métodos para seleccionar valores de las variables de entrada; en esta tesis se utilizó el método de los hipercubos latinos.

Hipercubos latinos

El método estadístico de muestreo por hipercubos latinos fue diseñado una distribución de colecciones plausibles de valores de parámetros de una distribución multidimensional. En el contexto de muestreo estadístico, una malla cuadrada que contiene posiciones muestreadas, es un cuadrado latino si y sólo si existe una muestra en cada fila y en cada columna. Un hipercubo latino es una generalización de este concepto a un número arbitrario de dimensiones, por lo tanto, cada muestra es la única en cada hiperplano coordenado que lo contiene.

Si se tiene un campo restringido $\mathcal{S} \in \mathbb{R}^K$, este campo puede ser dividido en celdas, las cuales se construyen de la siguiente forma: cada rango de cada una de las K dimensiones de $\mathcal{S}$ debe ser dividido en N intervalos con probabilidad $1/N$. El producto cartesiano de esos intervalos dividirán a $\mathcal{S}$ en N^K celdas, cada una con probabilidad N^{-K}. Al seleccionar una celda i, ésta puede etiquetarse mediante un conjunto de K coordenadas $\mathbf{c}_i = (c_{i1}, c_{i2}, \ldots, c_{iK})$ donde m_{ij} es el número del intervalo de la componente j de $\mathcal{S}$ representado en la celda i. El muestreo de un hipercubo latino de tamaño N se obtiene de la selección aleatoria N de las celdas $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N$ con la condición de que, por cada componente j, el conjunto $\{c_{ij}\}_{i=1}^{N}$ sea una permutación de los enteros $1, \ldots, N$. Luego se realiza una observación aleatoria en cada celda.

Dos celdas $(l_1, \ldots, l_K)$ y $(m_1, \ldots, m_K)$ sin coordenadas en común pueden seleccionarse de la siguiente manera:

- Seleccionando de manera aleatoria dos enteros (R_{11}, R_{21}) sin reemplazo de los primeros N enteros $1, \ldots, N$.

- Sean entonces $l_1 = R_{11}$ y $m_1 = R_{21}$.
- Repetir el procedimiento para obtener $(R_{12}, R_{22}), (R_{13}, R_{23}), \ldots, (R_{1K}, R_{2K})$ y se asignarán $l_k = R_{1k}$ y $m_k = R_{2k}$.

De esta forma se han seleccionado dos celdas que cumplen con $l_k \neq m_k$ para $k = 1, \ldots, K$. El procedimiento general para crear una población bien distribuida basada en hipercubos latinos se muestra en el algoritmo 4.1. Como entrada al algoritmo se tienen los parámetros de la población, es decir, su tamaño, el número de dimensiones que se manejan para cada individuo así como los rangos en cada dimensión (Dim_{min}, Dim_{max}). Para cada dimensión se numeran las celdas en que será dividida, que son tantas como elementos tenga la población que se quiere inicializar. Esta numeración hace las veces de tómbola, que tiene como elementos los números correspondientes a las celdas $(\mathcal{C})$, se toma uno de ellos al azar $(\mathcal{S})$ y el valor de la celda seleccionado será ajustado para obtener un valor dentro de los límites establecidos para la dimensión en cuestión, el cual se situará en el centro de la celda seleccionada. Para cumplir con las características de un hipercubo latino, el valor de la celda seleccionada es desechado del conjunto. El procedimiento se repite para cada uno de los elementos en $\mathcal{C}$ y para todas las dimensiones.

Algoritmo 4.1: Algoritmo de inicialización de población

Entrada: Tamaño de población, dimensionalidad
Salida: Población distribuida aleatoriamente

1 N =tamaño de poblacion;
2 K =número de dimensiones;
3 **for** *d=1* **to** K **do**
4 $\quad \mathcal{C} = \{1, 2, \ldots, N\}$;
5 $\quad Cel = \dfrac{Dim^d_{max} - Dim^d_{min}}{N}$;
6 $\quad$ **for** i *= 1* **to** N **do**
7 $\quad\quad$ Sea $\mathcal{S}$ un elemento seleccionado aleatoriamente de $\mathcal{C}$;
8 $\quad\quad Ind^d_i = Cel \times \mathcal{S} - \dfrac{Cel}{2}$;
9 $\quad\quad \mathcal{C} = \mathcal{C} - \mathcal{S}$;
10 $\quad$ **end**
11 **end**

4.1.2. Entrenamiento del meta-modelo

Para realizar una aproximación de la función objetivo, es necesario contar con un conjunto de puntos previamente evaluados, los cuales nos sirven para entrenar el meta-modelo seleccionado. El algoritmo de aproximación utilizado en la tesis se basa en la RFBR, explicada anteriormente en el capítulo 3 debido a que se observaron

mejores aproximaciones al utilizar la RFBR con la función base monomial, Gaussiana, cuadrática inversa y multi-cuadrática inversa.

En el algoritmo híbrido propuesto se utiliza un conjunto $\mathcal{P}$ de puntos pre-evaluados y es con este conjunto de puntos con los que entrenamos en cada ciclo el meta-modelo.

En la RFBR, el entrenamiento consiste en encontrar los valores de $\vec{w}^*$ de la ecuación (3.15) (en la página 36). Sin embargo, primero se deben calcular los valores de la matriz $\mathbf{Z}$ mediante la ecuación (3.14). La selección del tipo de kernel $K(\rho)$ se realiza en tiempo de ejecución del algoritmo mediante parámetros de entrada. En esta implementación, el valor de $\sigma_j = 1, \quad j = 1, 2, \ldots, m$ se usa para acelerar el entrenamiento.

4.1.3. Proceso de sub-optimización

Una vez realizado el entrenamiento del meta-modelo, se aplica el NSGA-II durante *nGen* generaciones utilizando el meta-modelo como función objetivo. El valor de *nGen* no debiera ser muy grande, para evitar incrementar demasiado el tiempo de procesamiento del algoritmo, recordando que el proceso de sub-optimización se repetirá en cada ciclo del algoritmo principal.

La evaluación de un punto $\vec{x}$ en el meta-modelo se realiza mediante la ecuación (3.6) en la página 31. El pseudocódigo del proceso de sub-optimización se muestra a continuación:

Algoritmo 4.2: Algoritmo de sub-optimización

Entrada: *nGen*, Población de entrada $\mathcal{P}_e$
Salida: Población sub-optimizada $\mathcal{P}_s$

```
1 pobPadres ← P_e;
2 i = 0;
3 while i < nGen do
4     pobHijos_i = seleccion-cruza(pobPadres_i);
5     pobHijos_i = muta(pobHijos_i);
6     evalua-meta-modelo(pobHijos_i);
7     pobMezclada = mezcla(pobPadres_i, pobHijos_i);
8     pobPadres_{i+1} = elitismo-NSGA-II(pobMezclada, tamPob);
9 end
```

El algoritmo 4.2 utiliza los operadores de selección, cruza y mutación del NSGA-II. El algoritmo de elitismo del NSGA-II al que se hace referencia es el expuesto en el algoritmo 2.1 (en la página 26), con la diferencia de que en `elitismo-NSGA-II`() se especifica sólo una población de entrada y la población resultante tendrá un tamaño definido en los parámetros de la función.

El proceso de sub-optimización nos guiará a las regiones que es preciso detallar en la aproximación. El algoritmo NSGA-II es utilizado como algoritmo de optimización durante *nGen* generaciones. El resultado de este sub-proceso será la población $\texttt{pobPadres}_{nGen}$, de la cual se tomarán únicamente las mejores *nRed* soluciones

utilizando una vez más el criterio de elitismo del NSGA-II. La población generada *pobReduc* es una población reducida de 20 elementos, la cual se busca agregar a $\mathcal{P}$. Se realiza entonces una comparación de los puntos en *pobReduc* para evitar repeticiones en $\mathcal{P}$. Los puntos no repetidos serán evaluados directamente en la función objetivo original y son agregados a $\mathcal{P}$.

4.1.4. Población con diversidad

Dado el caso de que los resultados del proceso de sub-optimización devuelva puntos repetidos en $\mathcal{P}$ y que el número de puntos distintos sea menor a un umbral (en este trabajo se utilizó un umbral de 5), es necesario agregar diversidad al conjunto $\mathcal{P}$; de lo contrario, el meta-modelo formado será muy parecido al anterior y por lo tanto, existe una alta probabilidad de que los resultados de la sub-optimización sean muy similares, con lo cual se podría estancar el algoritmo híbrido. La forma de diversificar el conjunto $\mathcal{P}$ se realiza mediante el algoritmo 4.3 que se muestra a continuación:

Algoritmo 4.3: Algoritmo de diversificación de $\mathcal{P}$

Entrada: $\mathcal{P}$
Salida: Población con diversidad *pobReduc*

1 `pobPadres` = elitismo-NSGA-II($\mathcal{P}$, $tamPob$);
2 $\texttt{pobHijos}_i$ = seleccion-cruza($\texttt{pobPadres}_i$);
3 $\texttt{pobHijos}_i$ = muta($\texttt{pobHijos}_i$);
4 `pobReduc` = $\texttt{pobHijos}_i[1 : nRed]$;
5 $\mathcal{P} = \mathcal{P} +$ `pobReduc` ;

Se utiliza el proceso de elitismo del NSGA-II para obtener los mejores puntos de $\mathcal{P}$ y se asignan a una población llamada `pobPadres` a la cual se le aplican los procesos de selección y cruza para producir otra población llamada `pobHijos` a la que se le aplica mutación y del resultado, se seleccionan únicamente $nRed$ individuos que serán agregados a $\mathcal{P}$, verificando previamente que no existan repeticiones. Como puede apreciarse, el proceso de diversidad de la población es prácticamente una generación en el algoritmo del NSGA-II, por lo tanto, en caso de que el meta-modelo no aporte un avance en la búsqueda (debido a que se encuentren las mismas soluciones), el proceso de evolución no se queda estancado. De esta forma, las evaluaciones que se realicen servirán para guiar la búsqueda de soluciones y al mismo tiempo, le sirven al meta-modelo para realizar una aproximación más precisa de la función objetivo real en las áreas de interés.

4.1.5. Proceso de reducción de la población de entrenamiento

En los pasos anteriores, se describe cómo agregar nuevos puntos a $\mathcal{P}$, la cual consiste de un conjunto de puntos evaluados directamente en la función objetivo. Al tener más puntos pre-evaluados con los cuales entrenar el meta-modelo, éste se vuelve

más preciso en las regiones que el NSGA-II va explorando, por lo que este proceso se podría repetir hasta que, eventualmente, se alcance el verdadero frente de Pareto.

Sin embargo, el costo computacional del proceso de entrenamiento del modelo aproximado [2] se incrementa rápidamente al aumentar el número de puntos utilizados en el entrenamiento. En un principio, el meta-modelo se entrena con el número establecido en el tamaño de la población inicial y después se van agregando a lo más $optM$ puntos pre-evaluados en cada ciclo. Si se continuara con el mismo procedimiento, el número de elementos $|\mathcal{P}|$ se incrementaría permanentemente hasta la finalización del programa, lo cual haría más preciso el meta-modelo, pero su entrenamiento llevaría un tiempo excesivo. Por lo tanto, se establece un tamaño máximo para $|\mathcal{P}|$. En el presente trabajo se fijó este número a 400, debido a que con este número de puntos, el tiempo de entrenamiento aún es aceptable y se tiene un buen desempeño del algoritmo. Al llegar el algoritmo híbrido al número máximo de elementos de entrenamiento, se ejecuta un proceso de reducción de la población de entrenamiento, para que el tiempo de entrenamiento también sea reducido y así poder seguir agregando los resultados de la sub-optimización al conjunto de entrenamiento.

El proceso de reducción de la población de entrenamiento se describe en el algoritmo 4.4 que se muestra a continuación:

Algoritmo 4.4: Algoritmo de reducción de $\mathcal{P}$

Entrada: $\mathcal{P}$
Salida: $\mathcal{P}$ (reducido)

```
1 pobMejores = elitismo-NSGA-II(P, 100);
2 pobDistrib = k-medias(P,250);
3 P = pobMejores + pobDistrib;
```

Para controlar el incremento del número de puntos utilizados en el entrenamiento del meta-modelo se utiliza una vez más el proceso de elitismo del NSGA-II para obtener la población `pobMejores`, que son los 100 mejores puntos del conjunto $\mathcal{P}$. Con este procedimiento evitamos perder los mejores puntos ya evaluados y a su vez, mantenemos información para mejorar la aproximación en dichas zonas. Después se obtiene la población `pobDistrib`, que son los centros obtenidos de aplicar el algoritmo de k-medias (ver algoritmo 4.5). Existen regiones en el meta-modelo con una mala aproximación, donde podría ocurrir que el NSGA-II encontrase soluciones al utilizar el meta-modelo, sin embargo, en el modelo real corresponden a regiones no óptimas. Aunque los puntos correspondientes a zonas no óptimas no sirven directamente para la optimización, estos puntos son útiles para obtener una aproximación adecuada en dichas regiones, de tal forma que no volverán a ser indicadas como regiones óptimas por el NSGA-II. Por lo tanto es importante mantener algunas de las soluciones "malas" para ayudar a que el meta-modelo indique al NSGA-II las regiones donde no existen soluciones óptimas. Sin embargo, no se pueden tener todos estos puntos

[2] $O(n^3)$ debido a la inversión de la matriz $\mathbf{Z}$ en la ecuación (3.15), donde n es el número de puntos pre-evaluados.

durante todo el proceso de optimización, por lo que se utiliza el algoritmo de k-medias para obtener puntos representativos de las regiones previamente evaluadas. El algoritmo de k-medias se aplica al conjunto $\mathcal{P}$ y se obtienen 250 centroides. De la suma de `pobMejores` y de `pobDistrib` se tienen 350 puntos. Sin embargo, en la práctica $|\mathcal{P}|$ se reduce a un aproximado de 220 a 250 puntos debido a las repeticiones obtenidas en el proceso, las cuales son eliminadas.

En el algoritmo 4.5, se definen inicialmente los k centros, tomando de forma aleatoria k elementos contenidos en $\mathcal{P}$. Después se inicia un ciclo en el cual se clasifica cada elemento $\mathcal{P}^l \quad \forall l = 1, 2, \ldots, |\mathcal{P}|$, asignándole una clase M de la forma $\mathcal{P}^l[M]$ si el centro $\mathcal{C}^M$ es el más cercano al elemento en cuestión. Una vez realizada la clasificación de $\mathcal{P}$, se recalculan los centros, creando los nuevos centros $\mathcal{C}_{i+1}$ a partir del cálculo del promedio de los puntos con la misma clasificación $\mathcal{P}[M]$. El algoritmo converge cuando no existan cambios al recalcular los centros. En la implementación realizada, no se requieren los centros finalmente calculados, sino los puntos pre-evaluados en $\mathcal{P}$ que sean los más cercanos a cada uno de los centros $\mathcal{C}_i$, para no evaluar puntos adicionales.

Algoritmo 4.5: Algoritmo de k-medias

Entrada: $\mathcal{P}$,k
Salida: $\mathcal{C}_R$
$\mathcal{C}_0$= obtiene-k-elementos($\mathcal{P}$,k);
$i = 0$;
repeat
    $\mathcal{P}[M]$=clasifica($\mathcal{P}$,$\mathcal{C}_0$);
    for $j = 1$ **to** k **do**
        $\mathcal{C}_{i+1}^j = \frac{\sum^{\forall l} \mathcal{P}^l[j]}{|\mathcal{P}[j]|}$;
    end
    $i = i + 1$;
until $\mathcal{C}_i = \mathcal{C}_{i-1}$;
$\mathcal{C}_R$ = masCercano($\mathcal{C}_i$, $\mathcal{P}$);

4.2. Pseudocódigo

El pseudocódigo del algoritmo híbrido propuesto en esta tesis se muestra a continuación:

Algoritmo 4.6: Algoritmo híbrido propuesto

Entrada: Parametros de entrada
Salida: Población optimizada $\mathcal{R}$

```
1  P = inicializaPoblacion(|P|, dimensionalidad);
2  evaluacionDirecta(P);
3  numEval = |P|;
4  while numEval < maxEv do
5      entrenaMetamodelo(P);
6      pobSuboptimizada = sub-optimizacion(P,genInt);
7      pobReducida = elitismo-NSGA-II(pobSuboptimizada, 20);
8      pobReducida = eliminaRepeticiones(P, pobReducida);
9      evaluacionDirecta(pobReducida); numEval = numEval + |pobReducida|;
10     P = P + pobReducida;
11     if |pobReducida| < 5 then
12         pobDiversa = diversifica(P);
13         pobDiversa = eliminaRepeticiones(P, pobDiversa);
14         evaluacionDirecta(pobDiversa); numEval = numEval + |pobDiversa|;
15         P = P + pobDiversa;
16     end
17     if |P| > 400 then
18         P = reduccion(P );
19     end
20 end
21 R = noDominados(P);
```

En el algoritmo 4.6 se tienen como parámetros de entrada los requeridos por el NSGA-II, además de parámetros que especifican el número de generaciones $genInt$ utilizadas en el proceso de sub-optimización y el número máximo de evaluaciones directas a la función objetivo $maxEv$.

Primero se realiza la inicialización de la población $\mathcal{P}$, la cual inicia con el número de individuos definido para las poblaciones del NSGA-II utilizado en el proceso de sub-optimización. Se realiza la evaluación directa de los individuos en $\mathcal{P}$ y se inicia el conteo del número de evaluaciones realizadas. Después se inicia un ciclo que terminará cuando se rebase el número de evaluaciones directas a la función objetivo real. En el ciclo se realiza el entrenamiento del meta-modelo; dicho proceso utiliza los individuos en $\mathcal{P}$ como puntos de entrenamiento. Del resultado del proceso de sub-optimización, se obtiene la población `pobSuboptimizada`, de la cual se escogen los mejores 20 puntos, los cuales formarán la población `pobReducida`. De esta población se eliminan los individuos que ya se encuentran en $\mathcal{P}$ para evitar evaluar más de una vez una misma solución. Se realiza la evaluación directa a la función objetivo de los elementos en `pobReducida` y se actualiza el contador $numEval$ de evaluaciones directas realizadas. Finalmente se agregan los elementos evaluados a la población de entrenamiento $\mathcal{P}$. En el caso de que el número de individuos agregados a $\mathcal{P}$ sea menor

a 5, la diferencia entre el meta-modelo anterior y el resultante de agregar sólo 5 elementos puede no ser significativa, por lo tanto se realiza el proceso de diversificación, para buscar agregar más elementos al conjunto de entrenamiento.

Cuando el número de individuos de $\mathcal{P}$ es mayor a 400, el conjunto de entrenamiento se reduce en número para continuar agregando elementos a dicho conjunto y con esto mantener acotado el tiempo requerido para el entrenamiento del meta-modelo.

4.3. Análisis de resultados

En este capítulo se presentan los resultados del algoritmo propuesto, el cual fue descrito en la sección anterior. Se denominó AGMOCAF (Algoritmo Genético Multi-Objetivo Con Aproximación de Funciones) al algoritmo híbrido propuesto, el cual será comparado en esta sección con el NSGA-II original.

Con el fin de comparar dos algoritmos multi-objetivo, es necesario cuantificar su desempeño en un conjunto de problemas de prueba, los cuales deben ser conocidos por su dificultad y de los que se conozca la localización exacta de las soluciones Pareto-óptimas, tanto en el espacio de las variables de decisión, como en el de sus objetivos. Es por esto que se ha decidido utilizar el conjunto de funciones de prueba ZDT [85], el cual ha sido utilizado anteriormente para mostrar el buen desempeño del NSGA-II [20]. Evidentemente el reto fue mejorar que el híbrido propuesto puede lograr mejores resultados que el NSGA-II cuando se realiza un número reducido de evaluaciones de la función objetivo.

La comparación puede ser realizada de forma visual, sin embargo, para mostrar de manera formal la mejora del AGMOCAF frente al NSGA-II, serán utilizadas algunas métricas de uso común en la literatura especializada. Con el fin explicar en qué consisten estas métricas y cuál es el significado de los resultados obtenidos, a continuación se hará una breve descripción de las mismas.

4.3.1. Métricas

Existen dos metas en la optimización multi-objetivo [18, pág. 307]:

- Descubrir soluciones lo más cercanas posible a las soluciones Pareto-óptimas.
- Encontrar soluciones tan diversas como sea posible en el frente no dominado obtenido.

La primera meta requiere una búsqueda hacia la región Pareto-óptima, mientras que la segunda requiere una búsqueda a lo largo del frente Pareto-óptimo. Por lo tanto, una sola métrica no puede ser suficiente para decidir el desempeño de un algoritmo evolutivo multi-objetivo.

En esta tesis se utilizarán dos métricas para evaluar la proximidad al frente óptimo de Pareto de las soluciones (*distancia generacional invertida* y *cobertura de conjuntos*) y una para evaluar la diversidad entre las soluciones no-dominadas (*distribución*).

Métrica de distancia generacional invertida ($\mathcal{DGI}$)

Esta métrica se basa en otra, llamada distancia generacional (DG). La DG fue propuesta por David Van Veldhuizen y Gary Lamont [78, 77] y consiste en encontrar una distancia promedio de las soluciones de Q desde P^* de la manera siguiente:

$$DG = \frac{\sqrt{\sum_{i=1}^{|Q|} d_i^2}}{|Q|} \tag{4.1}$$

En la ecuación (4.1), el parámetro d_i es la distancia Euclidiana (en el espacio de los objetivos) entre la solución $i \in Q$ y el miembro más cercano de P^*:

$$d_i = \min_{k=1}^{|P^*|} \sqrt{\sum_{m=1}^{M} (f_m^{(i)} - f_m^{*(k)})^2} \tag{4.2}$$

donde $f_m^{*(k)}$ es el valor de la m-ésima función objetivo del k-ésimo miembro de P^*.

La $\mathcal{DGI}$ utiliza el mismo concepto, sin embargo, a diiferencia de la DG, se utiliza como base el verdadero frente de Pareto P^*.

Métrica de cobertura de conjuntos ($\mathcal{CC}$)

La métrica $\mathcal{CC}$ fue propuesta por Eckart Zitzler y Lothar Thiele en 1998 [86]. La métrica también puede ser usada para tener una idea de la distribución relativa de soluciones entre dos conjuntos de vectores de soluciones A y B. La métrica $\mathcal{CC}(A, B)$ calcula la proporción de soluciones en B que son dominadas o iguales a las soluciones en B:

$$\mathcal{CC}(A, B) = \frac{|\{b \in B; \exists a \in A : a \preceq b\}|}{|B|} \tag{4.3}$$

Si todos los miembros de B son dominados por los de A, la métrica $\mathcal{CC}(A, B) = 1$. Por otra parte, si ningún punto de A domina a algún punto de B, entonces $\mathcal{CC}(A, B) = 0$. Debido a que el operador de dominancia no es simétrico, se necesita calcular $\mathcal{CC}(A, B)$ y $\mathcal{CC}(B, A)$ para comprender cuántas soluciones de A son cubiertas por B y viceversa.

Métrica de distribución ($\mathcal{D}$)

La métrica $\mathcal{D}$ fue propuesta por Deb et al. en el año 2000 [21]. Mide la distribución de las soluciones obtenidas por un algoritmo. La métrica se encuentra definida en la siguiente ecuación:

$$\mathcal{D} = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} |d_i - \overline{d}|}{\sum_{m=1}^{M} d_m^e + |Q|\overline{d}} \tag{4.4}$$

donde d_i puede ser cualquier medida de distancia entre soluciones vecinas (puede utilizarse la ecuación (4.2)). El parámetro d_m^e es la distancia entre las soluciones extremo de P^* y Q correspondientes a la m-ésima función objetivo.

4.3.2. Comparación de resultados

El objetivo de principal de esta tesis es diseñar un algoritmo híbrido que reduzca el número de evaluaciones de la función objetivo mediante el uso de métodos de aproximación de funciones. Por lo tanto, para mostrar la capacidad del AGMOCAF para llegar a soluciones aceptables, se compara con el algoritmo NSGA-II, que es un algoritmo del estado del arte. El número de evaluaciones de la función objetivo se fijó a 2,000 para las funciones ZDT [85] utilizadas, las cuales se describen en el apéndice A. Sin embargo, cabe destacar que en la función ZDT1 se alcanza el frente de Pareto real con únicamente 800 evaluaciones de la función objetivo y en la función ZDT2 con sólo 500. Debido a que el número mínimo de evaluaciones, requerido por nuestro algoritmo para alcanzar el frente de Pareto en cada función de prueba, varía considerablemente, se observó que 2,000 evaluaciones es una cantidad intermedia y es posible notar que nuestro algoritmo mejora notablemente al NSGA-II en la mayoría de las funciones de prueba.

Con el número fijo de evaluaciones a la función objetivo no se logra alcanzar el frente de Pareto óptimo de las funciones ZDT3 y ZDT4, pero incrementando el número de evaluaciones permitidas se alcanza dicho frente (ver apéndice B). Se eligió un número reducido de evaluaciones debido al objetivo de la aplicabilidad del AGMOCAF, el cual se pretende utilizar en problemas donde la evaluación de la función objetivo es costosa, por lo que la competitividad de los algoritmos, en este caso, debe mostrarse con soluciones cercanas o en el frente de Pareto real del problema.

En esta sección se muestra el estudio comparativo realizado entre el AGMOCAF y el NSGA-II, ambos descritos anteriormente en las secciones 2.3 y 4.1, respectivamente.

Los parámetros utilizados en los algoritmos se muestran en la tabla 4.1.

Parámetro	AGMOCAF	NSGA-II
T_p	100	100
P_c	0.9	0.9
P_m	$1/N$	$1/N$
η_c	15	15
η_m	20	20
G_i	40-140	–
AF	Gaussiana	–

Tabla 4.1: Parámetros utilizados en la comparación de algoritmos

En la tabla 4.1, T_p representa el tamaño de población, P_c el porcentaje de cruza del algoritmo evolutivo, P_m es el porcentaje de mutación, que depende del número de variables N del problema, η_c es el índice de distribución para la cruza (simulación

binaria [19]) y η_m es el índice de distribución para la mutación utilizada por el algoritmo NSGA-II (mutación polinomial [17]). Los parámetros G_i son el número de generaciones que se realizan en el proceso de sub-optimización del AGMOCAF y AF es el método de aproximación de funciones utilizado. El parámetro G_i tiene un rango debido a que se utilizaron distintos valores dependiendo del problema, ya que éstos no son de igual dificultad. En la tabla 4.2 se muestran los valores empleados para cada problema de prueba.

Problema	G_i
ZDT1	40
ZDT2	40
ZDT3	80
ZDT4	140
ZDT6	120

Tabla 4.2: Valores empleados de G_i para cada problema

Resumiendo el proceso de comparación, se realizaron 30 ejecuciones de los algoritmos para cada problema de prueba. Las ejecuciones se restringieron a 2,000 evaluaciones de la función objetivo. Por cada función de prueba se mostrarán tres tablas, correspondientes a las métricas $\mathcal{DGI}$, $\mathcal{CC}$ y $\mathcal{D}$. Cada tabla tendrá los valores de la media y la desviación estándar σ de cada algoritmo. Además, para tener una referencia adicional de comparación, se mostrarán las gráficas correspondientes a la media respecto a la métrica $\mathcal{DGI}$ en cada algoritmo.

Función ZDT1

En la tabla 4.3 se puede observar que, para la función ZDT1, el AGMOCAF es mejor que el NSGA-II tanto en las métricas de aproximación al frente de Pareto, así como en la de distribución en la media; sin embargo, con respecto a la desviación estándar, el NSGA-II tiene mejores valores que el AGMOCAF. Esto no significa que el NSGA-II sea mejor que el nuestro, sino que éste se acerca a la media en las 30 corridas realizadas.

ZDT1				
Métrica	AGMOCAF		NSGA-II	
	media	σ	media	σ
$\mathcal{DGI}$	0.015112	0.060189	0.154848	0.021058
$\mathcal{CC}$	0.031667	0.170530	0.960870	0.181136
$\mathcal{D}$	0.332647	0.156692	0.681252	0.064496

Tabla 4.3: Métricas de la función ZDT1

En la figura 4.1 se muestra gráficamente como el AGMOCAF alcanza el frente de

Pareto real de la función, mientras que el NSGA-II no alcanza siquiera a colocar un punto en dicho frente.

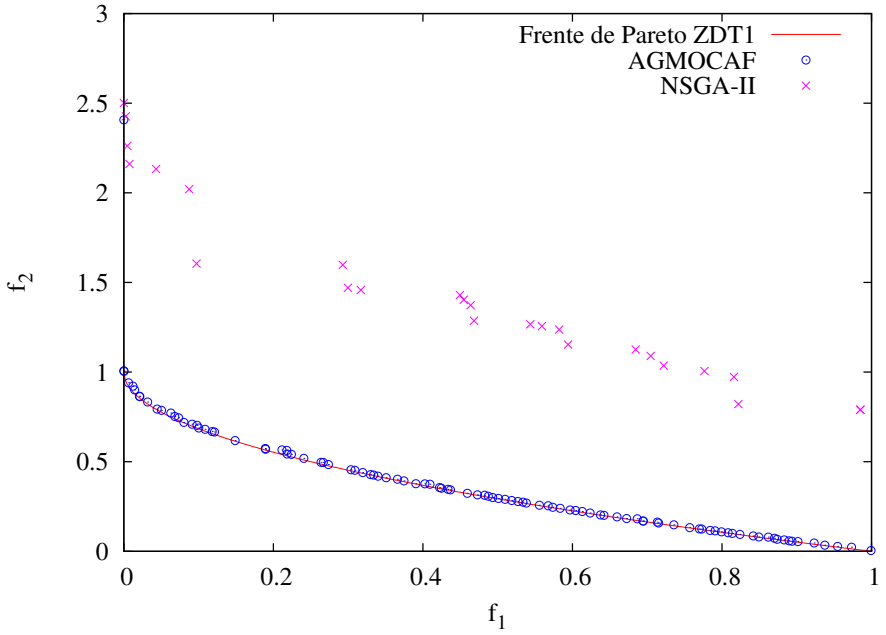

Figura 4.1: Comparación de resultados del AGMOCAF y NSGA-II en la función ZDT1.

Función ZDT2

En la función ZDT1, el resultado también es positivo para el AGMOCAF respecto al NSGA-II en las métricas realizadas como se puede apreciar en la tabla 4.4. En este caso, el NSGA-II es mejor en la desviación estandar de la métrica de distribución, que como se explicó anteriormente no significa que el NSGA-II sea mejor que el AGMOCAF.

ZDT2				
Métrica	AGMOCAF		NSGA-II	
	media	σ	media	σ
$\mathcal{DGI}$	0.007249	0.025693	0.342095	0.059717
$\mathcal{CC}$	0.000000	0.000000	0.920729	0.129333
$\mathcal{D}$	0.367405	0.273210	0.870624	0.055276

Tabla 4.4: Métricas de la función ZDT2

En la figura 4.2 se muestran los resultados obtenidos en la media de las 30 corridas. Gráficamente se puede apreciar que los resultados del AGMOCAF son mejores que los obtenidos por el NSGA-II.

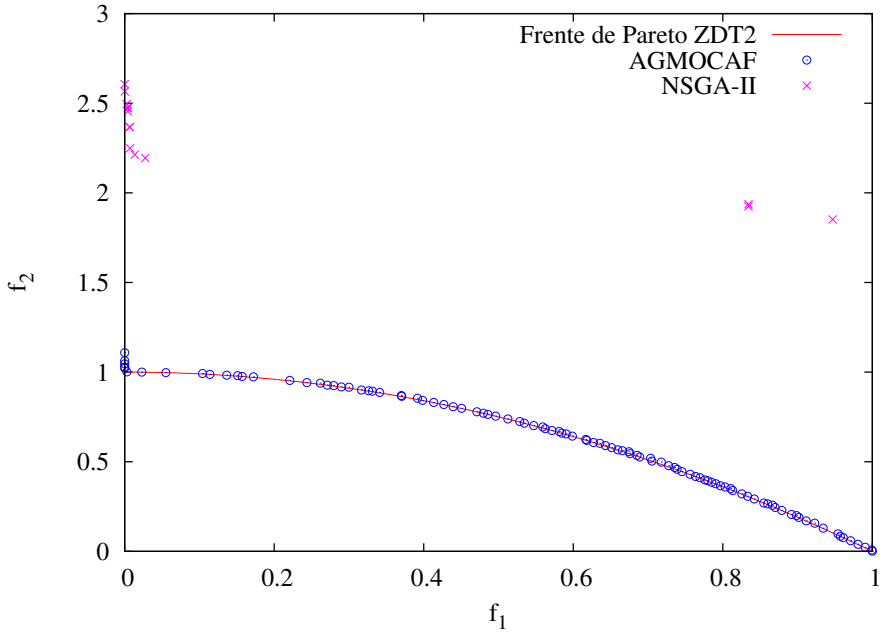

Figura 4.2: Comparación de resultados del AGMOCAF y NSGA-II en la función ZDT2.

Función ZDT3

En esta función, también se obtienen mejores resultados en la media, como puede apreciarse en la tabla 4.5. En la desviación estándar se observan mejores valores para el NSGA-II en las métricas $\mathcal{DGI}$ y $\mathcal{D}$. Sin embargo la diferencia con los valores del AGMOCAF no son muy grandes.

ZDT3				
Métrica	AGMOCAF		NSGA-II	
	media	σ	media	σ
$\mathcal{DGI}$	0.118214	0.049984	0.132451	0.025197
$\mathcal{CC}$	0.012617	0.050701	0.958510	0.083734
$\mathcal{D}$	0.748945	0.070189	0.766034	0.042966

Tabla 4.5: Métricas de la función ZDT3

En la figura 4.3 se aprecia gráficamente cómo las soluciones obtenidas por el AGMOCAF se encuentran más cercanas al verdadero frente de Pareto, sin embargo, también se observa para esta solución que no se cubre el cuarto frente (de izquierda a derecha), ya que este problema presenta frentes discontinuos. La comparación gráfica no implica que nuestro algoritmo no tenga una buena dispersión de las soluciones, sino simplemente que la solución en la media de la métrica $\mathcal{DGI}$ tuvo estas características.

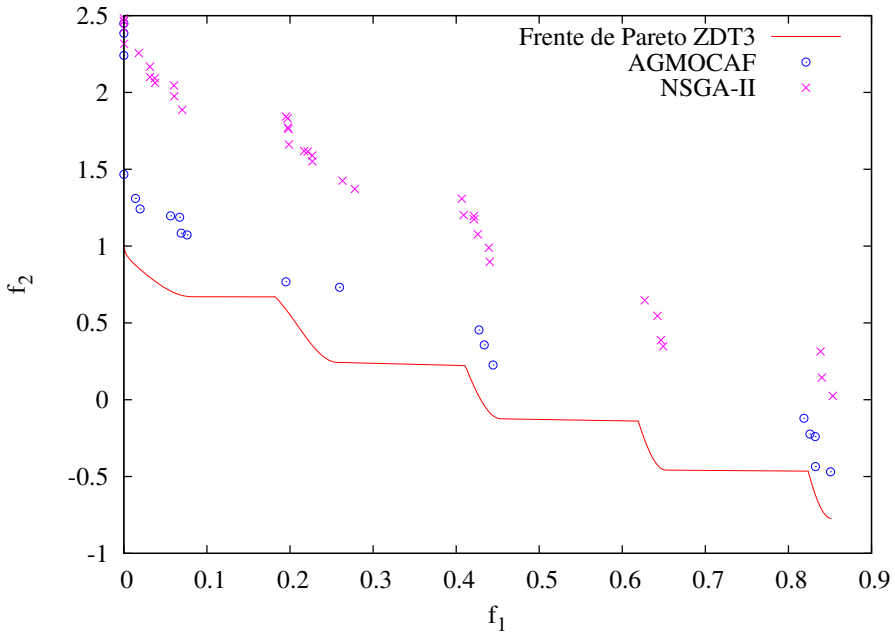

Figura 4.3: Comparación de resultados del AGMOCAF y NSGA-II en la función ZDT3.

Función ZDT4

En la tabla 4.6 podemos observar que el NSGA-II supera en todas las métricas al AGMOCAF. El mal desempeño de nuestro algoritmo se atribuye a que no se encuentra la zona que es necesario aproximar en el número de evaluaciones fijadas. Sin embargo, en los experimentos se mostró que con algunas evaluaciones más se logra mejorar al NSGA-II, aunque se fijó el número de evaluaciones a 2,000 a fin de permitir una comparación justa de resultados.

ZDT4				
Métrica	AGMOCAF		NSGA-II	
	media	σ	media	σ
$\mathcal{DGI}$	18.306819	4.385379	11.175525	3.527664
$\mathcal{CC}$	0.228319	0.212550	0.448832	0.272924
$\mathcal{D}$	0.993982	0.005593	0.987492	0.007164

Tabla 4.6: Métricas de la función ZDT4

Gráficamente se puede apreciar en la figura 4.4 la magnitud de la ventaja que tienen las soluciones del NSGA-II respecto a nuestra implementación, la cual no es mucha. También se aprecia que ambas soluciones carecen de una buena dispersión tras realizar 2,000 evaluaciones.

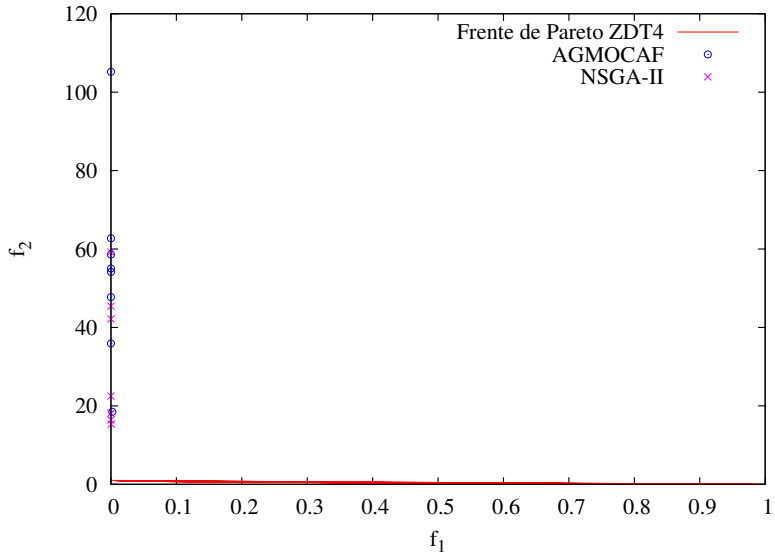

Figura 4.4: Comparación de resultados del AGMOCAF y NSGA-II en la función ZDT4.

Función ZDT6

De acuerdo a los resultados de la tabla 4.7, el AGMOCAF supera al NSGA-II en todas las métricas. También se puede apreciar que las desviaciones estándar son menores para el AGMOCAF, a excepción de la desviación estándar de la métrica $\mathcal{D}$. En la métrica $\mathcal{CC}$ se aprecia que, de acuerdo a los resultados, todas las soluciones obtenidas por el AGMOCAF dominan a los del NSGA-II y eso sucede en todos los casos, ya que se reporta desviación estándar de cero en dicha métrica.

ZDT6				
Métrica	AGMOCAF		NSGA-II	
	media	σ	media	σ
$\mathcal{DGI}$	0.221356	0.145131	1.280730	0.179904
$\mathcal{CC}$	0.000000	0.000000	1.000000	0.000000
$\mathcal{D}$	0.805382	0.108052	0.920585	0.036648

Tabla 4.7: Métricas de la función ZDT6

En la figura 4.5, podemos apreciar la ventaja que nuestro algoritmo tiene sobre el NSGA-II. Se puede observar que, a pesar de realizar únicamente 2000 evaluaciones, algunas soluciones del AGMOCAF tocan el verdadero frente de Pareto, mientras que las soluciones del NSGA-II se encuentran a una distancia considerable y ninguna solución toca el frente.

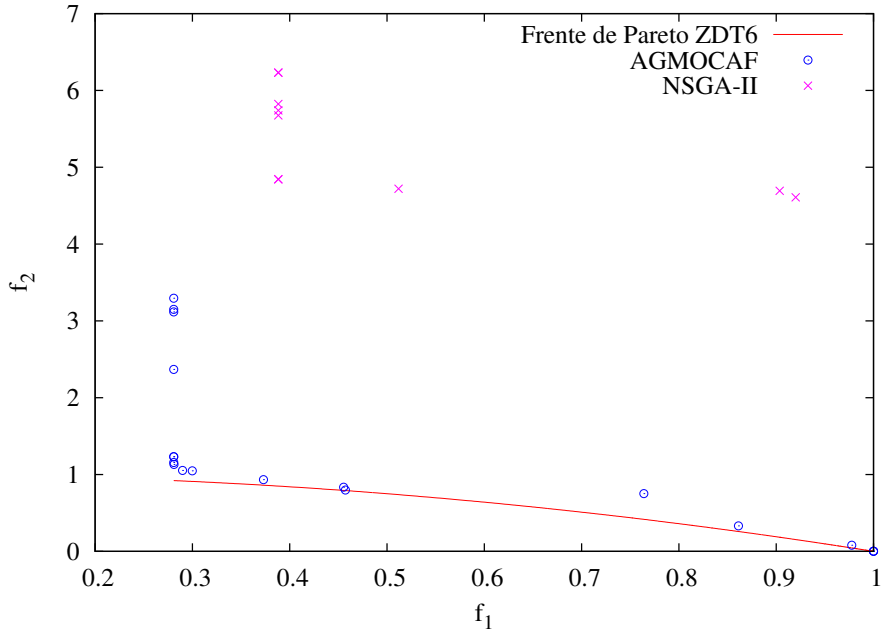

Figura 4.5: Comparación de resultados del AGMOCAF y NSGA-II en la función ZDT6.

4.4. Conclusiones sobre los resultados obtenidos

El algoritmo propuesto, el AGMOCAF, mostró ser mejor en todas las funciones de prueba a excepción de la ZDT4, donde el NSGA-II obtiene una ventaja marginal sobre nuestro algoritmo. Sin embargo, en las pruebas donde nuestro algoritmo demuestra ser mejor, llega al verdadero frente de Pareto, como es el caso de las funciones ZDT1 y ZDT2, se toca con algunos puntos el frente, como es el caso de la ZDT6, o se aproxima suficientemente, como en la ZDT3. Cabe recalcar que son necesarias pocas evaluaciones para llegar al verdadero frente en la funciones ZDT1 y ZDT2, que en nuestros experimentos requirieron únicamente de 650 evaluaciones en promedio.

El objetivo de obtener buenos resultados con pocas evaluaciones se cumple, aunque es evidente que, en problemas en los que se requiera de una mayor exploración, será necesario fijar un número mayor de evaluaciones reales e incluso incrementar el número de generaciones del proceso de sub-optimización G_i. Sin embargo, al utilizar el AGMOCAF en problemas del mundo real, puede reinicializarse el proceso partiendo de un modelo aproximado creado por los puntos evaluados en la ejecución anterior, aprovechando la característica de que se pueden almacenar las evaluaciones directas.

En general se puede decir que nuestro algoritmo es mejor que el NSGA-II en cuanto a la reducción de evaluaciones de la función objetivo requeridas para la convergencia del mismo. Las características de las funciones de prueba utilizadas se muestran en el Apéndice A. Las gráficas de convergencia del AGMOCAF se muestran en el Apéndice B.

CAPÍTULO 5

Conclusiones y trabajo a futuro

5.1. Conclusiones

Mediante el análisis de los resultados obtenidos en la presente tesis, llegamos a la conclusión de que el algoritmo propuesto AGMOCAF supera al algoritmo del estado del arte NSGA-II cuando se usa un número reducido de evaluaciones de la función objetivo. Con excepción de una función de prueba, nuestro AGMOCAF es considerablemente mejor en las funciones de prueba utilizadas con respecto a todas las métricas adoptadas. A continuación se mencionan los puntos más importantes que resumen las conclusiones de este trabajo de tesis:

1. El hacer uso de un método de aproximación de funciones es una técnica útil para reducir el número de evaluaciones directas a la función objetivo.

2. El éxito de la implementación de los métodos de aproximación de funciones depende de la ubicación de los puntos de muestra requeridos para su entrenamiento.

3. Al aplicar un método de aproximación de funciones a un algoritmo evolutivo es necesario modelar de forma precisa las zonas donde el AE se encuentra realizando la exploración, pero también se debe contar con muestras de regiones lejanas a las zonas de exploración para evitar que el AE explore nuevamente zonas donde se tengan malos resultados al realizar la evaluación directa, ya que la predicción hecha por el meta-modelo no es confiable en zonas lejanas a los puntos de prueba utilizados para entrenarlo.

4. Al momento de decidir cuales serán los puntos que deben emplearse podemos hacer uso del algoritmo de k-medias, con el fin de limitar la cantidad de puntos utilizados para crear el meta-modelo. Sin embargo deben usarse también todos los puntos donde el AE se encuentra realizando la exploración. De no ser así,

se puede perder la propiedad de elitismo en un AE y el meta-modelo podría no ser lo suficientemente preciso en las regiones de más importancia.

5. El algoritmo propuesto reduce el número de evaluaciones requeridas para alcanzar el frente de Pareto. Sin embargo, puede darse el caso de que, con pocas evaluaciones, no se tengan muestras de la zona donde se encuentran las soluciones óptimas, y en ese caso, el modelo aproximado no será preciso en dicha zona y no se llegará al frente de Pareto. No obstante, hemos mostrado experimentalmente que, en esos casos, basta con incrementar el número de evaluaciones para llegar al frente de Pareto.

6. Las RFBR resultaron ser un método lo suficientemente preciso y eficiente para crear un meta-modelo que represente a una función objetivo para usarse en un algoritmo genético.

5.2. Trabajo a futuro

Existen algunas inquietudes e ideas que debido a las limitantes obvias de tiempo para realizar esta tesis, no fueron realizadas y que, probablemente ofrezcan mejoras al algoritmo que se presenta en este trabajo:

1. Agregar un mecanismo que decida cuándo explotar más la búsqueda mediante el meta-modelo y cuándo mediante el algoritmo evolutivo base.

2. Utilizar otro tipo de algoritmo evolutivo multi-objetivo como motor de búsqueda base.

3. Idear un método que determine una distancia umbral entre un punto no evaluado y el punto pre-evaluado más cercano, de forma que se decida si el valor obtenido por el meta-modelo es suficientemente preciso o es necesario realizar la evaluación directa a la función objetivo.

4. En el trabajo se experimentó con el método kriging utilizando variogramas para determinar los valores de los parámetros requeridos para entrenar el meta-modelo. Sin embargo, existe la opción de estimar los parámetros requeridos mediante el método de máxima verosimilitud directamente de la especificación de las condiciones en el método kriging [15, pág. 142]. Se pueden obtener de esta manera los parámetros y determinar si de esta forma el método kriging ofrece mejores resultados que las RFBR.

APÉNDICE A

Funciones de prueba

A.1. Función ZDT1

Esta función fue propuesta por Zitzler, Deb y Thiele [85] y consiste en:

$$ZDT1 : \begin{cases} \text{Minimizar} & f_1(x) = x_1, \\ & g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i, \\ & h(f_1, g) = 1 - \sqrt{f_1/g}, \\ \text{Minimizar} & f_2(x) = g(x)h(f_1(x), g(x)), \\ & 0 \leq x_i \leq 1 \qquad \text{i=1, } \ldots 30. \end{cases} \tag{A.1}$$

Cuenta con un frente de Pareto convexo y conectado.

A.2. Función ZDT2

Esta función fue propuesta por Zitzler, Deb y Thiele [85] y consiste en:

$$ZDT2 : \begin{cases} \text{Minimizar} & f_1(x) = x_1, \\ & g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i, \\ & h(f_1, g) = 1 - (f_1/g)^2, \\ \text{Minimizar} & f_2(x) = g(x)h(f_1(x), g(x)), \\ & 0 \leq x_i \leq 1 \qquad \text{i=1, } \ldots 30. \end{cases} \tag{A.2}$$

Posee un frente de Pareto no convexo y conectado.

A.3. Función ZDT3

Esta función fue propuesta por Zitzler, Deb y Thiele [85] y consiste en:

$$ZDT3: \begin{cases} \text{Minimizar} & f_1(x) = x_1, \\ & g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i, \\ & h(f_1, g) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1), \\ \text{Minimizar} & f_2(x) = g(x)h(f_1(x), g(x)), \\ & 0 \leq x_i \leq 1 \qquad \text{i=1, } \ldots 30. \end{cases} \tag{A.3}$$

Tiene un frente de Pareto discontinuo segmentado en cinco partes.

A.4. Función ZDT4

Esta función fue propuesta por Zitzler, Deb y Thiele [85] y consiste en:

$$ZDT4: \begin{cases} \text{Minimizar} & f_1(\vec{x}) = x_1, \quad 0 \leq x_1 \leq 1 \\ & g(\vec{x}) = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i)), \\ & h(f_1, g) = 1 - \sqrt{f_1/g}, \\ \text{Minimizar} & f_2(\vec{x}) = g(\vec{x})h(f_1, g), \\ & -5 \leq x_i \leq 5 \qquad \text{i=2, } \ldots 10. \end{cases} \tag{A.4}$$

Posee un frente óptimo de Pareto convexo y conectado. Existen 21^9 frentes de Pareto locales.

A.5. Función ZDT6

$$ZDT6: \begin{cases} \text{Minimizar} & f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1), \\ & g(x) = 1 + 9\left[\left(\sum_{i=2}^{10} x_i\right)/9\right]^{0.125}, \\ & h(f_1, g) = 1 - (f_1/g)^2, \\ \text{Minimizar} & f_2(x) = g(x)h(f_1(x), g(x)), \\ & 0 \leq x_i \leq 1 \qquad \text{i=1, } \ldots 10. \end{cases} \tag{A.5}$$

Tiene un frente de Pareto no convexo y continuo.

APÉNDICE B

Convergencia del AGMOCAF en las funciones de prueba

En este apéndice se presentan las gráficas de convergencia del algoritmo propuesto AGMOCAF a fin de validar la convergencia del algoritmo al frente de Pareto real de las funciones de prueba adoptadas.

El número de evaluaciones de la función objetivo no se fijó, debido a que se busca destacar el reducido número de evaluaciones requerido en algunas funciones de prueba, como fue el caso de las funciones ZDT1 que requirió de 800 evaluaciones y la función ZDT2, que únicamente requiere de 500 evaluaciones para alcanzar el frente de Pareto real. Para la función ZDT3 se necesitaron 10,000 evaluaciones de la función objetivo. En la función ZDT4 se alcanzó el frente de Pareto hasta las 25,000 evaluaciones. Finalmente para la función ZDT6 fue posible llegar al frente con 8,000 evaluaciones de la función objetivo.

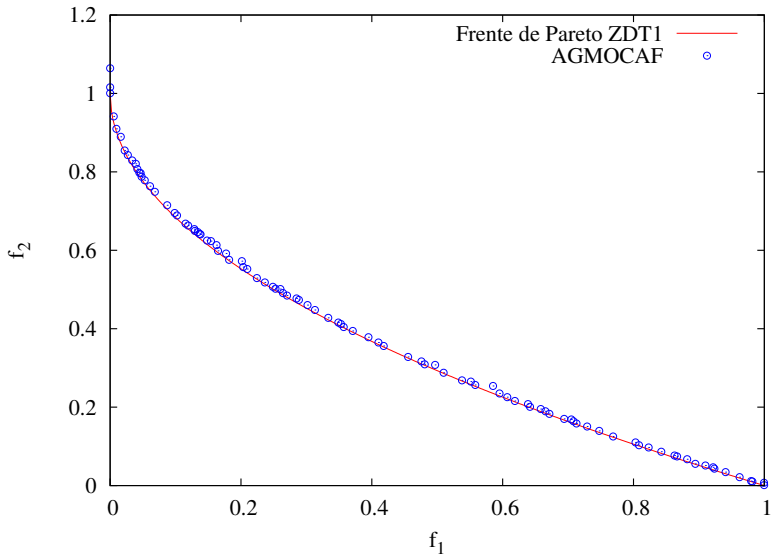

Figura B.1: Gráfica de convergencia del algoritmo AGMOCAF en la función ZDT1 (utilizando 800 evaluaciones de la función objetivo).

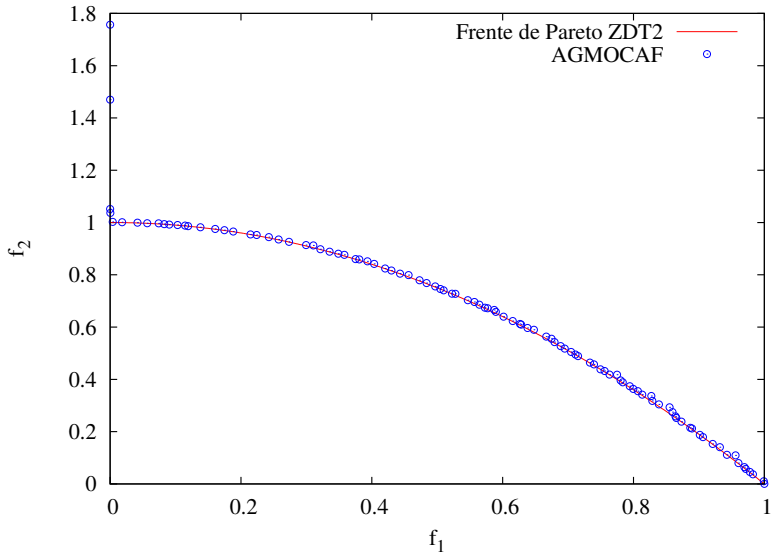

Figura B.2: Gráfica de convergencia del algoritmo AGMOCAF en la función ZDT2 (utilizando 500 evaluaciones de la función objetivo).

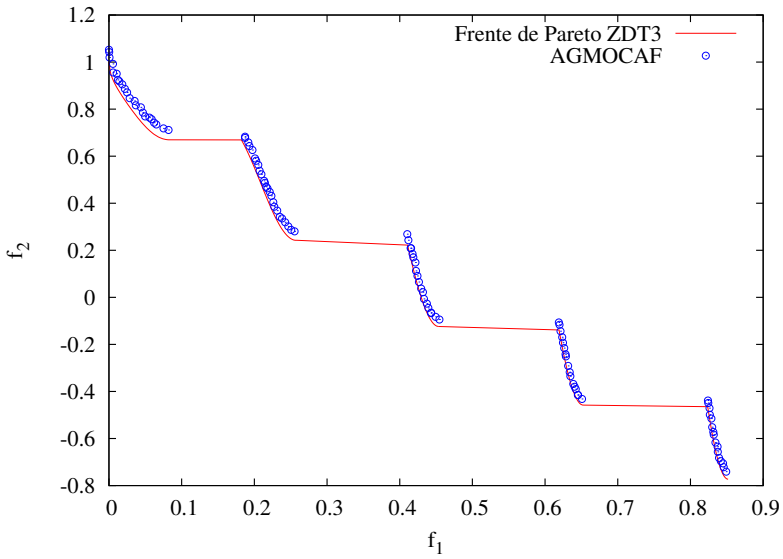

Figura B.3: Gráfica de convergencia del algoritmo AGMOCAF en la función ZDT3 (utilizando 10,000 evaluaciones de la función objetivo).

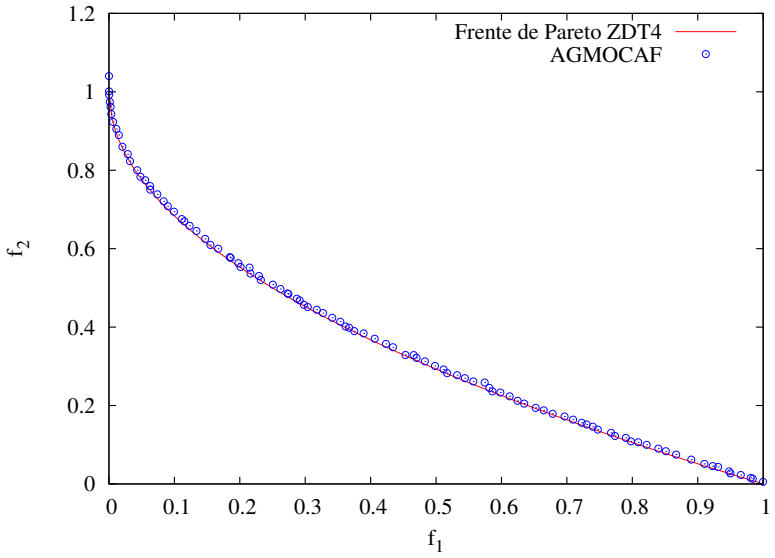

Figura B.4: Gráfica de convergencia del algoritmo AGMOCAF en la función ZDT4 (utilizando 25,000 evaluaciones de la función objetivo).

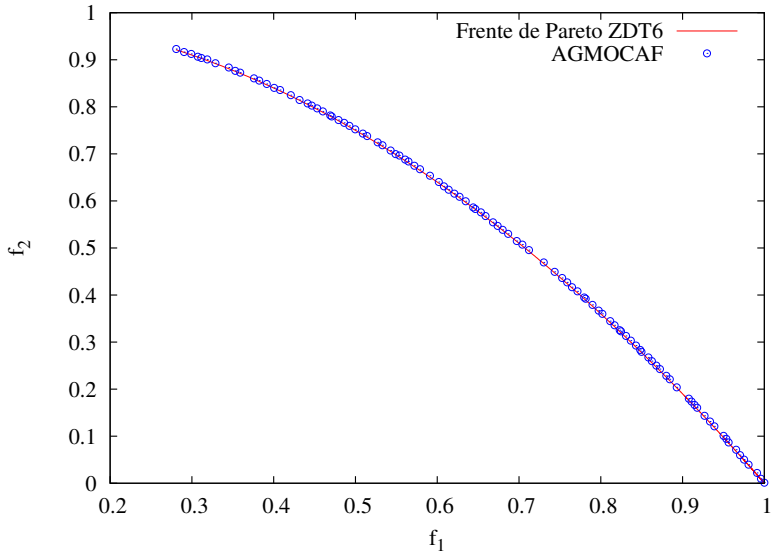

Figura B.5: Gráfica de convergencia del algoritmo AGMOCAF en la función ZDT6 (utilizando 8,000 evaluaciones de la función objetivo).

BIBLIOGRAFÍA

[1] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, New York, 1973.

[2] J. D. Bagley. *The Behavior of Adaptive Systems Which Employ Genetic and correlation Algorithms*. PhD thesis, Univ. Michigan, Ann Arbor, 1967.

[3] Pierre Baldi. Computing with arrays of bell-shaped and sigmoid functions. In *NIPS-3: Proceedings of the 1990 conference on Advances in neural information processing systems 3*, pages 735–742, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[4] H. J. Bremerman. *Optimization Through Evolution and Recombination in self-organizing Systems*, pages 93 – 106. Spartan Books, Washington DC, 1962.

[5] D. S. Broomhead and D. Lowe. Multivariate functional interpolation and adaptative networks. *Complex Systems*, 2:321–355, 1988.

[6] G. Toscano Pulido C. A. Coello Coello. Multiobjective optimization using a micro-genetic algorithm. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H-M Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 274–282, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[7] M. Casdagli. Nonlinear prediction of chaotic time series. *Phyzca*, 35(D), 1989.

[8] Carlos A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):129–156, 1999.

[9] Carlos A. Coello Coello and Gregorio Toscano Pulido. A micro-genetic algorithm for multiobjective optimization. In *EMO '01: Proceedings of the First Interna-*

tional Conference on Evolutionary Multi-Criterion Optimization, pages 126–140, London, UK, 2001. Springer-Verlag.

[10] Carlos A. Coello Coello and Carlos E. Mariano Romero. *Multiple Criteria Optimization. State of the Art Annotated Bibliographic Survey*, volume 52, chapter Evolutionary Algorithms and Multiple Objective Optimization, pages 277–331. Kluwer Academic Publishers, June 2002.

[11] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[12] Carlos A. Coello Coello. Treating Constraints as Objectives for Single-Objective Evolutionary Optimization. *Engineering Optimization*, 32(3):275–308, 2000.

[13] David W. Corne, Nick R. Jerram, Joshua D. Knowles, and Martin J. Oates. PESA-II: Region-based selection in evolutionary multiobjective optimization. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 283–290, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann.

[14] 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, J. J. 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.

[15] Noel A. C. Cressie. *Statistics for Spatial Data*. John Wiley & Sons, New York, 1993.

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

[17] K. Deb and M. Goyal. A combined genetic adaptive search geneas for engineering design, 1996.

[18] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. Interscience series in systems and optimization. John Wiley & Sons, LTD, 2001.

[19] Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1995.

[20] 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, J. J. 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.

[21] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[22] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Willey & Sons, New Yotk, 1973.

[23] Francis Ysidro Edgeworth. *Mathematical Psychics*. P. Keagan, London, England, 1881.

[24] M Erickson, A. Mayer, and J. Horn. The Niched Pareto Genetic Algorithm 2 applied to the design of groundwater remediation systems. In E. Zitzler, K. Deb, L. Thiele, and C. A. Coello Coello, editors, *First international conference on evolutuinary Multi-Criterion optimization*, pages 681–695. Lecture Notes in Computer Science, Spring Verlag, 2001.

[25] David B. Fogel. *Evolutionary computation, toward a new philosophy of machine intelligence*. IEEE press, 2000.

[26] L. J. Fogel. Autonomous automata. *Industrial Research*, 4:14 –19, 1962.

[27] L. J. Fogel. Decision-making by automata. Technical Report GDA-ERR-AN-222, General Dynamics, San Diego, 1962.

[28] L. J. Fogel. Toward inductive inference automata. In *Proc. of the Int. Federation for Information Processing Congress*, pages 395 – 400, Munich, 1962.

[29] L. J. Fogel, A. J. Owens, and M. J. Walsh. On the evolution of artificial intelligence. In *Proc. of the 5th National Symp. on Human Factors in Electronics*, pages 63 – 76, San Diego, 1964. IEEE.

[30] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence Through a Simulation of Evolution*. Spartan Books, Washington DC, 1965.

[31] L. J. Fogel, A. J. Owens, and M. J. Walsh. Intelligent decision-making through a simulation of evolution. *IEEE trans. of the professional Technical Group on Human Factors in Electronics*, 6(1), 1965.

[32] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Genetic Algorithms:*

Proceedings of the Fifth International Conference, pages 416–423. Morgan Kaufmann, 1993.

[33] A. S. Fraser. Simulation of genetic systems by automatic digital computers. *Australian J. of Biol. Sci.*, 10:484 – 491, 1957.

[34] L. S. Gandin. *Objective analysis of meteorological fields*. Gidrometeorologicheskoe Izdaltel'stvo, Leningrad, 1963. Translated by Israel Program for Scientific Translations, Jerusalem, 1965.

[35] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1989.

[36] S. J. Hanson and D. J. Burr. Knowledge representation in connectionist networks. Technical report, Bell Communications Research, Livingston, N.Y., 1987.

[37] Eric Hartman, James D. Keeler, and Jacek M. Kowalski. Layered neural networks with gaussian hidden units as universal approximations. *Neural Comput.*, 2(2):210–215, 1990.

[38] Mohamad H. Hassoun. *Fundamentals of Artificial Neural Networks*. The MIT Press, Cambridge, Massachussets, 1995.

[39] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pages 82–87, Piscataway, New Jersey, 1994. IEEE Service Center.

[40] Y. Jin, T. Okabe, and B. Sendhoff. Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how? In L. In Spector, E. D. Goodman, A. Wu, W. B. Langdon, H. M. Voight, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 1042 – 1049, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[41] Andre G. Journel and C. J. Huijbregts. *Mining Geostatistics*. Academic Press, London, 1978.

[42] Joshua D. Knowles and David Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[43] A. N. Kolmogorov. Interpolation and extrapolation of stationary random sequences. *Izvestiia akademii Nauk SSSR, seriia matematicheskiia*, 5(1):3–14, 1941. Translation (1962), memo RM-3090-PR, Rand Corp. Santa Monica, CA.

[44] Danie G. Krige. A statistical approach to some basic mine valuations problemas on the Witwatersrand. *Journal of the Chemical, Metallurgical and mining society of South Africa*, 52(6):119–139, 1951.

[45] A. S. Lapedes and R. Farber. Nonlinear signal processing using neural networks: Prediction and system modeling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, New Mexico, 1987.

[46] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematics, Statics and Probability*, pages 281–297. University of California Press, Berkeley, 1967.

[47] G. Matheron. Principles of geostatistics. *Economic Geology*, 58(8):1246–1266, 1963.

[48] G. Matheron. The theory of regionalized variables and its applications. *Cahiers du centre de morphologie mathematique*, (5), 1971.

[49] Georges Matheron. *Traité de géostatistique appliquée*, volume 1 of *Memoires du Bureau de Recherches Geologiques et Minieres*. Paris, technip edition, 1962.

[50] Georges Matheron. Le krigeage universel. *Cahiers du centre de morphologie mathematique*, 1(1), 1969. Fontainebleau, France.

[51] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

[52] Charles A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2(1):11–12, 1986.

[53] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[54] J. Moody and C. Darken. Learning with localized receptive fields. In D. Touretzky, G. Hinton, and T. Sejnowsky, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 133–143, San Mateo, California, 1989. Morgan Kaufmann.

[55] J. N. Morse. Reducing the size of the nondominated set: Pruning by clustering. *Computers and Operations Research*, 7(1-2):55–66, 1980.

[56] Pawan K. S. Nain and Kalyanmoy Deb. A multi-objective optimization procedure with successive approximate models. Technical Report 2005002, Kanpur Genetic Algorithms Laboratory, March 2005.

[57] M. Niranjan and F. Fallside. Neural networks and radial basis functions in classifying static speech patterns. Technical Report CUEDIF-INFENG17R22, Engineering Department, Cambridge University, 1988.

[58] C. K. Oei, D. E. Goldberg, and S-J Chang. Tournament selection, niching, and the preservation of diversity. Technical report 91011, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, Illinois, December 1991.

[59] A. Osyczka. *Multicriteria optimization for engineering design*, chapter Design optimization, pages 193 – 227. Academic Press, Cambridge, 1985.

[60] V. Pareto. *Cours d'economie politique*, volume 1 & 2. Lausanne, 1896.

[61] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246 – 257, 1991.

[62] E. Parzen. On the estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[63] T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer perceptrons. *Science*, 247:978–982, 1990.

[64] Tomaso Poggio and Federico Girosi. A theory of networks for approximation and learning. Technical Report AIM-1140, 1989.

[65] Tomaso Poggio and Federico Girosi. A theory of networks for approximation and learning. Technical Report AIM-1140, MIT, Cambridge, Mass., 1989.

[66] M. J. D. Powell. *Algorithms for the Approximation of Functions and Data*, chapter Radial basis functions for multivariate interpolation: A review. Clarendon Press, 1987.

[67] I. Rechenberg. *Cybernetic Solution Path of an Experimental Problem*. Royal Aircraft Establishment, August 1965.

[68] R. Rosenberg. *Simulation of Genetic Populations with Biochemical Properties*. PhD thesis, Univ. Michigan, Ann Arbor, 1967.

[69] J. David Schaffer. *Multiple objective optimization with vector evaluated genetic algorithms*. PhD thesis, Vanderbilt University, Nashville TN, 1984.

[70] H. P. Schwefel. *Kybernetische Evolution als Strategie der Experimentellen Forschung in der Stromungstechnik*. PhD thesis, Technical University of Berlin, 1965.

[71] H. P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley, Chichester, U.K., 1981.

[72] Donald F. Specht. Probabilistic neural networks. *Neural Netw.*, 3(1):109–118, 1990.

[73] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[74] J. F. Steffensen. *Interpolation*. Chelsea Publishing Company, New York, second edition, 1950.

[75] Lee Sukhan and R. M. Kil. Multilayer feedforward potential function network. In *IEEE International Conference on Neural Networks*, pages 161–171, San Diego, CA, 1988.

[76] K. C. Tan, E. F. Khor, and T. H. Lee. *Multiobjective Evolutionary algorithms and applications*. Advanced information and Knowledge processing. Springer Verlag, 2005.

[77] David A. Van Veldhizen and Gary B. Lamont. On measuring multiobjective evolutionary algorithm performance. *2000 Congress on Evolutionary Computation*, 1:204–211, July 2000.

[78] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.

[79] Dietrich Wettschereck and Thomas G. Dietterich. Improving the performance of Radial Basis Function Networks by Learning Center Locations. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Neural Information Processing Systems 4*, pages 1133–1140. Morgan Kaufmann, San Mateo, CA, Dec. 1992.

[80] Norbert Wienner. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. MIT Press, Cambridge, MA., 1949.

[81] Herman Wold. A study in the analysis of stationary time series. *The journal of political economy*, 48(1):119–120, February 1940.

[82] Xin Yao. *Evolutionary computation, theory and applications*. World Scientific, 1999.

[83] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer engineering and networks laboratory, Swiss Federal Institute of Technology, Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, may 2001.

[84] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case studyand the strength pareto approach. *IEEE transactions on evolutionary computation*, 3(4):257–271, November 1999.

[85] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 8(2):173–195, 2000.

[86] Eckart Zitzler and Lothar Thiele. Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In A. E. Eiben, editor, *Parallel Problem Solving from Nature V*, pages 292–301, Amsterdam, September 1998. Springer-Verlag.