Desarrollo de algoritmo evolutivo multi-objetivo auto-organizado para la optimización de Pareto

Oscar David Salamanca Gómez

Asesor
Andrés Leonardo González Mancera, PhD.

Trabajo de tesis presentado para el grado de
Maestría en ingeniería mecánica

Departamento de ingeniería mecánica
Universidad de Los Andes
Bogotá, Colombia
Julio 2019

Desarrollo de algoritmo evolutivo multi-objetivo auto-organizado para la optimización de Pareto

Abstract

Los problemas de diseño en ingeniería suelen ser por naturaleza problemas multi-objetivo. Gracias a esto, no existe una única solución que optimice todos los requerimientos de diseño, sino un conjunto de soluciones óptimas llamado *frente de Pareto*. Bajo condiciones leves, este frente forma un espacio topológico de $(m-1)$ dimensiones para un problema multi-objetivo de m número de objetivos. Bajo esta propiedad, en el siguiente trabajo se propone un algoritmo evolutivo multi-objetivo basado en mapas auto-organizados para la obtención de frentes de Pareto en problemas de ingeniería. Utilizando un mapa auto-organizado con $(m-1)$ variables latentes, se encuentran las relaciones de vecindad actuales entre las soluciones de una población por medio de proceso de agrupamiento. De esta manera, una solución solo puede cruzarse con soluciones vecinas para la generación de una nueva solución.

Diferentes instancias de prueba son aplicadas a dicho algoritmo y comparado contra los mejores algoritmos de la actualidad. Los resultados muestran ventajas y desventajas respecto a otras aproximaciones. Adicionalmente, el algoritmo es probado en la exploración del espacio de diseño en un problema de optimización aerodinámico mostrando buenos resultados.

Para Blanca

Agradecimientos

Gracias a mis padres Yolanda y Oscar, mis hermanas Sophia y Laura, y a mi abuela Lidia, por su amor y apoyo incondicional. Su guía, paciencia y sacrificios me han traído hasta aquí. Gracias por creer en mí.

A mi asesor Andrés Gonzalez Mancera Ph.D., por su guía, apoyo y consejo. Sin su visión, entendimiento y experiencia no hubiese sido posible este proyecto.

A Valentina Garzón Parra, tu generosidad, apoyo, y constante amor fueron fundamentales para terminar este proceso. Te estaré eternamente agradecido.

A mis amigos Amalia Reyes, Paola Londoño, Juan Felipe Céspedes, Juan Camilo Osorio y Juan Diego Silva. Su amistad, compañía y apoyo hicieron de este camino uno que no me arrepentiré nunca de haber cruzado. Me queda una familia para toda la vida.

A Blanca, por su incansable generosidad y gran corazón. Gracias por estar siempre con mi familia.

A quien me impulsó a tomar este camino, siempre sin dejar de creer en mí, gracias.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Optimización es el proceso de encontrar y comparar soluciones factibles hasta no poder encontrar solución que sea mejor. Estas soluciones se consideran "buenas" o "malas" en términos de un objetivo establecido. En ingeniería, ejemplos de estos objetivos pueden ser costo de fabricación, peso, eficiencia, resistencia, entre otros factores.

De manera frecuente, los problemas de diseño en ingeniería son multi-objetivos por naturaleza, dado que son más de uno los objetivos de diseño que deben ser optimizados. Estos objetivos suelen imponer requerimientos potencialmente conflictivos en el desempeño, tanto económico como técnico, de un sistema dado [1]. Es aquí cuando el diseñador debe formular un problema de optimización con estos objetivos conflictivos con el fin de explorar las opciones de diseño realizables.

Dado que para objetivos conflictivos no existe la posibilidad de encontrar una única solución óptima que satisfaga todos los objetivos, el resultado de una optimización multi-objetivo es encontrar los compromisos óptimos que pueden ser aceptados entre estos objetivos. Sin embargo, múltiples compromisos pueden ser realizados, siendo cada uno de ellos óptimo de alguna manera. El conjunto formado por estas soluciones óptimas es llamado el frente de Pareto, donde mejorar un objetivo implica la desmejora de al menos un objetivo distinto; esto gracias a la naturaleza conflictiva de los objetivos. De este conjunto queda la decisión de tomar el mejor compromiso según el criterio del diseñador [2].

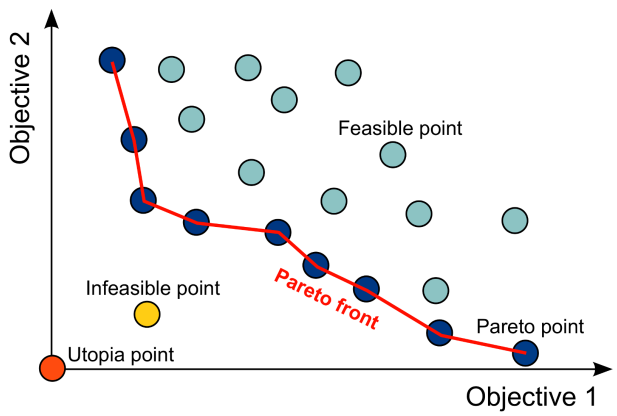


Figura 1.1: Visualización de un frente de Pareto [3]

Una herramienta estándar utilizada en optimizaciones de Pareto (i.e. encontrar el frente de

Pareto a un problema dado) son los *Algoritmos evolutivos* (AE). Su propiedad más atractiva es la búsqueda basada en poblaciones, la cual permite generar un frente de Pareto en una sola ejecución. Estos algoritmos tratan de imitar la evolución Darwiniana de tal manera que una población de soluciones gradualmente se adapta a lo largo de múltiples generaciones, alterando las variables de decisión hasta alcanzar un objetivo. Cada generación de la población es sujeta a procesos inspirados en los mecanismos de la evolución tales como *selección*, *mutación* y *cruce*. Estos procesos son llamados *operadores* y son la materia principal en investigación en este tema.

En optimizaciones de Pareto, especialmente con el uso de algoritmos evolutivos multi-objetivo, los mayores avances se han evidenciado en los operadores de selección y técnicas de selección de funciones objetivo. Algoritmos como el NSGA-II [4] y SPEA2 [5] son resultados de esto y ofrecen estrategias de selección de los mejores individuos en cada población de manera eficiente. Sin embargo, los operadores de mutación y recombinación usados en algoritmos multi-objetivo utilizan los mismos que en sus semejantes de objetivo único.

Gracias a las técnicas utilizadas tales como la Estrategia de Evolución con Matriz de Adaptación de Covarianza (CMA-ES) [6], la cual almacena en una matriz de covarianza el camino de mutaciones exitosas, el operador puede explorar el conocimiento previo para adaptarse y converger de manera acelerada. Esto hace que los algoritmos adaptativos tengan mejor desempeño que algoritmos que no lo son. Por otra parte, el camino en una optimización multi-objetivo no está definido de manera clara. Debido a que la población converge a diferentes lugares para obtener el frente de Pareto, esto no puede ser descrito por una sola matriz de covarianza.

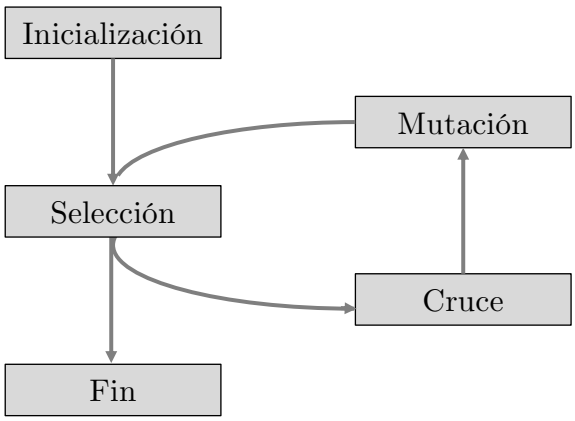


Figura 1.2: Secuencia de un algoritmo evolutivo

Una gran variedad de métodos de aprendizaje tanto autónomo como estadístico ha sido propuesta para la extracción de patrones de un conjunto de datos [7]. La motivación del algoritmo propuesto aquí radica en el uso de un mapa auto-organizado para extraer información de la relación entre individuos de una población y utilizarla para la generación de nuevas soluciones. Inspirados por los trabajos desarrollados por Milano *et al.* [8], Buche *et al.* [9] y Kan *et al.*[10], se introduce una variación al nuevo método de adaptación para el operador de mutación y recombinación utilizando mapas auto-organizados (SOMs) de Kohonen [11]. Este método, es entrenada con las mejores soluciones de cada generación y de esta manera, en un proceso de aprendizaje autónomo, sigue el camino de los mejores resultados de la evolución. Luego de un proceso de selección típico, el mapa (SOM) recombina las soluciones vecinas que mejor describen el espacio topológico de esta selección de soluciones,

asegurando que la descendencia de estas soluciones siga aportando al mapa. Este proceso obliga al algoritmo a centrarse en soluciones más prometedoras, acelerando la convergencia del frente de Pareto.

Se propone probar el algoritmo propuesto en un problema de diseño aerodinámico. El diseño y rendimiento eficiente de perfiles aerodinámicos comienza con la selección de un perfil alar. Este puede ser un diseño propio o escogido de una familia conocida de perfiles. Desde que el Comité Asesor Nacional para la Aeronáutica (NACA por sus siglas en inglés) en 1933 publicó el catálogo de 78 perfiles aerodinámicos en su reporte anual [12], los perfiles NACA han sido la referencia estándar a la hora de iniciar un diseño. Sin embargo, existen situaciones en las que la simple selección de un perfil no es suficiente, como lo es en el diseño de rotor de un helicóptero, turbomaquinaria o plantas de energía eólica.

Los fenómenos aerodinámicos que ocurren en un perfil alar son de gran importancia a la hora de diseñar uno. La sustentación, típica fuerza a maximizar en un problema de diseño, entra en conflicto con el arrastre, fuerza que comúnmente se desea minimizar. Este escenario es clásico pero problemático de atacar desde un punto de vista computacional debido a la incapacidad de predecir correctamente el arrastre mediante simulaciones, sin embargo, los avances en métodos computacionales hacen que estos cálculos sean cada vez más confiables.

Además, los problemas de optimización en perfiles aerodinámicos son fuertemente dependientes del método de parametrización utilizado para la generación de curvas que forman el perfil, ya que este determina el número de variables de diseño a trabajar. En este trabajo se explora el espacio de diseño dado el método de parametrización de splines de Ferguson a un número de Reynolds fijo. El solucionador utilizado fue XFOIL, desarrollado por Mark Drela del MIT [13].

Este documento está organizado de la siguiente manera. Primero una gran revisión de literatura es presentada. Esta incluye los conceptos básicos de optimización multi-objetivo, algoritmos evolutivos y mapas auto-organizados. Luego, el algoritmo propuesto es descrito junto a resultados experimentales del mismo en instancias de prueba tradicionales. Finalmente, el algoritmo es usado en un problema de optimización de perfiles aerodinámicos.

Objetivo general

Desarrollar un algoritmo de optimización multi-objetivo para problemas en ingeniería utilizando mapas auto-organizados.

Objetivos específicos

1. Proponer el uso de mapas auto-organizados (SOMs) como paso intermedio recombinador en un algoritmo genético.
2. Evaluar el algoritmo propuesto respecto a los algoritmos utilizados en la literatura.
3. Evaluar el algoritmo propuesto en un problema de interés en diseño en ingeniería: diseño óptimo de un perfil aerodinámico dado un método de parametrización.

Capítulo 2

Optimización multi-objetivo

Como su nombre lo indica, los problemas de optimización multi-objetivo (MOOP, por sus siglas en inglés) tienen más de una función objetivo a optimizar. La mayoría de problemas prácticos de toma de decisiones tienen múltiples objetivos o criterios evidentes. La falta de metodologías de resolución de MOOP han hecho que estos sean resueltos como problemas mono-objetivo. Sin embargo, existen diferencias fundamentales entre un problema mono y multi-objetivo. En problemas mono-objetivo se busca una solución que optimice una única función. De manera errónea se puede pensar que en una optimización multi-objetivo se busca una solución óptima para cada una de las funciones objetivo. A continuación se presentan los principios para que una solución sea óptima ante la presencia de múltiples objetivos.

2.1. Problemas de optimización multi-objetivo

Un problema de optimización multi-objetivo tiene un número de funciones que deben ser minimizadas o maximizadas. Al igual que problemas de optimización mono-objetivo, existen restricciones que cualquier solución factible (incluyendo la solución óptima) debe cumplir. A continuación se muestra la forma general de un MOOP:

$$\left.\begin{array}{rl} \text{Minimizar/Maximizar} & \mathbf{F}(\mathbf{x}) = \left(f_1(\mathbf{x}), \cdots, f_m(\mathbf{x})\right) \\ \text{Sujeto a} & \mathbf{G}(\mathbf{x}) = \left(g_1(\mathbf{x}), \cdots, g_j(\mathbf{x})\right) \geq 0 \\ & \mathbf{H}(\mathbf{x}) = \left(h_1(\mathbf{x}), \cdots, h_k(\mathbf{x})\right) = 0 \\ & \mathbf{x} = (x_1, \cdots, x_n) \in \Omega \end{array}\right\} \tag{2.1}$$

donde $\mathbf{x} = (x_1, \cdots, x_n)$ es el vector de variables de decisión, $\Omega = [a_i, b_i]^n$ es el espacio de variables de decisión o espacio de búsqueda, $\mathbf{F} : \Omega \to \mathbb{R}^m$ consiste en m número de funciones objetivos $f_i(\mathbf{x})$, $i = 1, \cdots, m$, y $\mathbb{R}^n$ denota el espacio objetivo. Cuando en esta tesis nos referimos a una *solución*, esta es un vector $\mathbf{x}$.

Asociado al problema se encuentran k y j número de restricciones como igualdades y desigualdades respectivamente, donde los términos $h_k(x)$ y $g_j(x)$ son las funciones de restricción. Las restricciones de desigualdad se enuncian de tipo "mayor que-igual a", sin embargo si se tienen desigualdades de tipo "menor que-igual a"pueden ser multiplicadas por -1. Una solución $\mathbf{x}$ que no satisfaga las $k + j$ restricciones es una solución no factible. Por otro lado, si se tiene una solución $\mathbf{x}$ que satisface todas las restricciones y se encuentra en el rango de

soluciones dadas por Ω, esta será una solución factible. La región factible está compuesta por el conjunto de soluciones factibles.

Teniendo en cuenta la notación utilizada en la formulación del MOOP, se tienen m funciones objetivo, las cuales pueden ser maximizadas o minimizadas. Por el principio de dualidad es posible simplificar problemas con funciones objetivo mixtas al desarrollar algoritmos que transformen estas funciones para tener un conjunto homogéneo para optimizar. Es decir, si se tienen funciones objetivo mixtas y se desea plantear un problema de maximización, es posible transformar aquellas funciones objetivo de minimización a maximización.

La principal diferencia entre los problemas de optimización mono-objetivo y los MOOP es que las funciones objetivo de estos últimos constituyen un espacio multi-dimensional, el cual es llamado espacio objetivo, $\mathbf{Z}$. Por lo tanto, para cada solución $\mathbf{x}$ existe un punto en el espacio objetivo, denotado $\mathbf{F}(\mathbf{x}) = z = (z_1, \cdots, z_m)^T$.

2.1.1. MOOPs lineales y no lineales

Cuando todas las funciones objetivo y las funciones de restricción son lineales, se tiene un problema multi-objetivo lineal o MOLP por sus siglas en inglés. Por otro lado, si alguna de las funciones objetivo o restricciones no son lineales, se tiene un problema multi-objetivo no lineal. Sin embargo, según [14] no se tienen pruebas de convergencia de las técnicas de solución de problemas multi-objetivo no lineales.

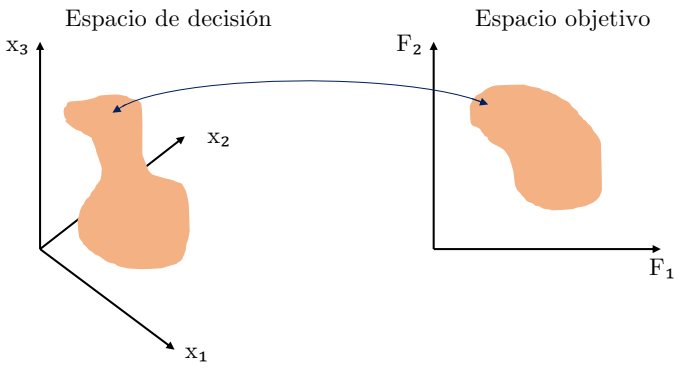


Figura 2.1: Representación del espacio de la variable de decisión y el correspondiente espacio objetivo

2.1.2. MOOPs convexos y no convexos

Antes de discutir la convexidad de los MOOPs se definirá una función convexa. Una función $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}$ es convexa si para todo par de soluciones $\mathbf{x}^{(1)}$ y $\mathbf{x}^{(2)} \in \mathbb{R}$ se cumplen la siguiente condición:

$$f\left(\alpha\mathbf{x}^{(1)} + (1-\alpha)\mathbf{x}^{(2)}\right) \leq \alpha f(\mathbf{x}^{(1)}) + (1-\alpha)f(\mathbf{x}^{(2)}), \text{ para todo } 0 \leq \alpha \leq 1 \tag{2.2}$$

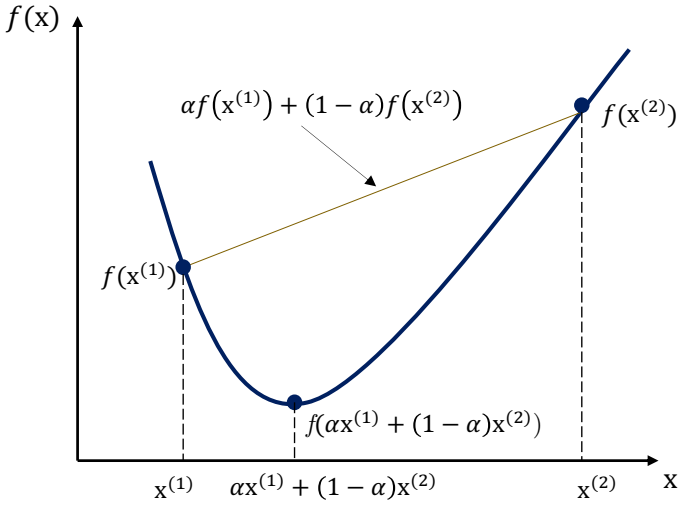


Figura 2.2: Ilustración de una función convexa

De la definición anterior se pueden inferir las siguientes propiedades para una función convexa:

1. La matriz Hessiana de $f(\mathbf{x})$ es positiva para todo $\mathbf{x}$.
2. Para una función convexa, un mínimo local siempre es un mínimo global.
3. La aproximación lineal de $f(\mathbf{x})$ para cualquier punto en el intervalo $[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}]$ siempre subestima el valor real de la función.

Por otro lado, según [14] un MOOP es convexo si todas las funciones objetivo y la región factible son convexas. Teniendo en cuenta esta definición, es posible afirmar que un MOLP es un problema convexo. Debido a que un MOOP tiene dos espacios, la convexidad en cada uno de ellos es importante en los algoritmos multi-objetivos.

2.2. Principios de la optimización multi-objetivo

Para ilustrar los principios de la optimización multi-objetivo se presentará el problema de las rutas de una aerolínea. Es de conocimiento de todos las escalas que las aerolíneas nos obligan a incluir en los itinerarios al recorrer grandes distancias. En la figura 2.3 se muestra un mapa con rutas hipotéticas de una aerolínea en Colombia. Es posible evidenciar que esta aerolínea tiene dos ciudades en las cuales se centra su actividad (Bogotá y Cartagena). Es de suponer que debido a esto la aerolínea cuente con múltiples vuelos entre estas dos ciudades durante el día. Si un viajero decide ir de Bogotá a Cartagena, cuenta con un vuelo directo. Sin embargo, si se quisiera transportar entre Bogotá y Medellín seria necesario que tome un vuelo Bogotá-Cartagena y otro Cartagena-Medellín. En este caso, la distancia que tendrá que recorrer es mayor a la distancia geográfica entre Bogotá y Medellín. Para la aerolínea tener este tipo de red de rutas es más fácil de coordinar y mantener. De esta manera podrá generar un gran ahorro debido a que no debe contar con empleados en todas las ciudades, lo que les permitiría tener mejores instalaciones y mejor servicio al cliente en las ciudades principales (Bogotá y Cartagena).

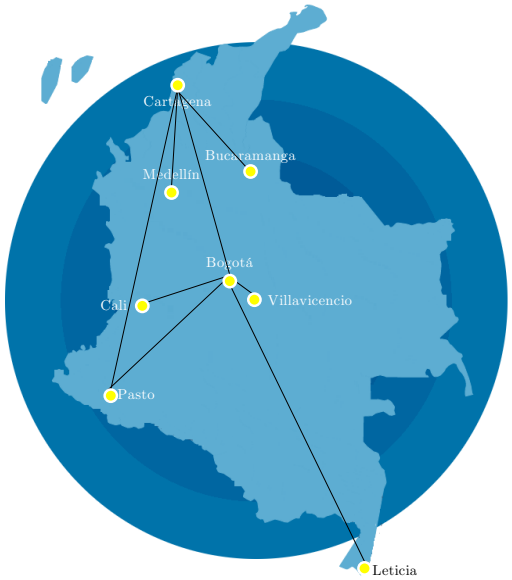


Figura 2.3: Red de rutas de una aerolínea que requiere de escalas

Por otro lado, para el viajero que debe recorrer grandes distancias esta solución no es conveniente debido a las incomodidades causadas. Sin embargo, esta situación no es del todo perjudicial para él debido a que posiblemente adquirió el tiquete de esta aerolínea más económico en comparación con otras. La reducción de costos de la aerolínea antes mencionada le permitió vender sus tiquetes más económicos. Si se quisiera considerar únicamente la comodidad del pasajero, se debería tener una red de rutas como la ejemplificada en la figura 2.4. Esta red hipotética de rutas se cuenta con vuelos directos entre todas las ciudades, por lo que no se requerirán escalas. Debido a que los costos de operación de la aerolínea serían muy altos, los costos de los tiquetes también lo serán.

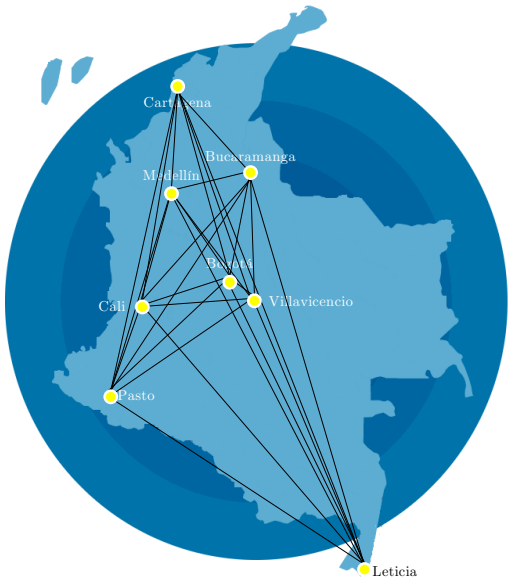


Figura 2.4: Red de rutas hipotética de una aerolínea que no requiere de escalas

De esta manera se evidencia una "negociación" entre los dos objetivos de este problema (costo versus comodidad). Un vuelo más económico implica más escalas causando más incomodidades al pasajero. Mientras que un vuelo más cómodo, con ruta directa, es más costoso.

Los dos mapas expuestos representan soluciones extremas para el problema de optimización expuesto. En algunos casos las aerolíneas hacen esfuerzos por crear más rutas entre las ciudades y vender los tiquetes a bajos precios. Teniendo en cuenta este ejemplo es posible afirmar que no existen soluciones óptimas únicas que optimicen ambos objetivos al tiempo. El resultado obtenido de los MOOPs será un conjunto de soluciones óptimas.

2.2.1. Objetivos en optimizaciones multi-objetivo

En lugar de considerar todo el espacio de diseño en un MOOP para encontrar las soluciones que pertenecen al frente de Pareto y las que no, se puede considerar este espacio dividido bajo el criterio de dominancia. Basados en una comparación por pares, este espacio puede ser dividido en dos conjuntos, $\mathbf{P}_1$ y $\mathbf{P}_2$, que no se sobreponen entre sí, de tal manera que el conjunto $\mathbf{P}_1$ contiene todas las soluciones que no se dominan entre sí y, a su vez, cada una de estas soluciones domina por lo menos una solución del conjunto $\mathbf{P}_2$. El conjunto $\mathbf{P}_1$, llamado *conjunto no dominado*, contendrá las soluciones óptimas, mientras que el conjunto $\mathbf{P}_2$ contendrá soluciones no óptimas. En el caso de problemas con objetivos contradictorios, el conjunto de soluciones óptimas contiene más de una solución. Si se tienen múltiples soluciones óptimas es difícil seleccionar una sin tener más información del problema. En el caso de no contar con mayor información todas las soluciones del frente de Pareto tienen la misma importancia. Idealmente es de interés encontrar la mayor cantidad de soluciones óptimas de Pareto posibles en cada problema. Los objetivos de los MOOPs son:

1. Encontrar un conjunto de soluciones lo más cercano posible el frente de Pareto óptimo.
2. Encontrar un conjunto de soluciones lo mas diverso posible.

El primer objetivo es el más importante en todo problema de optimización. No es deseable que el problema converja a un conjunto de soluciones lejano al conjunto verdadero de soluciones óptimas. Únicamente cuando las soluciones convergen a valores cercanos a las soluciones óptimas reales se pueden asegurar las propiedades del óptimo. Este objetivo se presenta tanto en problemas de optimización mono-objetivo como en MOOPs.

El segundo objetivo es específico para los MOOPs. Las soluciones además de ser cercanas al frente óptimo de Pareto deben estar distribuidas en la región óptima de Pareto. Solo si se tiene un conjunto de soluciones diverso se puede asegurar unas soluciones con buenas "negociaciones" o balances entre los objetivos.

2.3. Concepto de dominancia

Comúnmente los algoritmos de problemas de optimización multi-objetivo comparan dos soluciones teniendo en cuenta la dominancia de una sobre las otras.

Suponiendo que se tienen N funciones objetivo. Se dice que la solución $\mathbf{x}^{(1)}$ domina a la solución $\mathbf{x}^{(2)}$ si se cumplen las siguientes condiciones:

1. La solución $\mathbf{x}^{(1)}$ es no peor que la solución $\mathbf{x}^{(2)}$ para todo N.
2. La solución $\mathbf{x}^{(1)}$ es estrictamente mejor que la solución $\mathbf{x}^{(2)}$ en al menos uno de los N objetivos.

Teniendo en cuenta esta definición de dominancia se discutirán las propiedades de este ope-

rador de relación.

- Reflexivo: las relaciones de dominancia son *no reflexivas* debido a que ninguna solución s domina sobre ella misma. Luego, no se satisfacería la segunda condición.
- Simétrico: las relaciones de dominancia son *no simétricas* debido a que $s \preceq r$ no implica $r \preceq s$. Sin embargo, si s domina a r, implica que r no domina a s.
- Antisimétrico: debido a que las relaciones de dominancia no pueden ser siméticas, tampoco pueden ser antisimétricas.
- Transitivo: las relaciones de dominancia son *transitivas* debido a que si $s \preceq r$ y $r \preceq t$, luego $s \preceq t$.

2.4. Eficiencia de Pareto

En un problema de optimización, se tiene un conjunto de soluciones S. El conjunto de soluciones no dominadas S' son aquellas soluciones que no son dominadas por ningún componente de S. Cuando todo el espacio de búsqueda L pertenece al conjunto S, el resultado del conjunto no dominado S' es el conjunto óptimo de Pareto. En la figura 2.5 de color marrón se pueden evidenciar el conjunto óptimo de Pareto para problemas de optimización con diferentes combinaciones de dos tipos de objetivos. Si se quisiera minimizar la función f_1 y maximizar la función f_2, el conjunto óptimo de Pareto es la unión de dos regiones. En cualquier caso o combinación de objetivos, el conjunto óptimo de Pareto es el borde de la región factible del problema.

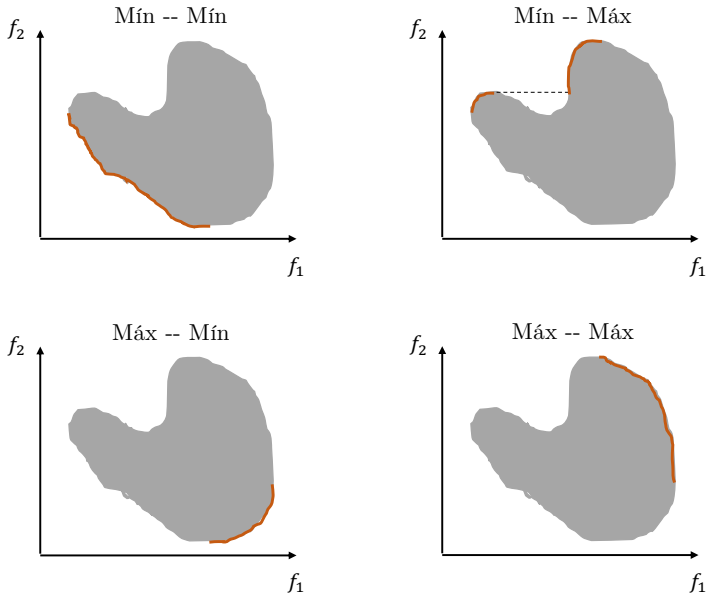


Figura 2.5: Soluciones óptimas de Pareto para diferentes combinaciones de dos tipos de objetivos

El conjunto no dominado del espacio de búsqueda L es el conjunto óptimo global de Pareto. Lo anterior debido a que las soluciones de este conjunto no son dominadas por otras.

Por otro lado, si para cada $\mathbf{x}$ del conjunto S no existe solución $\mathbf{y}$ (en el vecindario de $\mathbf{x}$ tal que $\|\mathbf{y} - \mathbf{x}\|_\infty \leq \epsilon$, donde ϵ es un número positivo pequeño) dominando cualquier componente de S, luego, las soluciones pertenecientes al conjunto S representan un conjunto local óptimo de Pareto.

Capítulo 3

Algoritmos evolutivos

Los algoritmos evolutivos (EAs, por sus siglas en inglés) simulan los principios de la evolución en procesos de optimización. Una de las mayores ventajas de este método es que se requiere poca información del problema para su resolución. Sólo se requiere de una función objetivo (fitness) para ser optimizada. Por lo tanto, los EAs pueden ser aplicados en problemas más complejos con discontinuidades, no diferenciables o con ruido.

3.1. Dificultades de algoritmos de optimización clásicos

Se llaman algoritmos clásicos a aquellos que actualizan una única solución en cada iteración y utilizan reglas de transición determinista para aproximarse a la solución óptima. Estos algoritmos inician con valores aleatorios y debido a las reglas de transición sugieren una dirección para la búsqueda. Por lo tanto, la búsqueda se lleva a cabo en un única dirección para encontrar la solución óptima. En la figura 3.1 se puede ver la región factible de un problema de optimización y el procedimiento antes descrito.

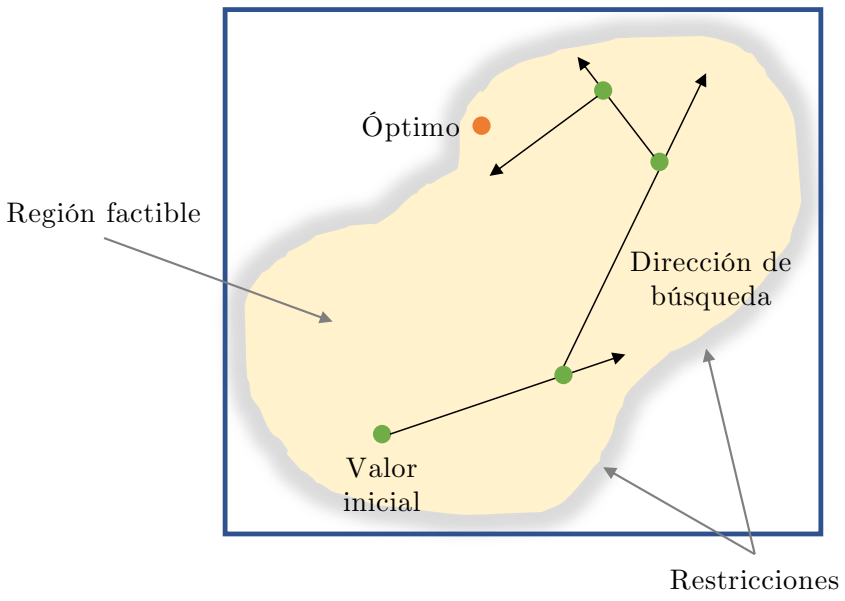


Figura 3.1: Aproximación punto a punto resuelto con algoritmo de optimización clásico

Estos algoritmos de optimización clásicos se pueden dividir en dos grupos. En primera intancia se tienen los métodos directos en los cuales la búsqueda es guiada teniendo en cuenta únicamente la función objetivo y las restricciones del problema. Por otro lado, se tienen los métodos basados en el gradiente en los que se utilizan la primera o segunda derivada de la función objetivo y las restricciones para guiar la búsqueda. Debido a esta diferencia, los métodos directos suelen necesitar de mayor número de iteraciones para converger a un resultado óptimo. Sin embargo, los métodos basados en el gradiente no suelen ser eficientes en problemas discontinuos o no diferenciables.

A pesar de estas diferencias, los métodos clásicos tienen dificultades comunes:

- Los algoritmos no son eficientes con problemas con espacios de búsqueda discretos. Esto debido a que las soluciones de las regiones no factibles son permitidas en los métodos clásicos de optimización, por lo tanto el algoritmo invierte tiempo en soluciones que no son las óptimas.
- Los algoritmos no son eficientes al usar máquinas paralelas. En la actualidad, la mayoría de los problemas requieren de simulaciones, las cuales son resueltas con métodos numéricos y presentan diferentes funciones objetivo y restricciones. Sin embargo, debido a que los métodos clásicos son evaluados punto a punto, esta ventaja computacional no puede ser aprovechada en su totalidad.
- La mayoría de los algoritmos suelen estancarse en soluciones sub-óptimas. Al resolver problemas no lineales bastante complejos se puede encontrar más de un óptimo en el espacio de búsqueda. Si en el proceso de búsqueda el algoritmo decide seleccionar un óptimo cuya función objetivo es menor que otro, es posible afirmar que se encontró un óptimo local y posiblemente no se encontrará el óptimo global del problema.
- Si un algoritmo es eficiente resolviendo un problema de optimización puede no serlo resolviendo otro problema. Cada algoritmo de optimización clásico es diseñado para resolver un tipo de problema específico. Por ejemplo, los algoritmos de gradiente conjugado o dirección conjugada son eficientes en la solución de problemas cuya función objetivo es cuadrática.
- La convergencia del método a una solución óptima depende del valor inicial seleccionado aleatoriamente.

3.2. Operador de selección

El objetivo principal de un operador de reproducción es eliminar las soluciones no factibles y duplicar las factibles de una población, pero siempre manteniendo el tamaño de la población constante. Para esto, en primera instancia se deben identificar las soluciones buenas de la población. Seguido de esto se realizan múltiples copias de las soluciones encontradas y finalmente se elimina la cantidad de soluciones malas necesarias para mantener constante la población.

Uno de los métodos existentes para llevar a cabo esta tarea es el de torneos. Para explicar este concepto se asumirá una población de 6 latas de diferente volumen (fenotipo) como se muestra en la figura 3.2, cada una de ella contiene una posible solución (genotipo).

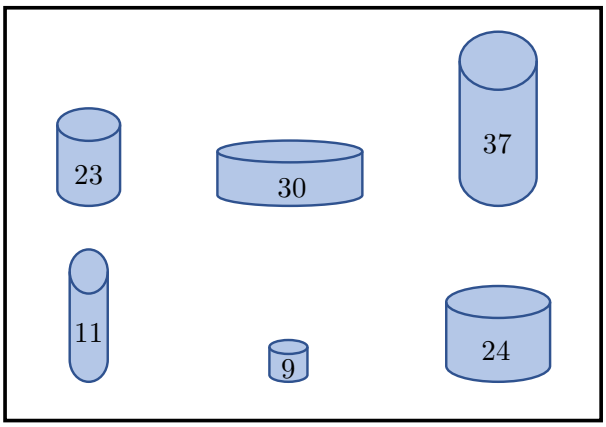


Figura 3.2: Población aleatoria de 6 latas

En este método se juegan torneos entre dos soluciones de la población, o en este caso dos latas. En cada torneo una solución ganará y esta se ubicará en el *mating pool* o población intermedia. Teniendo en cuenta esta metodología, cada lata podrá participar en dos torneos diferentes y la cantidad de veces que gane determinará la cantidad de veces que se debe reproducir. Es decir, si gana ambos torneos se podrá reproducir dos veces, pero si pierde ambos torneos será eliminada. De esta manera, cada solución tendrá dos, una o ninguna copia en la nueva población. Los torneos entre las 6 latas se pueden ver en la figura 3.3, donde se evidencia que las mejores soluciones tienen múltiples copias en el *mating pool* y las malas soluciones fueron descartadas.

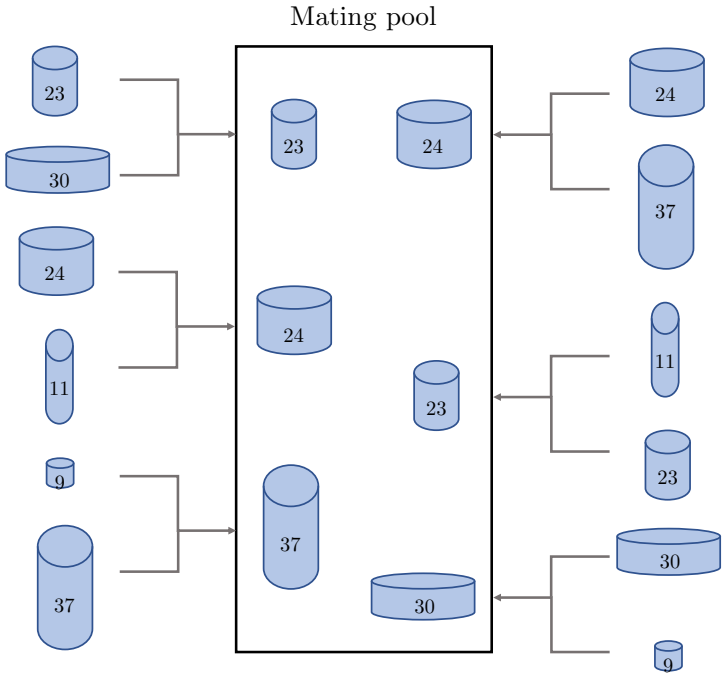


Figura 3.3: Torneos jugados entre los miembros de la población de latas.

Un aspecto importante de este operador de selección es que al cambiar la forma de comparación, se puede facilitar la maximización o minimización de los problemas. Cabe resaltar que el operador de reproducción no crea nuevas soluciones, tan solo hace copas de las mejores soluciones. La creación de nuevas soluciones le corresponde al operador de cruce y al operador de mutación.

3.2.1. Operador de cruce

Existe múltiples operadores de cruce, sin embargo, la mayoría de estos toman dos soluciones del *mating pool* y las cruzan en diferentes proporciones determinadas aleatoriamente. Es importante resaltar que el operador de cruce crea mejores soluciones en comparación con sus padres. Esto se debe a que los cruces no se realizan con cualquier solución, sino con aquellas que ganaron los torneos en la fase de selección. Si alguna solución creada no es buena, esta será eliminada en el siguiente proceso de reproducción, por lo tanto, tendrá una vida relativamente corta.

Al crear buenas soluciones a partir del cruce de otras, se tendrá una buena descendencia de la cual se generarán más copias en la siguiente etapa de reproducción y tendrá más posibilidades de cruzarse con otras buenas soluciones en las siguientes generaciones. Para preservar una buena descendencia, no todos los strings o soluciones del *mating pool* son utilizados en el cruce.

El operador de cruce básico es el de único punto. Para ilustrar el concepto nos referiremos a la figura 3.4, en la cual se tienen dos latas o soluciones del *mating pool*. Las latas tienen genotipos propios los cuales representan sus valores. Se selecciona de manera aleatoria una posición en del valor de su genotipo expresado en números binarios. Seguido de esto, se intercambian estos valores y se crean 2 nuevos hijos.

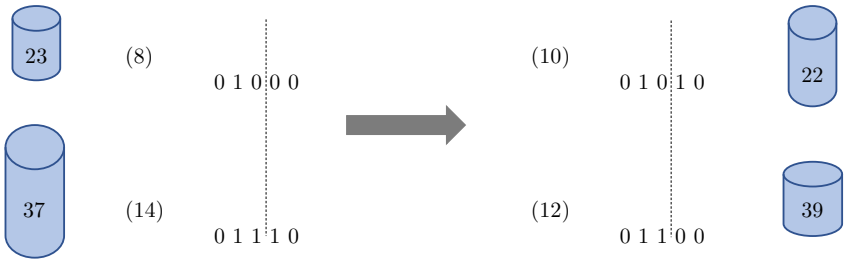


Figura 3.4: Operador de cruce de único punto

Cruce binario simulado

El cruce binario simulado funciona con dos padres, a partir de los cuales se crean dos descendientes. Este tipo de operador de cruce simula el principio de cruce de único punto explicado anteriormente, pero con binarios. A continuación se explicará el procedimiento para calcular la descendencia $x_i^{(1,t+1)}$ y $x_i^{(2,t+1)}$ a partir de los padres $x_i^{(1,t)}$ y $x_i^{(2,t)}$. Se define un factor de dispersión β_i como la razón entre la diferencia de los descendientes y la diferencia de los padres.

$$\beta_i = \left| \frac{x_i^{(2,t+1)} - x_i^{(1,t+1)}}{x_i^{(2,t)} - x_i^{(1,t)}} \right| \tag{3.1}$$

Para iniciar se crea aleatoriamente un número u_i entre 0 y 1. Seguido de esto, se encuentra un valor de β_{q_i} de tal manera que el área bajo la curva de probabilidad entre 0 y β_{q_i} es igual

a u_i. La distribución de probabilidad usada para la descendencia es derivada de la siguiente manera

$$\mathrm{P}(\beta_i) = \begin{cases} 0.5(\eta_c + 1)\beta_i^{\eta_c} & \text{si } \beta_i \leq 1 \\ 0.5(\eta_c + 1)\frac{1}{\beta_i^{\eta_c+2}} & \text{de lo contrario} \end{cases} \tag{3.2}$$

Un valor alto de η_c lleva a mayores probabilidades de crear descendencia cercana a los padres. Teniendo en cuenta la ecuación 3.2 es posible calcular una ecuación para describir el área bajo la curva de probabilidad

$$\beta_{q_i} = \begin{cases} (2u_i)^{\frac{1}{\eta_c+1}} & \text{si } u_i \leq 0.5 \\ \left(\frac{1}{2(1-u_i)}\right)^{\frac{1}{\eta_c+1}} & \text{de lo contrario} \end{cases} \tag{3.3}$$

Finalmente, las nuevas soluciones se calculan a partir de las ecuaciones 3.4 y 3.5.

$$x_i^{(1,t+1)} = 0.5\left[(1 + \beta_{q_i})x_i^{1,t} + (1 - \beta_{q_i})x_i^{2,t}\right] \tag{3.4}$$

$$x_i^{(2,t+1)} = 0.5\left[(1 - \beta_{q_i})x_i^{1,t} + (1 + \beta_{q_i})x_i^{2,t}\right] \tag{3.5}$$

Como se puede observar, la nueva descendencia es simétrica al rededor de los padres. De esta manera se evita que sea tenido en cuenta un único padre o se incline más hacia el valor de este.

El operador SBX se sesga a soluciones cercanas a los padres de manera más favorable que a soluciones alejadas de estos. Esencialmente, este operador tiene dos propiedades:

1. La diferencia entre descendencia es proporcional a la diferencia de los padres.
2. Soluciones cercanas a los padres son monótonamente más probables de ser escogidas que descendencia alejada de los padres.

3.3. Operador de mutación

El operador de mutación es el encargado de la búsqueda en el algoritmo. La mutación de un bit de una solución cambia su valor de 1 a 0 o en el sentido contrario. La mutación es necesaria en los algoritmos genéticos para preservar la diversidad de soluciones o strings de la población. Esta mutación no es un proceso del todo aleatorio ya que el algoritmo se enfoca en la creación de nuevas soluciones en el espacio de búsqueda.

Debido a que ninguno de los operadores descritos (selección, cruce y mutación) siguen procesos deterministicos, no es posible garantizar que se tendrá la mejor descendencia. Sin embargo, al pasar las generaciones se irán refinando las soluciones.

3.3.1. Mutación polinomial

En este operador, la distribución de probabilidad de mutación sigue una función polinomial.

$$y_i^{(1,t+1)} = x_i^{(1,t+1)} + \left(x_i^{(U)} - x_i^{(L)}\right)\overline{\delta}_i \tag{3.6}$$

Donde $\overline{\delta}_i$ es calculado a partir de la distribución polinomial $P(\delta) = 0.5(\eta_m + 1)(1 - |\delta|)^{\eta_m}$:

$$\overline{\delta}_i = \begin{cases} (2r_i)^{1/(\eta_m+1)} - 1 & \text{si } r_i < 0.5 \\ 1 - \left[2(1 - r_i)^{1/(\eta_m+1)}\right] & \text{si } r_i \geq 0.5 \end{cases} \tag{3.7}$$

Es posible evidenciar que el valor de η_m produce perturbaciones de orden $\mathrm{O}(1/\eta_m)$ en la variable de decisión normalizada. De esta manera, la distribución de probabilidad es controlada directamente por un valor externo η_m y no cambia dinámicamente con las generaciones.

Capítulo 4

Mapas auto-organizados

Los mapas auto-organizados o SOMs por sus siglas en inglés (*Self-organazing maps*), son un método de aprendizaje no supervisado desarrollado por el profesor Teuvo Kohonen [15]. Estos son una herramienta muy poderosa para la visualización de datos de alta dimensionalidad. Este convierte las relaciones estadísticas no lineales entre datos de alta dimensión en relaciones geométricas simples de sus puntos de imagen en una pantalla de baja dimensión, generalmente una cuadrícula de nodos bidimensional regular. A pesar que los SOMs comprimen la información de entrada, estos mantienen la topología de estos datos. Sin embargo, esto también genera *abstracciones*. Estos aspectos de abstracción y visualización pueden ser utilizados de múltiples procesos complejos tales como análisis de procesos, control, comunicación, etc [11].

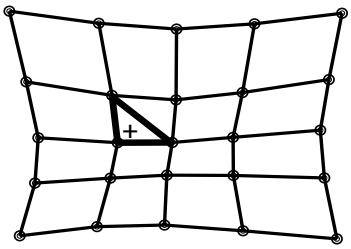


Figura 4.1: SOM con 25 neuronas y en un *lattice* 2D

El SOM puede ser descrito formalmente como un mapeo no lineal, ordenado y uniforme de múltiples datos de entrada de alta dimensión en los elementos de una matriz regular de baja dimensión. Por simplicidad, asumamos que el conjunto de variables de entrada $\{\zeta_j\}$ es definido como el vector $x = [\zeta_1, \zeta_2, \cdots, \zeta_n]^T \in \mathbb{R}^n$. A cada elemento de la matriz regular del SOM se le asocia un vector paramétrico real $m_i = [\mu_{i1}, \mu_{i2}, \cdots, \mu_{in}]^T \in \mathbb{R}^n$ que llamaremos un *modelo*. Asumiendo el cálculo de la distancia entre x_i y m_i denotado como $d(x, m)$, la *imagen* del vector de entrada x en el SOM es definido como el elemento de la matriz regular m_c que mejor aproxima x. Esto quiere decir que

$$c = \arg\min_i \{d(x, m_i)\}. \tag{4.1}$$

4.0.1. Proceso de entrenamiento

Sea $x \in \mathbb{R}^n$ sea un vector de datos estocásticos. Se podría decir que un SOM es una "proyección no lineal" de la función de densidad de probabilidad $p(x)$ del vector de entrada de alta dimensión x, en la pantalla bidimensional. Para definir el mejor nodo coincidente se utiliza la *distancia Euclidiana*, así $d(x, m_i) = \|x - m_i\|_2$.

El SOM es entrenado con el conjunto de entrada x_j y para cada valor, se calcula c de tal manera que todas las neuronas se acercan al valor inicial x_j de la siguiente manera:

$$w_i^{\text{new}} = w_i^{\text{old}} + h(c, i) \cdot (x_i - w_i), \quad i = 1...s \tag{4.2}$$

Donde $h(c, i)$ es el *neighborhood kernel* en su forma más simple es definido como $h(c, c) = 1$, $h(c, i) \geq 0 \; \forall \; w_i$. También se utiliza el *bubble kernel* definido como:

$$h(c, i) = \begin{cases} \alpha & \text{si } r(c, i) < r_0 \\ 0 & \text{de lo contrario} \end{cases} \tag{4.3}$$

En la ecuación 4.3, α es la tasa de aprendizaje, $r(c, i)$ es la distancia Euclidiana entre los nodos c e i del *lattice* y r_0 define el tamaño de la burbuja o *bubble*. La función de vecindad escogida para este trabajo fue la *función Gaussiana*. El algoritmo 1 muestra cómo es el proceso de aprendizaje del SOM.

Algoritmo 1 SOM

1: Inicializar vector de pesos de cada neurona como punto de entrenamiento seleccionado al azar de $\mathbf{S}$
2: **for** $g = 1, \cdots, G$ **do**
3: Ajustar radio de vecindad, σ, y la tasa de aprendizaje, τ:

$$\sigma = \sigma_0 \times \left(1 - \frac{g}{G}\right), \tau = \tau_0 \times \left(1 - \frac{g}{G}\right)$$

4: Seleccionar de manera aleatoria punto $\mathbf{x} \in \mathbf{S}$
5: Encontrar la neurona más cercana:

$$u' = \underset{1 \leq u \leq D}{\arg\min} \|\mathbf{x} - \mathbf{w}^u\|_2$$

6: Encontrar neuronas vecinas:

$$\mathbf{U} = \{u | 1 \leq u \leq D \wedge \|\mathbf{z}^u - \mathbf{z}^{u'}\|_2 \leq \sigma\}$$

7: Actualizar neuronas vecinas ($u \in \mathbf{U}$) así

$$\mathbf{w}^u = \mathbf{w}^u + \tau \cdot \exp\Big\{(-\|\mathbf{z}^u - \mathbf{z}^{u'}\|_2)\Big\}(\mathbf{x} - \mathbf{w}^u)$$

8: **end for**
9: **return** Vector de pesos de todas las neuronas $\mathbf{w}^u, u = 1, \cdots, D$.

Capítulo 5

Algoritmo evolutivo multi-objetivo auto-organizado

Una gran variedad de métodos de aprendizaje tanto autónomo como estadístico han sido propuestos para la extracción de patrones de un conjunto de datos [7]. La motivación del algoritmo propuesto aquí radica en el uso de un mapa auto-organizado para extraer información de la relación entre individuos de una población y utilizarla para la generación de nuevas soluciones. La propiedad más importante de este acercamiento es que el frente de Pareto de un MOOP de m número de objetivos, es un espacio topológico de dimensión $(m-1)$. De esta manera, la dimensión del frente de Pareto de un MOOP de dos objetivos es 1, lo cual corresponde a una curva; mientras que la dimensión de un frente de Pareto de un MOOP de tres objetivos es 2, lo cual corresponde a una superficie. Bajo esta propiedad, en este capítulo se explica a fondo el rol de los mapas auto-organizados dentro de algoritmo evolutivo, así como el funcionamiento general del algoritmo.

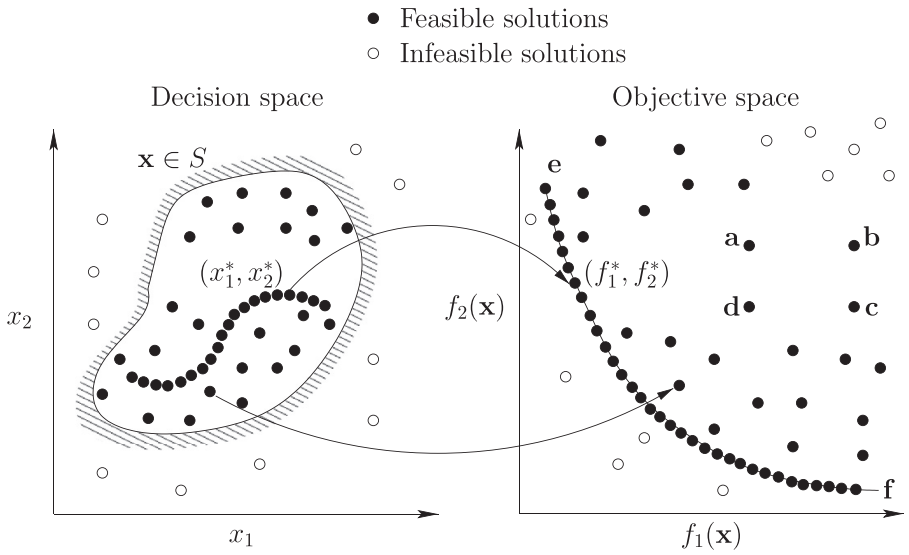
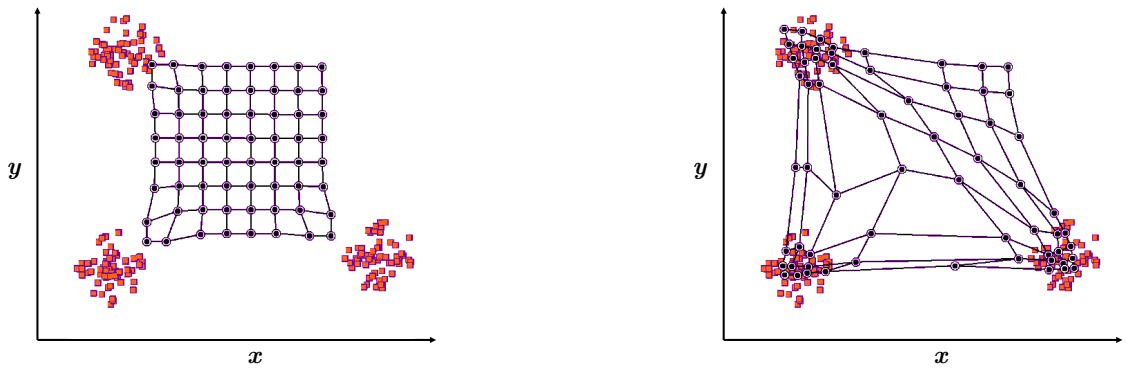


Figura 5.1: Mapeo de soluciones desde un espacio de decisión de 2 variables a un espacio objetivo de dimensión 2 [16]. Nótese que el frente de Pareto tiene dimensión 1.



(a) Antes de entrenamiento

(b) Después de entrenamiento

Figura 5.2: Visualización de ejemplo: SOM de 8x8 entrenada sobre un conjunto de datos de dos dimensiones

5.1. Rol del mapa auto-organizado

Como se mostró en el capítulo 4, los mapas auto-organizados (SOMs) son herramientas capaces de visualizar espacios de altas dimensiones, convirtiendo sus relaciones estadísticas no lineales en relaciones geométricas simples, en mallas de nodos de dos dimensiones [11]. Esto se puede visualizar de manera más sencilla con datos en bajas dimensiones.

Para un conjunto de datos en dos dimensiones, un mapa auto-organizado intentará cubrir, luego del proceso de aprendizaje, toda el área donde los datos están presentes. Como se puede observar en la figura 5.2, donde los puntos negros conectados representan un SOM de 8x8 (cada punto es una neurona), y donde los cuadros naranjas son los datos de entrenamiento, el SOM reubica la mayoría de sus neuronas sobre los datos de entrenamiento, pero no todas. Esto se debe a que el algoritmo de actualización de pesos de cada neurona depende tanto del dato de entrenamiento, como de la distancia a la neurona que está siendo actualizada en determinada iteración. Es por esto que no se puede obtener una relación 1:1 entre un mapa auto-organizado y un conjunto de datos; durante el entrenamiento de una neurona, sus vecinas siempre se verán alteradas, lo que afectará en mayor o menor medida el entrenamiento previo de dichas neuronas.

Sin embargo, esta relación "directa" entre el conjunto de datos de entrenamiento y el mapa no existe en el marco de este algoritmo. Aquí, el objetivo principal del SOM es la aproximación del frente de Pareto desde el espacio de diseño. Es decir, el conjunto de entrada al mapa serán soluciones al MOOP con n variables de decisión (conjunto de alta dimensionalidad) que será representado en un espacio de dimensión $m-1$ (baja dimensionalidad). De esta manera, se restringe el cruce entre soluciones que no sean vecinas durante el proceso evolutivo. Se puede elaborar entonces, que el principal uso de un mapa auto-organizado en este algoritmo es el de aproximar la distribución de soluciones sobre el frente de Pareto por medio de un proceso de *clustering*.

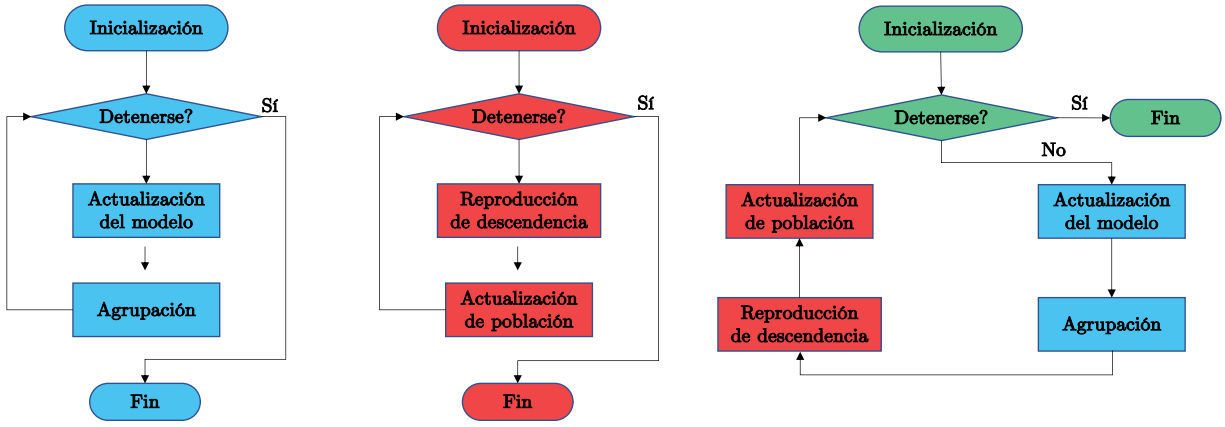
Un enfoque diferente al anteriormente mencionado fue implementado por el autor a manera de explorar diferentes maneras de aproximar el frente de Pareto de un MOOP. Este consistió en utilizar el mapa auto-organizado para aproximar el frente de Pareto utilizando información tanto de las variables de decisión de cada solución, como el valor de sus objetivo (i.e. soluciones evaluadas en el espacio objetivo). Esta propuesta resultó siendo contraproducente, lo que llevó a implementar el enfoque mostrado en este capítulo. Una discusión del porqué

creemos que esto ocurrió se ofrece más adelante.

Aproximaciones similares han sido realizadas por [8] y [17], sin embargo, ambos métodos difieren con el mostrado en este trabajo en el uso del mapa auto-organizado para la generación de nuevas soluciones.

5.2. Estructura del algoritmo

Como se puede observar en la figura 5.3, el algoritmo propuesto (AEAO) combina tanto los procesos del mapa auto-organizado como los del algoritmo evolutivo, realizándolos de manera alternada; primero el entrenamiento del mapa auto-organizado y con base en esta información, el proceso evolutivo de las soluciones.



(a) SOM (b) AE (C) AEAO

Figura 5.3: Fusión de mapa auto-organizado y algoritmo evolutivo multi-objetivo

En la figura, el bloque *Actualización de modelo* representa el proceso que ocurre en el algoritmo 1; soluciones aleatorias de la población actual son seleccionadas para el proceso de entrenamiento del mapa. En el bloque *Agrupación*, el mapa auto-organizado ya entrenado es aplicado a la población total actual. Esto asignará a cada neurona un conjunto de soluciones que mejor se acercan a cada una de las neuronas entrenadas. Este bloque representa el proceso de *clustering* antes mencionado.

En cada una de las generaciones del algoritmo, las relaciones de vecindad de las soluciones de la población de dicha generación son extraídas por el mapa auto-organizado. Seguidamente, basados en esta información encontrada, nuevas soluciones son generadas, restringiendo el proceso de recombinación entre soluciones vecinas.

La siguientes notaciones serán utilizadas durante la descripción del algoritmo:

- G: Número máximo de iteraciones de entrenamiento del mapa auto-organizado
- τ_0: Tasa de aprendizaje inicial del mapa auto-organizado
- σ_0: Radio inicial de vecindad para actualización de pesos de cada neurona
- N': Tamaño de población
- β: Probabilidad de reproducción con soluciones vecinas
- T: Número máximo de generaciones

El algoritmo 2 muestra el marco del algoritmo propuesto en este trabajo. De manera general se tiene que $rand()$ devuelve un número aleatorio distribuido uniformemente entre $[0.0, 1.0]$.

Algoritmo 2 AEMOAO

Inicializar de manera aleatoria población $\mathbf{P} = \{\mathbf{x}^1, \cdots, \mathbf{x}^N\}$. Definir conjunto de entrenamiento $\mathbf{S} = \mathbf{P}$. Inicializar vector de pesos de las neuronas como $\{\mathbf{w}^1, \cdots, \mathbf{w}^D\} = \{\text{rand}(\mathbf{P})\}$.
for $t = 1, \cdots, T$ **do**
    $\mathbf{P} = \text{SeleccionTorneo}(\mathbf{P})$
    $\mathbf{S} = \mathbf{P}$
    for $g = 1, \cdots, G$ **do**
        Actualizar parámetros de entrenamiento

$$\sigma = \sigma_0 \times (1 - \frac{g}{G})$$
$$\tau = \tau_0 \times (1 - \frac{g}{G})$$

        De manera aleatoria seleccionar un punto de entrenamiento $\mathbf{x}^s \in \mathbf{S}$.
        Encontrar la neurona más cercana a $\mathbf{x}^s$:

$$u' = \arg\min_{1 \leq u \leq D} \|\mathbf{x}^s - \mathbf{w}^u\|_2$$

        Encontrar neuronas vecinas:

$$\mathbf{U} = \{u | 1 \leq u \leq D \wedge \|\mathbf{z}^u - \mathbf{z}^{u'}\|_2 \leq \sigma\}$$

        Actualizar neuronas vecinas ($u \in \mathbf{U}$) así

$$\mathbf{w}^u = \mathbf{w}^u + \tau \cdot \exp\Big\{(-\|\mathbf{z}^u - \mathbf{z}^{u'}\|_2)\Big\}(\mathbf{x} - \mathbf{w}^u)$$

    end for
    Establecer nueva población $\mathbf{P}^{\text{new}} = \varnothing$
    while $\mathbf{P}^{\text{new}} < N'$ **do**
        Sea $\mathbf{Q}^d = \{\mathbf{x}^1, \cdots, \mathbf{x}^n\}$ el conjunto de soluciones agrupadas luego del entrenamiento en la neurona d
        Seleccionar aleatoriamente $d | \bar{\bar{\mathbf{Q}}}^d > 1$
        Definir si se utiliza la relación de vecindad o no:

$$\mathbf{Q}^* = \begin{cases} \mathbf{Q}^d & \text{si } rand() \leq \beta \\ \mathbf{P} & \text{de lo contrario} \end{cases}$$

        Generar una nueva solución $\mathbf{y} = \text{Generate}(\mathbf{Q}^*)$
        Añadir nueva solución a $\mathbf{P}^{\text{new}}$: $\{\mathbf{P}^{\text{new}} \cup \mathbf{y}\}$
    end while
    Actualizar población: $\mathbf{P} = \text{Select}(\mathbf{P}^{\text{new}})$
end for
return Población $\mathbf{P}$

Se puede entender el algoritmo propuesto como el proceso de encontrar información sobre la relación entre soluciones de una generación con un mapa auto-organizado. Este proceso de aprendizaje se reinicia una vez se han evaluado y seleccionado las soluciones de una población. Es decir, el algoritmo en cierta medida pierde información de pasadas generaciones con el fin de tener una aproximación más precisa del frente de Pareto en la generación actual. Esto evita que datos de poblaciones pasadas alteren la aproximación del frente con la población actual, que en teoría, está más cerca del frente que sus antecesores. Sin embargo, esto produce mayor costo computacional, algo a mejorar en trabajos futuros. El proceso de aprendizaje del mapa auto-organizado comienza en la línea 4, y de manera iterativa actualiza los parámetros de aprendizaje en las líneas 5-11. Una vez el entrenamiento del mapa ha finalizado, se realiza una agrupación de las soluciones de la población actual en *clusters*, dependiendo a qué neurona del mapa está más cerca cada solución (línea 14). El algoritmo propuesto comienza en la línea 1 y combina los procesos de un MOEA y un SOM. Las soluciones se generan en la línea 17 y actualizan la población en la línea 20.

La gran diferencia del algoritmo propuesto respecto a otros MOEAs se encuentra en la línea 16. Cada solución es asociada a una neurona diferente formando grupos de soluciones similares entre sí para luego ser cruzadas. Sin embargo, este proceso depende de una probabilidad β. Existe una probabilidad $1 - \beta$ que una solución se cruce con cualquier otra solución de la población y una probabilidad β de que se cruce con una solución vecina. Además, una neurona con menos de dos soluciones asociadas no será tenida en cuenta (línea 15). Esto, debido a que se necesitan por lo menos dos soluciones diferentes para el cruce de estas.

5.2.1. Cruce y mutación de soluciones

El procedimiento de generación de soluciones $\mathbf{y} = \text{Generate}(\mathbf{Q}^*)$ (línea 17 en algoritmo 1) emplea dos operadores para crear nuevas soluciones. En primer lugar, selecciona de manera aleatoria dos soluciones del conjunto $\mathbf{Q}^*$. Luego, utiliza el operador de *cruce binario simulado* (SBX), discutido en la sección 3.2.1 para crear una solución de prueba. Posteriormente, se utiliza el operador de *mutación polinomial*, discutido en la sección 3.3.1, para mutar la solución de prueba. Este se muestra en más detalle en el algoritmo 3.

En el algoritmo, rand() devuelve un número aleatorio distribuido uniformemente entre $[0.0, 1.0]$; η_c denota la distribución de probabilidad para el operador SBX; p_m denota la probabilidad de mutación de una solución dada en el operador de mutación polinomial y η_m denota la distribución de probabilidad para el operador de mutación polinomial.

La razón principal para escoger el operador SBX radica, como es discutido en la sección 3.2.1, en sus propiedades; el operador SBX se sesga a soluciones cercanas a los padres de manera más favorable que a soluciones alejadas de estos [14]. Dado que el objetivo del uso del mapa auto-organizado es el cruce de soluciones que sean cercanas, resulta natural utilizar un operador que siga la misma dinámica a la hora de crear nuevas soluciones. Nótese que cualquier otro operador de generación de descendencia puede ser usado, sin embargo, solo los especificados aquí han sido probados por el autor.

5.2.2. Operador de selección

El proceso $\mathbf{P} = \text{SeleccionTorneo}(\mathbf{P})$ (línea 3) prepara la población antes de iniciar el proceso de generación de descendencia. Esto con el fin de no eliminar las malas soluciones y no tenerlas en cuenta a la hora generar nuevas soluciones. Este proceso es selección por torneo, explicado en detalle en la sección 3.2. Los principales objetivos de este proceso son:

1. Identificar buenas (mejor que el promedio) soluciones en la población.
2. Realizar múltiples copias de buenas soluciones.
3. Eliminar malas soluciones de la población, de tal manera que existan más copias de buenas soluciones.

Algoritmo 3 $\mathbf{y} = \text{Generate}(\mathbf{Q}^*)$

1: De manera aleatoria seleccionar dos soluciones $\mathbf{x}^1$ y $\mathbf{x}^2$ de $\mathbf{Q}^*$ que sean diferentes entre sí.
2: Definir $u_i = \text{rand}()$
3: Calcular β_{q_i} así:

$$\beta_{q_i} = \begin{cases} (2u_i)^{\frac{1}{\eta_c+1}} & \text{si } u_i \leq 0.5 \\ \left(\frac{1}{2(1-u_i)}\right)^{\frac{1}{\eta_c+1}} & \text{de lo contrario} \end{cases}$$

4: Calcular la nuevas soluciones de prueba $\mathbf{x}^{1,t+1} = (x_1^{(1,t+1)}, \cdots, x_n^{(1,t+1)})$, $\mathbf{x}^{2,t+1} = (x_1^{(2,t+1)}, \cdots, x_n^{(2,t+1)})$ con $(i = 1, \cdots, n)$ como:

$$x_i^{(1,t+1)} = 0.5\left[(1+\beta_{q_i})x_i^{1,t} + (1-\beta_{q_i})x_i^{2,t}\right]$$

$$x_i^{(2,t+1)} = 0.5\left[(1-\beta_{q_i})x_i^{1,t} + (1+\beta_{q_i})x_i^{2,t}\right]$$

5: Seleccionar aleatoriamente entre $\mathbf{x}^{(1,t+1)}$ y $\mathbf{x}^{(2,t+1)}$:

$$\mathbf{y}' = \text{rand.choice}[\mathbf{x}^{(1,t+1)}, \mathbf{x}^{(2,t+1)}]$$

6: Mutar la solución de prueba con $(i = 1, \cdots, n)$

$$y_i = \begin{cases} y_i' + (b_i - a_i)\,\bar{\delta}_i & \text{si rand}() < p_m \\ y_i' & \text{de lo contrario} \end{cases}$$

donde a_i, b_i son los límites superior e inferior de las variables x_i, $r = \text{rand}()$ y

$$\bar{\delta}_i = \begin{cases} (2r)^{1/\eta_m+1} - 1, & \text{si } r < 0.5, \\ 1 - \left[2\,(1-r)\right]^{1/(\eta_m+1)}, & \text{si } r > 0.5. \end{cases}$$

7: **return** Nueva solución $\mathbf{y}$

Capítulo 6

Casos de prueba y métricas de rendimiento

6.1. Casos de prueba

El algoritmo propuesto fue probado con 5 problemas del banco de pruebas DTLZ encontrado en [18]. Este banco de pruebas es escalable a cualquier número de variables de decisión y cualquier número de objetivos. De esta manera, para la medición del desempeño del algoritmo propuesto, se definieron 10 variables de decisión para cada problema; todos los problemas ejecutados son bi-ojetivo. Los detalles de este banco de pruebas se encuentra en la tabla 6.1.

Por otro lado, se utilizó el banco de pruebas WFG. Esta banco tiene problemas con frentes de Pareto tanto sencillos como complicados. Esta instancia de prueba es mucho más sofisticada que los problemas ZDTL. Para el lector curioso que desee mayor información referente a la definición de los problemas, puede referirse a [19].

6.2. Métricas de rendimiento

Para mostrar y comparar el desempeño del algoritmo propuesto, dos indicadores de calidad comúnmente utilizados i.e. *distancia generacional invertida (IGD)* [20] e *hipervolumen (HV)* [21], se emplean en este trabajo.

6.2.1. Distancia generacional invertida

Sea $\mathbf{P}^*$ un conjunto de puntos uniformemente distribuidos sobre el frente de Pareto, y $\mathbf{P}$ un frente no dominado, se tiene la siguiente definición del indicador *IGD*, propuesta en [20]

$$IGD\left(\mathbf{P}^*, \mathbf{P}\right) = \frac{\sum_{\mathbf{x}^* \in \mathbf{P}^*} d\left(\mathbf{x}^*, \mathbf{P}\right)}{\bar{\bar{\mathbf{P}}}^*} \tag{6.1}$$

donde $d\left(\mathbf{x}^*, \mathbf{P}\right)$ es la mínima distancia entre $\mathbf{x}^*$ y cualquier punto perteneciente a $\mathbf{P}$, y $\bar{\bar{\mathbf{P}}}^*$ es la cardinalidad de $\mathbf{P}^*$. El frente de Pareto debe ser conocido a fin de utilizar este indicador. En este trabajo, 100 puntos uniformemente distribuídos fueron utilizados para formar $\mathbf{P}^*$.

Instancia	Dominio	Funciones objetivo	Características
DTLZ1	$[0,1]$	$f_1 = (1+g)0.5\prod_{i=1}^{M-1} y_i$ $f_{m=2:M-1} = (1+g)0.5\left(\prod_{i=1}^{M-m} y_i\right)(1-y_{M-m+1})$ $f_M = (1+g)0.5(1-y_1)$ $g = 100\left[k+\sum_{i=1}^{k}\left((z_i-0.5)^2-\cos\left(20\pi(z_i-0.5)\right)\right)\right]$	FP Lineal
DTLZ2	$[0,1]$	$f_1 = (1+g)\prod_{i=1}^{M-1}\cos\left(y_i\pi/2\right)$ $f_{m=2:M-1} = (1+g)\left(\prod_{i=1}^{M-m}\cos\left(y_i\pi/2\right)\right)\sin\left(y_{M-m+1}\pi/2\right)$ $f_M = (1+g)\sin\left(y_1\pi/2\right)$ $g = \sum_{i=1}^{k}(z_i-0.5)^2$	FP No convexo
DTLZ3	$[0,1]$	$f_1 = (1+g)\prod_{i=1}^{M-1}\cos\left(y_i\pi/2\right)$ $f_{m=2:M-1} = (1+g)\left(\prod_{i=1}^{M-m}\cos\left(y_i\pi/2\right)\right)\sin\left(y_{M-m+1}\pi/2\right)$ $f_M = (1+g)\sin\left(y_1\pi/2\right)$ $g = 100\left[k+\sum_{i=1}^{k}\left((z_i-0.5)^2-\cos\left(20\pi(z_i-0.5)\right)\right)\right]$	FP No convexo
DTLZ4	$[0,1]$	$f_1 = (1+g)\prod_{i=1}^{M-1}\cos\left(y_i^{\alpha}\pi/2\right)$ $f_{m=2:M-1} = (1+g)\left(\prod_{i=1}^{M-m}\cos\left(y_i^{\alpha}\pi/2\right)\right)\sin\left(y_{M-m+1}\pi/2\right)$ $f_M = (1+g)\sin\left(y_1\pi/2\right)$ $g = \sum_{i=1}^{k}(z_i-0.5)^2$	FP No convexo
DTLZ7	$[0,1]$	$f_{m=1:M} = y_m$ $f_M = (1+g)\left(M-\sum_{i=1}^{M-1}\left[\frac{f_i}{1+g}(1+\sin\left(3\pi f_i\right))\right]\right)$ $g = 1+9\sum_{i=1}^{M-1} z_i/k$	FP desconectado

Tabla 6.1: Instancias de prueba DTLZ

6.2.2. Hipervolumen

Este indicador reporta el tamaño (área, volumen o hipervolumen, dependiendo del número de objetivos) del espacio dominado por cualquier punto del frente de Pareto.

$$HV\left(\mathbf{P},\mathbf{r}\right) = VOL\left(\bigcup_{\mathbf{x}\in\mathbf{P}}\left[f_1\left(\mathbf{x}\right),r_1\right]\times\cdots\left[f_m\left(\mathbf{x}\right),r_m\right]\right) \tag{6.2}$$

donde $\mathbf{r} = (r_1,\cdots,r_m)$ es un punto de referencia en el espacio objetivo dominado por cualquier punto ubicado sobre el frente de Pareto y $VOL\left(\cdot\right)$ es la medida de Lebesgue. En este trabajo se utilizó $\mathbf{r} = (0.6, 0.6)$ para DTLZ1, $\mathbf{r} = (2, 2)$ para DTLZ2, DTLZ3 y DTLZ4 y $\mathbf{r} = (0.8, 4.0)$ para DTLZ7. Para todas las instancias del banco de pruebas WFG se utilizó $\mathbf{r} = (3, 5)$.

6.3. Estudios de comparación

En esta sección se compara el algoritmo propuesto con 4 algoritmos de vanguardia: *Non-dominated Sorting Genetic Algorithm* NSGA-II [4], *Strength Pareto Evolutionary Algortihm* SPEA2 [5], *Multiobjective Evolutionary Algorithm Based On Decomposition* MOEAD [22] y *ϵ-MOEA* EpsMOEA [23].

6.3.1. Parámetros experimentales

Parámetros públicos

- Las poblaciones de todos los algoritmos fue de $N = 100$ individuos.
- El número de variables de decisión para todos los problemas fue de $n = 10$.
- Cada algoritmo fue ejecutado de manera independiente 20 veces.
- Condición de parada: Todos los algoritmos realizaron 45,000 evaluaciones por ejecución.

Parámetros del algoritmo propuesto

- Estructura del SOM: 1×100
- Tasa de aprendizaje inicial: $\tau_0 = 0.7$
- Restricción de vecindad: $\beta = 0.8$
- Radio de vecinos afectados: $\sigma = 10.0$
- Parámetros de SBX: $p_c = 1.0$, $\eta_c = 15.0$
- Parámetros de PM: $p_m = 1.0$, $\eta_m = 1.0$

Parámetros de MOEAD

- Tamaño vecindad: $NS = 5$
- Probabilidad de seleccionar padres desde vecindario: $\beta = 0.9$
- Número de soluciones reemplazadas por una nueva solución: 2

6.3.2. Parámetros de NSGA-II

- Parámetros de control para el operador *Differential Evolution*: $F = 0.5$ y $CR = 1.0$
- Parámetros de control para el operador PM: $p_m = 1.0$ y $\eta_m = 20$

6.3.3. Parámetros para EpsMOEA

- Tamaño $\epsilon = 0.01$
- Parámetros de control para el operador *Differential Evolution*: $F = 0.5$ y $CR = 1.0$

Se puede observar en la figura 6.1 cómo el algoritmo propuesto encuentra de manera correcta los frentes de Pareto de la mayoría de los problemas del banco DTLZ. Sin embargo, para el problema DTLZ3 se puede observar que el algoritmo encontró una solución sub-óptima, al no estar directamente sobre el frente de Pareto real del problema. Se puede observar también que se el algoritmo encuentra el frente de manera correcta en los extremos.

En la figura 6.2 se puede observar nuevamente que el algoritmo encuentra un frente de Pareto sub-óptimo para el problema WFG1, teniendo también los peores indicadores en este problema, el cual se considera un frente de Pareto complicado. Esto puede ser resulto de

Tabla 6.2: Resultados estadísticos (Promedio (Desviación estándar) [Rango]) de 5 algoritmos sobre 20 ejecuciones independientes en el banco de pruebas ZTDL en términos de IDG y HV

Instancia	EpsMOEA	MOEAD	SPEA2	NSGA-II	Alg. Prop
			IGD		
DTLZ1	4.6543.E-03 [3] (1.382.E-05)	5.4234.E-03 [4] (1.152.E-05)	5.4765.E-03 [5] (2.332.E-01)	4.5943.E-03 [2] (1.543.E-03)	**4.5241.E-03 [1]** (1.123.E-04)
DTLZ2	**2.7945.E-03 [1]** (1.351.E-04)	4.6088.E-03 [5] (1.651.E-04)	4.3994.E-03 [4] (1.157.E-05)	3.9995.E-03 [3] (3.531.E-06)	3.0535.E-03 [2] (1.235.E-04)
DTLZ3	1.6247.E-03 [3] (1.234.E-03)	3.4824.E-03 [4] (1.234.E-04)	3.4965.E-03 [5] (1.234.E-03)	1.5921.E-03 [2] (1.396.E-05)	**1.5281.E-03 [1]** (1.654.E-04)
DTLZ4	**2.6518.E-03 [1]** (2.345.E-03)	4.6235.E-03 [4] (1.543.E-04)	4.6543.E-03 [5] (2.934.E-06)	4.0665.E-03 [2] (2.341.E-04)	4.3135.E-03 [3] (2.443.E-04)
DTLZ7	**2.6518.E-03 [1]** (3.456.E-03)	4.5689.E-03 [4] (4.654.E-04)	4.0649.E-03 [2] (3.457.E-04)	4.6038.E-03 [3] (1.235.E-04)	5.3135.E-03 [5] (1.375.E-03)
			HV		
DTLZ1	0.989065 [5] (6.834.E-05)	0.991533 [4] (2.174.E-05)	**0.991551 [1]** (1.636.E-05)	0.991541 [2] (2.029.E-05)	0.991389 [3] (1.480.E-04)
DTLZ2	0.94656704 [5] (1.351.E-04)	0.947328 [3] (1.655.E-04)	0.9473495 [2] (1.158.E-05)	0.94731105 [4] (8.508.E-06)	**0.94744846 [1]** (1.235.E-04)
DTLZ3	**0.94620503 [1]** (1.381.E-05)	0.94006839 [3] (1.158.E-05)	0.69016507 [5] (2.335.E-01)	0.945232 [2] (1.101.E-03)	0.90172814 [4] (1.123.E-04)
DTLZ4	0.77853497 [5] (1.654.E-04)	**0.94733396 [1]** (1.381.E-05)	0.89122031 [3] (1.123.E-01)	0.89118052 [4] (1.123.E-01)	0.9473327 [2] (1.655.E-04)
DTLZ7	0.43421102 [5] (1.168.E-05)	0.47366837 [4] (1.123.E-04)	0.47405282 [2] (2.934.E-06)	0.47401195 [3] (1.396.E-05)	**0.47502791 [1]** (1.654.E-04)
Rango promedio	3.0	3.6	3.4	2.7	2.3

varias maneras, siendo las más importantes aumentar el número de evaluaciones del algoritmo, permitiendo que este puedo encontrar soluciones mejores, o realizar una inicialización diferente de la población inicial. Estos algoritmos son altamente dependientes del problema y un ajuste en estos parámetros permite un mejor desempeño. Por otro lado, se puede observar el buen desempeño del algoritmo en las demás instancias del

6.4. Discusión

Los valores promedio y desviaciones estándar de los indicadores IDG e HV para 20 poblaciones finales obtenidas con los 5 algoritmos evaluados se presenta en la tabla 6.2. Los promedios de la métrica $IGD(HV)$ están ordenados de manera ascendente (descendiente), y los números entre corchetes son el rango que obtuvo. Los datos en negrita con fondo gris en la tabla indican que son el mejor indicador de los 5 para cada instancia.

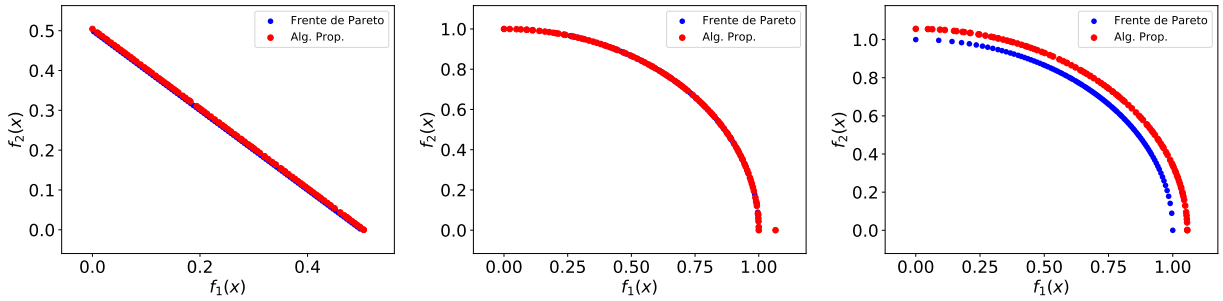
La tabla 6.3 muestra que EpsMOEA, MOEAD, SPEA2, NSGA-II, y el algoritmo propuesto lograron tener el mejor indicador 4, 1, 1, 0, 4 veces respectivamente. El orden por rango obtenido: es algoritmo propuesto, MOEAD, EpsMOEA, SPEA2, NSGA-II, respecto a los promedios obtenidos por cada algoritmo. Esto indica que el algoritmo propuesto es superior a los demás algoritmos en este banco de pruebas. MOEAD tiene un desempeño comparable al algoritmo propuesto. Una de las razones por las que se cree que esto sucede es el uso de el operador DE. Este operador supera significativamente a otros en problemas mono-objetivos.

Cabe notar que la mayoría de estos algoritmos, incluyendo el propuesto, son altamente dependientes del problema. Esto quiere decir que deben ser ajustados para el problema que atacará y puede que los resultados cambien al hacer un ajuste más fino de los parámetros

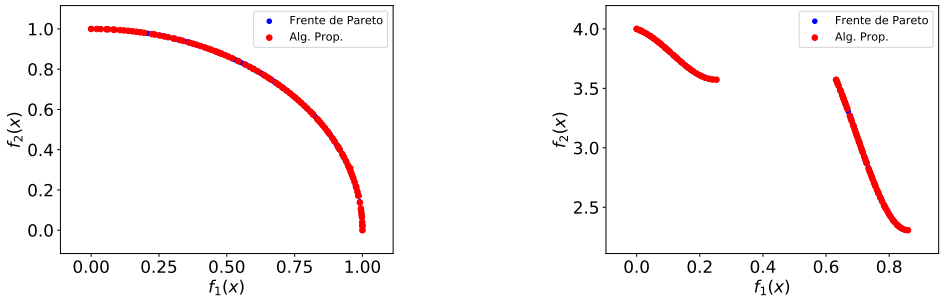
Tabla 6.3: Resultados estadísticos (Promedio (Desviación estándar) [Rango]) de 5 algoritmos sobre 20 ejecuciones independientes en el banco de pruebas WFG en términos de IDG y HV

Instancia	EpsMOEA	MOEAD	SPEA2	NSGA-II	Alg. Prop
			IGD		
WFG1	1.1465.E+00 [2] (7.881.E-03)	**1.1454.E+00 [1]** (1.813.E-04)	1.1535.E+00 [3] (8.813.E-04)	1.1623.E+00 [5] (2.445.E-03)	1.1543.E+00 [4] (5.400.E-03)
WFG2	**1.3654.E+00 [1]** (5.701.E-03)	1.3752.E+00 [2] (1.830.E-03)	1.3789.E+00 [3] (4.704.E-03)	1.4542.E+00 [5] (4.937.E-04)	1.4534.E+00 [4] (4.459.E-04)
WFG3	1.4534.E-01 [2] (7.822.E-03)	1.4543.E-01 [3] (8.687.E-04)	1.4654.E-01 [4] (7.099.E-04)	1.5765.E-01 [5] (3.424.E-03)	**1.4346.E-01 [1]** (4.966.E-04)
WFG4	4.8765.E-02 [5] (9.651.E-03)	4.7654.E-02 [3] (3.631.E-05)	**4.6542.E-02 [1]** (5.986.E-03)	4.7654.E-02 [4] (8.557.E-05)	4.6765.E-02 [2] (8.461.E-04)
WFG5	**7.6540.E-02 [1]** (7.905.E-03)	6.7765.E-02 [3] (6.167.E-03)	6.8877.E-02 [5] (1.517.E-04)	6.7645.E-02 [2] (5.159.E-03)	6.8760.E-02 [4] (1.660.E-05)
WFG6	3.6540.E-02 [5] (2.038.E-03)	3.4321.E-02 [2] (2.063.E-04)	3.5430.E-02 [4] (7.948.E-04)	3.4543.E-02 [3] (1.620.E-04)	**3.3123.E-01 [1]** (5.013.E-05)
WFG7	1.3432.E-02 [3] (9.940.E-03)	1.2432.E-02 [2] (4.586.E-03)	1.4325.E-02 [5] (3.974.E-05)	1.3543.E-02 [4] (9.215.E-04)	**1.2123.E-02 [1]** (5.089.E-03)
WFG8	2.7843.E-02 [5] (9.758.E-04)	**2.6543.E-02 [1]** (7.250.E-03)	2.6654.E-02 [2] (8.563.E-03)	2.7654.E-02 [4] (9.374.E-03)	2.6765.E-02 [3] (3.767.E-04)
WFG9	2.1835.E-02 [5] (8.340.E-04)	2.1654.E-02 [2] (1.104.E-04)	**2.1543.E-02 [1]** (5.334.E-05)	2.1789.E-02 [4] (1.009.E-03)	2.1765.E-02 [3] (4.321.E-05)
			HV		
WFG1	3.751.E+00 [4] (3.245.E-01)	**5.705.E+00 [1]** (6.231.E-01)	4.321.E+00 [3] (6.421.E-03)	4.431.E+00 [2] (3.946.E-02)	3.745.E+00 [5] (8.007.E-03)
WFG2	1.128.E+01 [2] (5.424.E-03)	**1.138.E+01 [1]** (2.549.E-02)	1.101.E+01 [5] (1.810.E-03)	1.112.E+01 [4] (5.328.E-02)	1.122.E+01 [3] (3.833.E-03)
WFG3	1.068.E+01 [4] (4.235.E-03)	1.023.E+01 [5] (4.908.E-02)	1.435.E+01 [2] (8.083.E-03)	1.334.E+01 [3] (2.222.E-02)	**1.493.E+01 [1]** (6.668.E-02)
WFG4	8.212.E+00 [2] (9.653.E-03)	**8.389.E+00 [1]** (9.728.E-02)	8.154.E+00 [3] (5.517.E-02)	8.124.E+00 [4] (9.622.E-02)	7.912.E+00 [5] (6.153.E-02)
WFG5	8.138.E+00 [2] (2.345.E-02)	8.004.E+00 [5] (7.727.E-02)	8.012.E+00 [4] (3.423.E-03)	8.123.E+00 [3] (3.123.E-03)	**8.142.E+00 [1]** (8.978.E-02)
WFG6	6.235.E+00 [2] (7.345.E-02)	6.142.E+00 [4] (4.456.E-03)	6.153.E+00 [3] (6.339.E-02)	6.141.E+00 [5] (4.440.E-03)	**6.364.E+00 [1]** (6.087.E-02)
WFG7	8.477.E+00 [2] (3.242.E-03)	8.412.E+00 [4] (7.662.E-03)	8.413.E+00 [3] (6.615.E-03)	8.401.E+00 [5] (5.454.E-02)	**8.535.E+00 [1]** (2.667.E-03)
WFG8	8.312.E+00 [2] (3.451.E-02)	8.301.E+00 [3] (7.673.E-03)	8.300.E+00 [4] (4.123.E-03)	8.221.E+00 [5] (9.632.E-03)	**8.456.E+00 [1]** (7.461.E-02)
WFG9	**6.145.E+00 [1]** (5.427.E-02)	6.131.E+00 [2] (5.890.E-02)	6.112.E+00 [4] (1.787.E-03)	6.054.E+00 [5] (1.338.E-02)	6.125.E+00 [3] (1.564.E-03)
Rango promedio	2.777	2.5	3.278	4.000	2.444

Figura 6.1: Poblaciones finales del algoritmo propuesto ejecutado sobre el banco de pruebas DTLZ



(a) DTLZ1 (b) DTLZ2 (c) DTLZ3



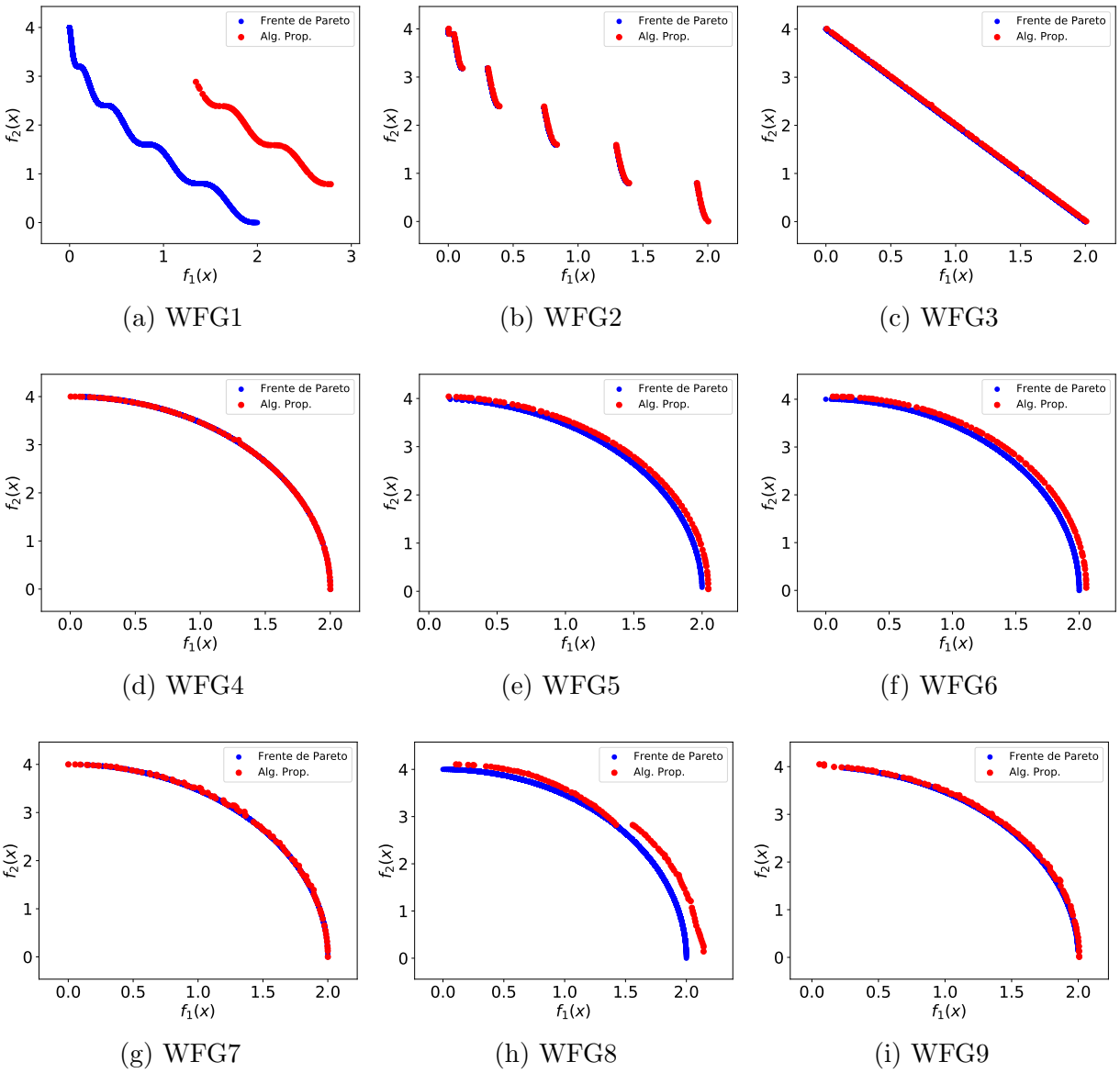
(d) DTLZ4 (e) DTLZ7

de cada uno. Es conocido que la optimización de los parámetros de algoritmos genéticos es una tarea ardua. [17]

La tabla 6.3 muestra que EpsMOEA, MOEAD, SPEA2, NSGA-II, y el algoritmo propuesto lograron tener el mejor indicador 3, 5, 2, 0, 8 veces respectivamente. El orden por rando obtenido es algoritmo propuesto, MOEAD, EpsMOEA, SPEA2, NSGA-II, respecto a los promedios obtenidos por cada algoritmo, lo que es consistente con los resultados obtenidos en el banco de pruebas DTLZ. Esto indica que el algoritmo propuesto es superior a los demás algoritmos en este banco de pruebas.

Las figuras 6.1 y 6.2 muestran los frentes obtenidos luego de 30 ejecuciones independientes, con el indicador IDG promedio de las ejecuciones.

Figura 6.2: Poblaciones finales del algoritmo propuesto ejecutado sobre el banco de pruebas WFG



(a) WFG1 (b) WFG2 (c) WFG3

(d) WFG4 (e) WFG5 (f) WFG6

(g) WFG7 (h) WFG8 (i) WFG9

Capítulo 7

Caso de estudio: Optimización de perfiles aerodinámicos

A pesar que el desempeño de un algoritmo puede ser medido y comparado con otros por medio de pruebas como las mostradas en el capítulo anterior, el propósito de este algoritmo no es solo tener un buen desempeño en dichas pruebas. También se desea que frente a un problema de ingeniería dicho algoritmo sea lo suficientemente robusto para poder atacarlo, se desempeñe consistentemente bien, y, sin resolver el problema dada su naturaleza multi-objetivo, provea las herramientas necesarias para la correcta toma de decisiones sobre el problema.

De esta manera, en el presente capítulo se presenta un problema de optimización en perfiles aerodinámicos; problema de especial interés del autor. A pesar que el problema presentado no es terriblemente complicado en el sentido que presenta únicamente 6 variables de decisión con dos objetivos, este sirve para mostrar como prueba de concepto las capacidades del algoritmo y cómo este sirve como herramienta para toma de decisiones.

7.1. Generalidades

El diseño aerodinámico suele ser complejo y generalmente se distribuye en diferentes fases: diseño conceptual, diseño preliminar y diseño detallado. Sin embargo, cada fase puede ser atacada de dos maneras: 1) diseño inverso y 2) optimización numérica directa. El diseño inverso resuelve la distribución de presión para una geometría preestablecida y acoplando una definición paramétrica de geometría y un modulo de optimización numérica, varía de manera iterativa dicha geometría hasta alcanzar los valores óptimos de diseño definidos, así como cumplir las restricciones impuestas al problema [24]. Dependiendo si el objetivo es mejorar un diseño existente o crear un diseño totalmente nuevo, la selección del método de parametrización de geometría será diferente. Generalmente, si el diseño requiere únicamente pequeños ajustes, un método de parametrización localizado será preferido. Pero, si el estudio que se desea realizar es sobre un diseño totalmente nuevo, un método de parametrización a utilizar debe ser capaz de generar una gran variedad de nuevas formas a explorar [25].

La forma de un perfil aerodinámico es algo que se estudia desde la época de los 20s. Funciones analíticas para el cálculo de un perfil han sido derivadas para representar toda una familia de perfiles, tal como la familia de perfiles NACA de cuatro o cinco dígitos [12]. Sin embargo, existen métodos para aproximar todo tipo de forma aerodinámica como lo son las curvas

Bezier, B-Splines, CST o PARSEC.

El concepto bajo el que se rigen estos métodos es la utilización de relativamente pocas funciones con el fin de aproximar la mayor cantidad de perfiles existentes. No obstante, estos métodos pueden ser utilizados como punto de partida para la generación de nuevos diseños, sin tener en cuenta si imitan o no la forma de perfiles ya existentes.

Dentro de los estudios publicados por el Instituto Americano de Aeronáutica y Astronáutica (AIAA) se encuentra la definición estándar de un proceso de diseño aerodinámico [26]. De acuerdo a la publicación, el proceso de optimización consta de:

1. Definición de configuración por un conjunto de variables de diseño
2. Método de optimización
3. Evaluación del desempeño aerodinámico para una nueva configuración

7.2. Definición del problema

Utilizando el método de parametrización de *splines paramétricas de Ferguson* para perfiles aerodinámicos, se realiza una exploración inteligente del espacio de diseño con el fin de encontrar el conjunto de soluciones que optimicen el coeficiente de sustentación y el coeficiente de arrastre.

El cálculo de los coeficientes aerodinámicos para la evaluación del desempeño del perfil generado fue realizado con **XFOIL**, un sistema de análisis y diseño de perfiles aerodinámicos a bajos números de Reynolds, desarrollado por el profesor Mark Drela del Instituto de Tecnología de Massachusetts (MIT). Este recibe como parámetro de entrada las coordenadas de los puntos que forman el perfil aerodinámico, junto con parámetros como número de Reynolds, número de Mach, ángulo de ataque, etc. Con esta información realiza el cálculo de distribución de presión sobre el perfil, así como de las características de sustentación, arrastre y momento de cabeceo. Las coordenadas del perfil aerodinámico son generadas de manera sistemática por un script escrito en el lenguaje de programación python, bajo la definición paramétrica de splines de Ferguson como se explica más adelante. Esta información se acopla al algoritmo propuesto para realizar la exploración del espacio de diseño.

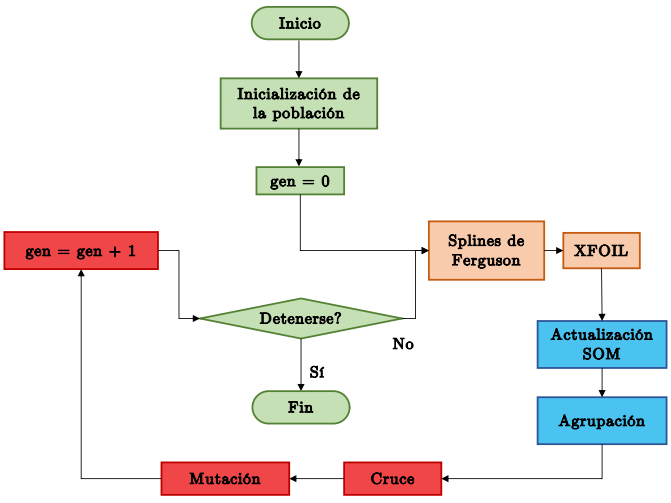


Figura 7.1: Esquema de optimización de perfiles

7.2.1. Spline paramétricas de Ferguson

Este método busca definir una curva paramétrica $\mathbf{r}(u)$ con $u \in [0,1]$, la cual conecta dos puntos $\mathbf{r}(0) = \mathbf{A}$ y $\mathbf{r}(1) = \mathbf{B}$. Se imponen dos tangentes a esta curva así: $\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}u}|_{u=0} = \mathbf{T}_A$ y $\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}u}|_{u=1} = \mathbf{T}_B$ como se muestra en la figura 7.2. La curva paramétrica se define como:

$$\mathbf{r}(u) = \sum_{i=0}^{3} \mathbf{a}_i u^i, \quad u \in [0,1] \tag{7.1}$$

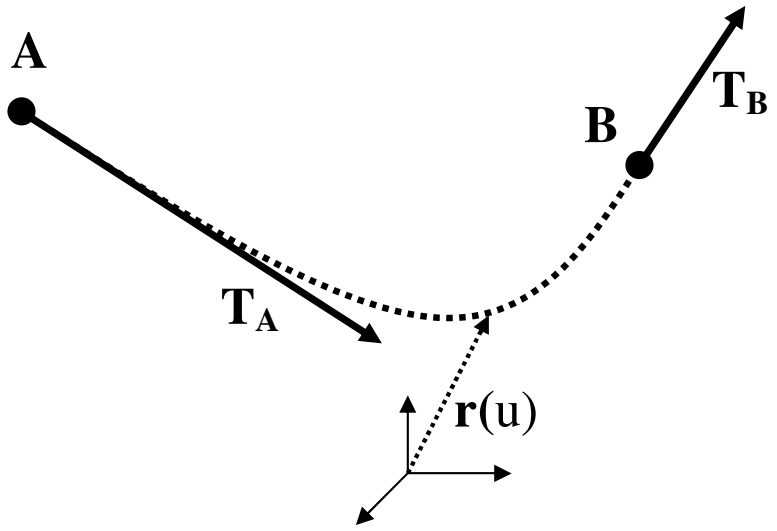


Figura 7.2: Spline de Ferguson con sus condiciones de frontera. Tomado de [27]

Aplicando las condiciones de frontera, se encuentran los cuatro vectores necesarios para definir la curva paramétrica:

$$\begin{aligned} \mathbf{A} &= \mathbf{a}_0 \\ \mathbf{B} &= \mathbf{a}_0 + \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3 \\ \mathbf{T}_A &= \mathbf{a}_1 \\ \mathbf{T}_B &= \mathbf{a}_1 + 2\mathbf{a}_2 + 3\mathbf{a}_3 \end{aligned} \tag{7.2}$$

y, reorganizando para cada vector:

$$\begin{aligned} \mathbf{a}_0 &= \mathbf{A} \\ \mathbf{a}_1 &= \mathbf{T}_A \\ \mathbf{a}_2 &= 3\left[\mathbf{B} - \mathbf{A}\right] - a\mathbf{T}_A - \mathbf{T}_A \\ \mathbf{a}_3 &= 2\left[\mathbf{A} - \mathbf{B}\right] + \mathbf{T}_A + \mathbf{T}_B \end{aligned} \tag{7.3}$$

Sustituyendo en la ecuación 7.1, se obtiene:

$$\mathbf{r}(u) = \mathbf{A}\left(1 - 3u^2 + 2u^3\right) + \mathbf{B}\left(3u^2 - 2u^3\right) + \mathbf{T}_A\left(u - 2u^2 + u^3\right) + \mathbf{T}_B\left(-u^2 + u^3\right), \tag{7.4}$$

o en forma matricial:

$$\mathbf{r}(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{T}_A \\ \mathbf{T}_B \end{bmatrix} \tag{7.5}$$

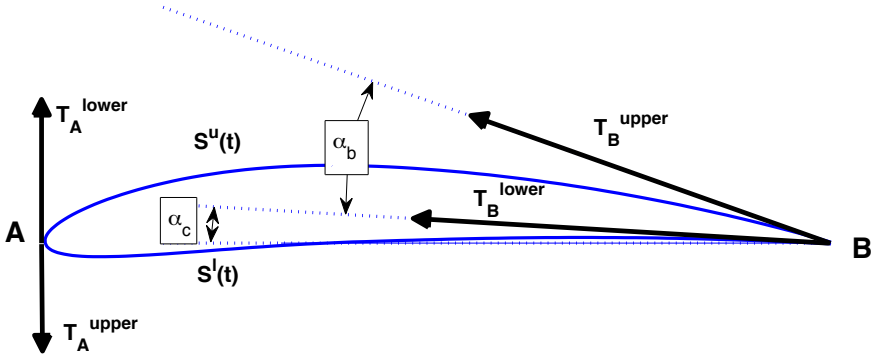


Figura 7.3: Esquema de parametrización de perfiles aerodinámicos basado en dos splines de Ferguson. $\mathbf{S}^u$ representa la superficie superior, $\mathbf{S}^L$ representa la superficie inferior.

En la figura 7.3 se ilustra cómo usando dos splines de Ferguson, una para la superficie superior otra para la superficie inferior, se logra la forma del perfil aerodinámico. Uniendo los puntos tanto iniciales como finales de dichas curvas se logra que variables de diseño se reduzcan a seis (asumiendo un borde de salida afilado del perfil) mientras se mantiene un espacio de diseño relativamente amplio. La tangente de la curva superior $\mathbf{S}^u$ en el punto $\mathbf{A}$ se denota $\mathbf{T}_A^{\text{upper}}$ y la tangente en el punto $\mathbf{B}$ se denota $\mathbf{T}_B^{\text{upper}}$. Esta misma lógica es utilizada para la superficie inferior.

Los vectores $\mathbf{T}_A^{\text{upper}}$ y $\mathbf{T}_A^{\text{lower}}$ siempre apuntan verticalmente hacia abajo y hacia arriba, respectivamente. La magnitud de estos vectores define la 'embotadura' del borde de ataque del perfil. El ángulo α_c, que podría ser llamado ángulo de combadura, define la dirección del vector $\mathbf{T}_B^{\text{lower}}$ y α_b determina la orientación de $\mathbf{T}_B^{\text{upper}}$. La magnitud de estos vectores determina la forma de la zona intermedia del perfil.

7.2.2. Parámetros del problema

Escoger los parámetros correctos para el desarrollo del problema es vital para llegar a un frente de Pareto final que sea de utilidad. Para esto, los límites de las variables de diseño fueron definidos bajo la recomendación de [28]. Es de notar que cerca del 1.7 % del espacio de diseño dentro de los límites mostrados en la tabla 7.1 resulta en soluciones no factibles. Estas soluciones son descartadas en proceso de creación de curvas.

Cada perfil generado es formado por 200 puntos, 100 por cada superficie. XFOIL fue configurado para operar a un Reynolds de 2×10^6 con un máximo de 100 iteraciones por perfil. Un máximo de 9500 perfiles fueron evaluados por ejecución del algoritmo con 100 perfiles por población.

Variable	$\lvert\mathbf{T}_A^{\text{upper}}\rvert$	$\lvert\mathbf{T}_A^{\text{lower}}\rvert$	$\lvert\mathbf{T}_B^{\text{upper}}\rvert$	$\lvert\mathbf{T}_B^{\text{lower}}\rvert$	α_c[deg]	α_b[deg]
Límite inferior	0.1	0.1	0.1	0.1	-15	1
Límite superior	0.4	0.4	2	2	15	30

Tabla 7.1: Rangos de las variables de diseño utilizados en el problema de optimización

7.3. Resultados

Se desarrollaron dos procesos de optimización con el método de parametrización de splines de Ferguson. El primero consistió en la exploración del espacio de diseño evaluando las geometrías con ángulo de ataque de 0º; la segunda fue con este ángulo con valor de 6º.

7.3.1. Optimización con ángulo de ataque a 0º

Luego de la evaluación de 9100 perfiles aerodinámicos, se encontraron 419 soluciones no dominadas, como se observa en la figura 7.4. Se puede observar como las poblaciones gradualmente se acercan al frente de Pareto no dominado encontrado. Cabe notar cómo el algoritmo deja de buscar en zonas dominadas y se centra alrededor de las soluciones no dominadas. La mayoría de soluciones no dominadas se encuentran en el rango de $[0.0, 1.0]$ para el coeficiente de sustentación.

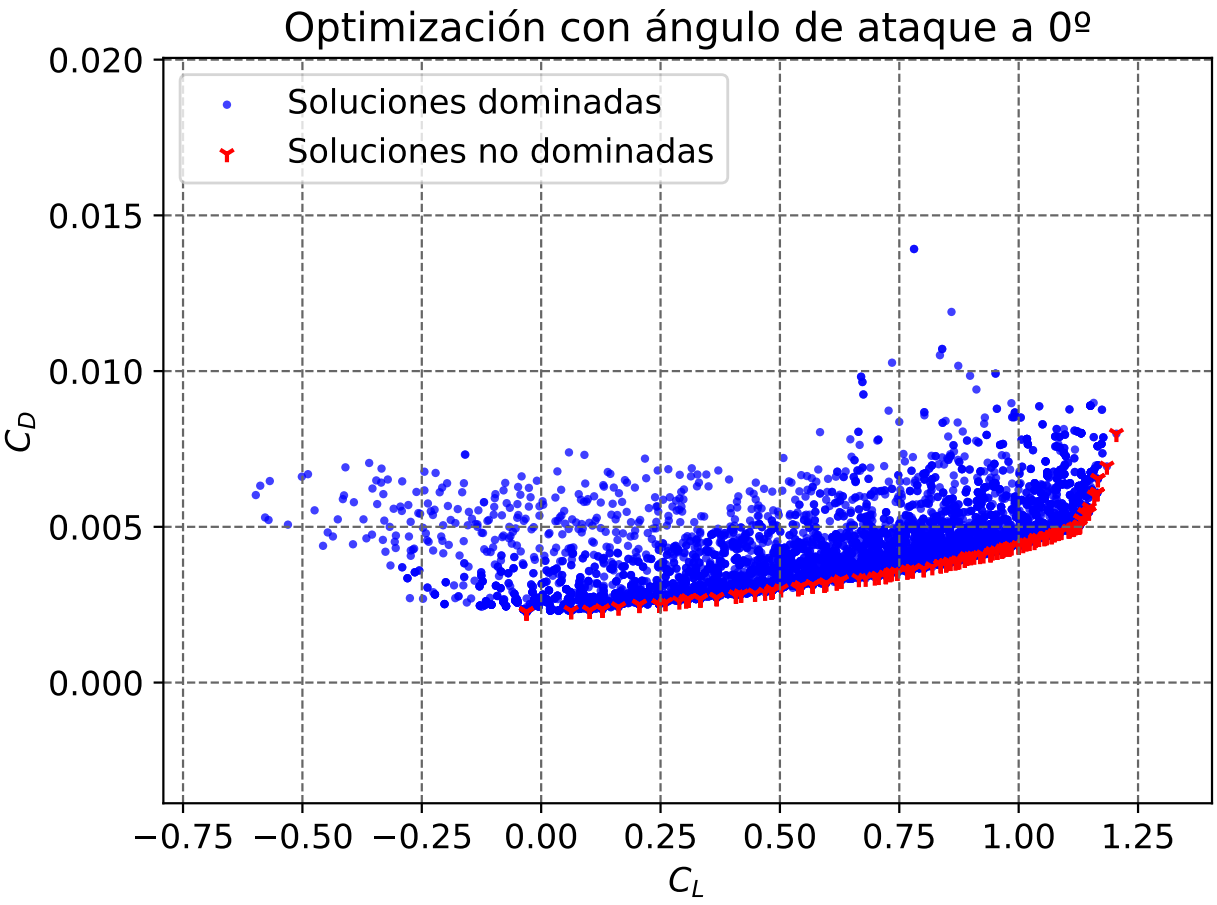
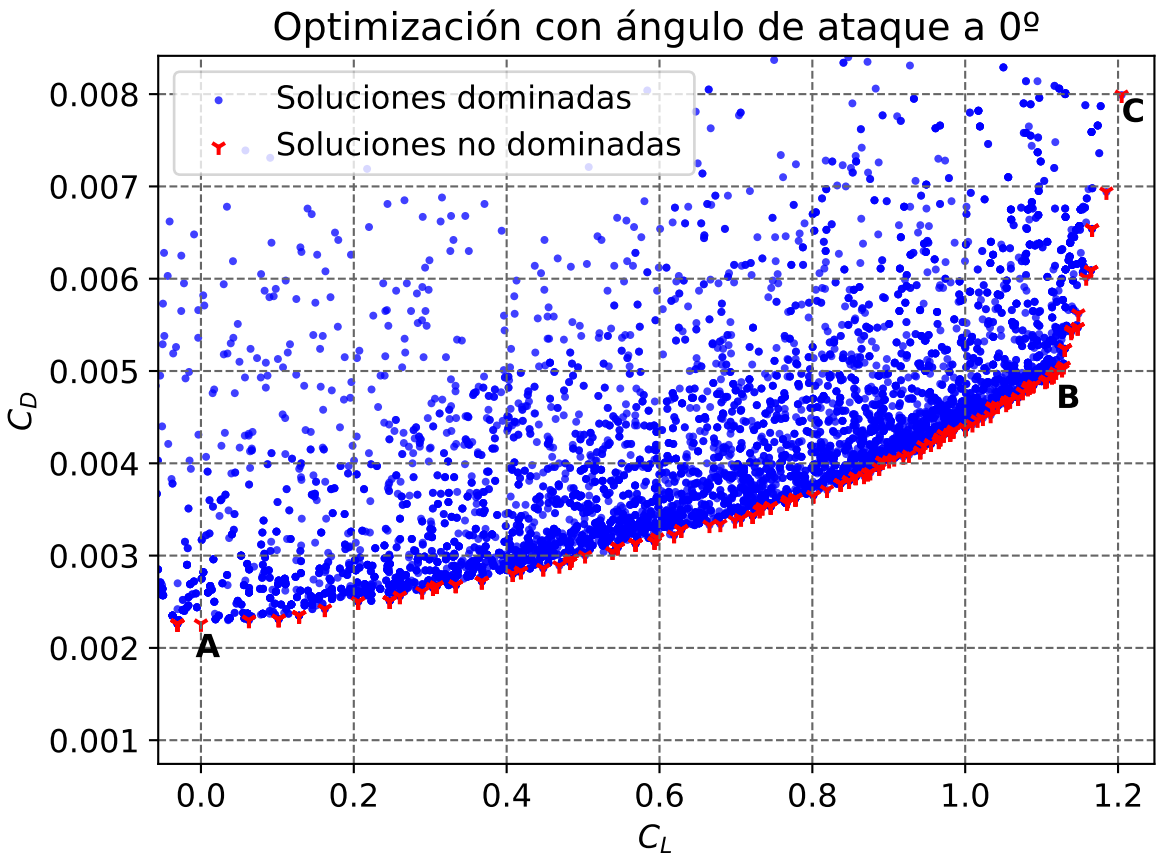
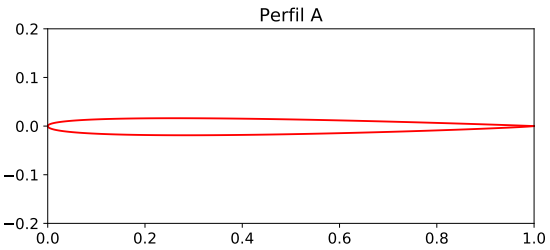


Figura 7.4: Espacio de diseño para optimización con ángulo de ataque a 0º

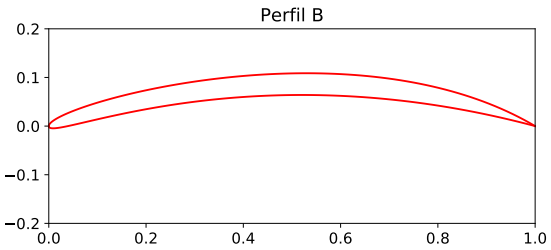
Figura 7.5: Frente de Pareto junto a perfiles seleccionados del mismo. AoA: 0º



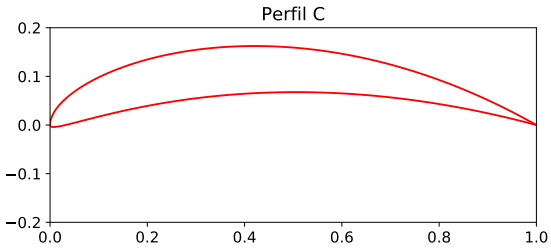
(a) Frente de Pareto para ángulo de ataque de 0º



(b) Perfil A: C_L : 0.0000, C_D : 0.00226



(c) Perfil B: C_L : 1.1191, C_D : 0.00499



(d) Perfil B: C_L : 1.2047, C_D : 0.00799

En la figura 7.5 se pueden observar 3 individuos de la población no dominada, así como el frente de Pareto con los puntos donde se ubican estas soluciones.

Se puede notar que la solución en la posición A (figura 7.5b) es un perfil simétrico, solución esperada para un perfil de cero sustentación debido a que el ángulo de ataque es 0º. Además,

este es delgado, condición necesaria para producir el arrastre menor posible. Por otro lado, el perfil de la solución B (figura 7.5c) muestra que, para poder iniciar el aumento de sustentación en un perfil con ángulo de ataque de 0º, es necesario el aumento de combadura del perfil, a cambio de un aumento natural en el coeficiente de arrastre. Por otra parte, la solución C (figura 7.5d) muestra el límite de combadura al que se tiende a llegar. Si la combadura llegase a un valor mayor al obtenido, el flujo se comenzaría a desprender en la superficie superior del perfil; esto provocaría un aumento significativo en el coeficiente de arrastre a cambio de un aumento casi nulo sobre el coeficiente de sustentación. Este perfil representa un caso donde se desea tener mayor sustentación, a expensas de un alto coeficiente de arrastre. Cabe notar que un coeficiente de 1.2047 es un alto coeficiente de sustentación para un ángulo de ataque de 0º. Este frente de Pareto muestra el resultado clásico en la teoría aerodinámica; siempre que se aumenta de manera inteligente la sustentación en un perfil, el arrastre aumentará. Este relación sustentación-arrastre no siempre es lineal, lo cual hace este problema aún más interesante.

7.3.2. Optimización con ángulo de ataque a 6º

Luego de las evaluaciones del problema para ángulo de ataque de 6º, se encontraron 2244 soluciones no dominadas. Aquí, el frente está visiblemente menos esparcido sobre el espacio de diseño, como se puede ver en la figura 7.6. La mayoría de soluciones no dominadas se encuentran en el rango de $[1.69, 1.7377]$.

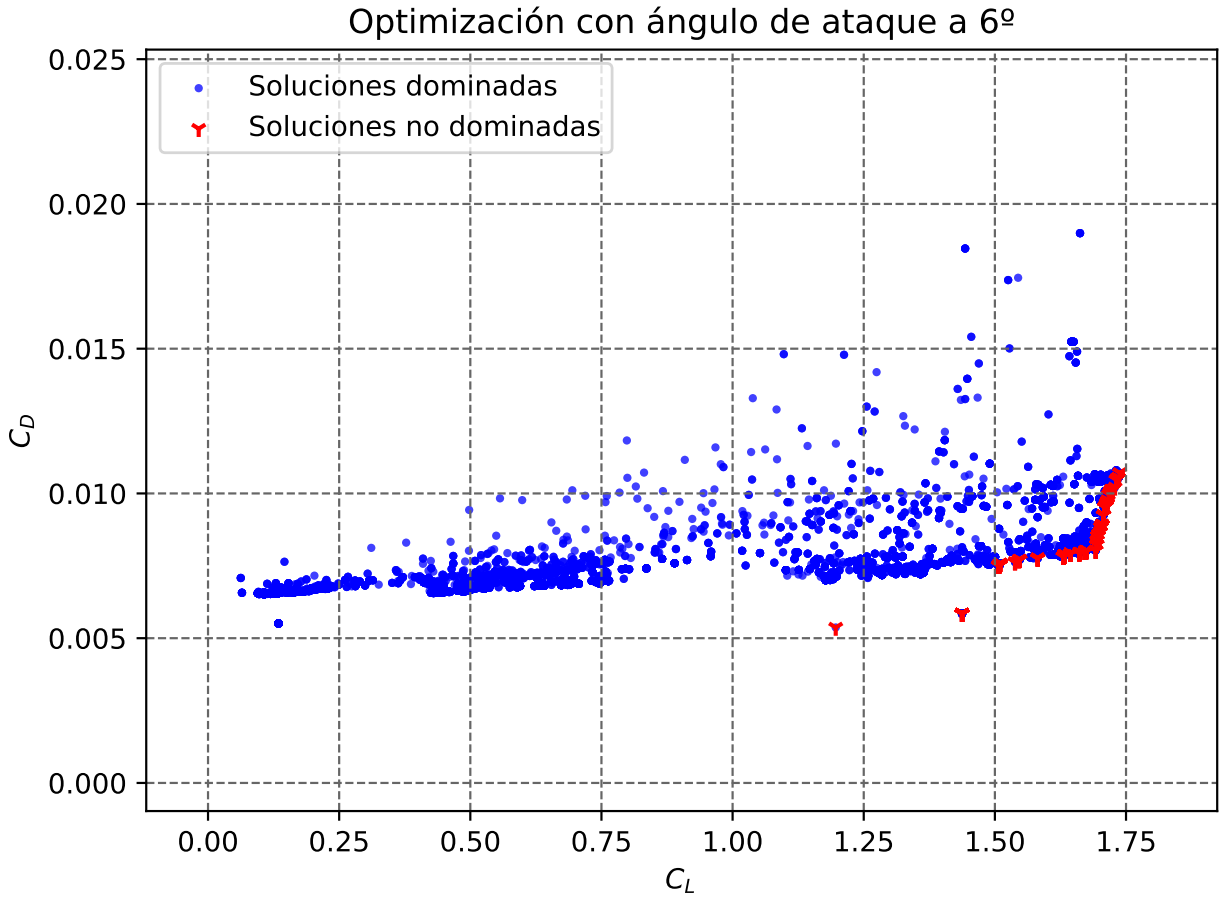


Figura 7.6: Espacio de diseño para optimización con ángulo de ataque a 6º

En la figura 7.8 se pueden observar 3 individuos de la población no dominada, así como el frente de Pareto con los puntos donde se ubican estas soluciones. En la tabla 7.2 se encuentran los valores de los parámetros que forman las curvas de los perfiles.

De nuevo, en este frente de Pareto, se nota la tendencia de aumento de arrastre a medida que se aumenta la sustentación; sumando a esto el efecto de añadir combadura a la geometría, lo cual induce arrastre naturalmente. Por otro lado, el perfil A (figura 7.8a) confirma que para minimizar el arrastre en un perfil a ángulos de ataque modestos ($\sim 6^o$) no se utilizan perfiles simétricos. Estos son reemplazados por perfiles con combadura, pero que tiene una geometría plana en la superficie inferior del perfil. Esta es la combinación generalmente usada para bajo ángulo de ataque, y mayor reducción del arrastre. Finalmente, el perfil B y C (figuras 7.8b, 7.8c) muestran que para aumentar la sustentación es necesario el aumento en la combadura del perfil, con un costo de aumento en el arrastre.

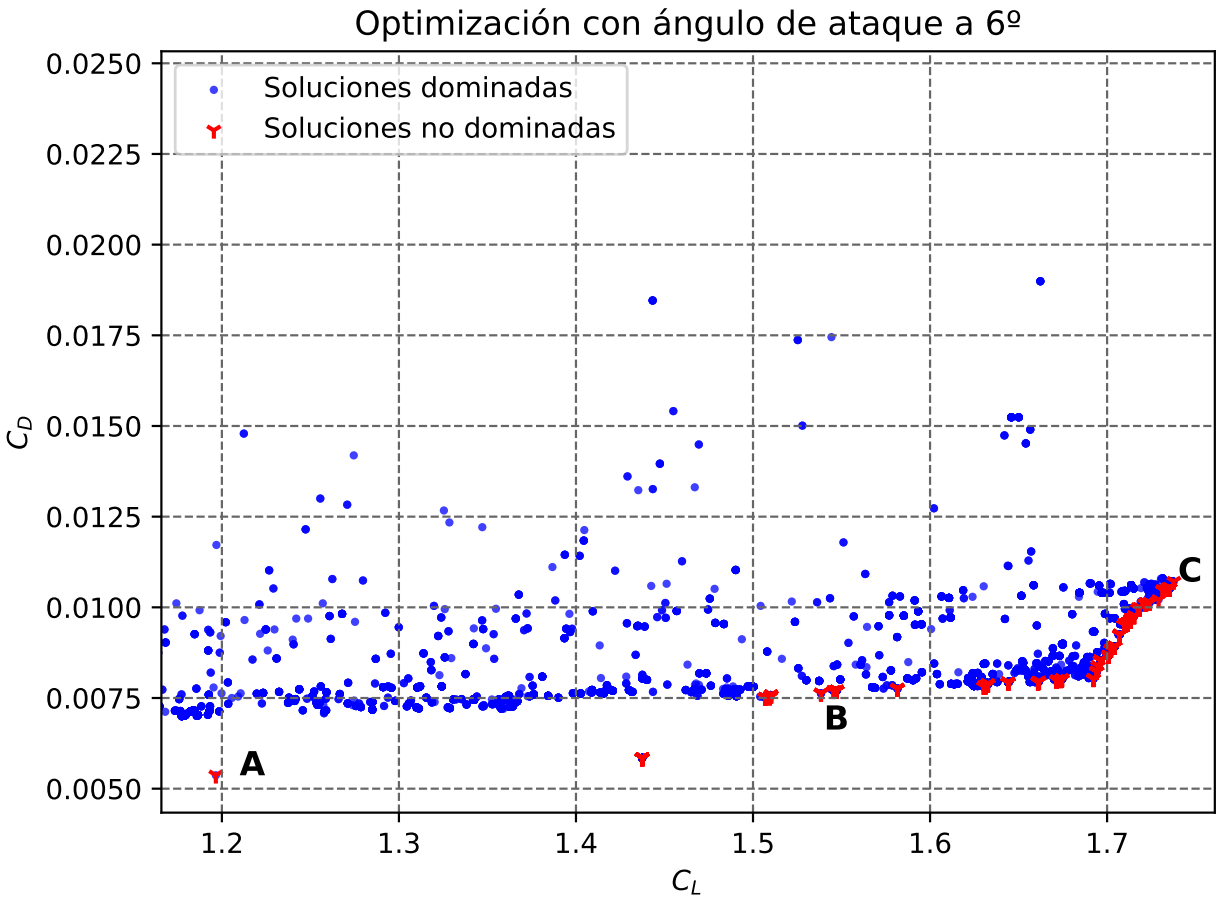
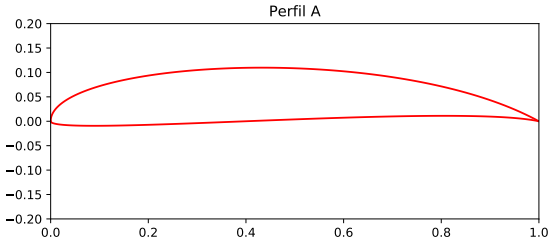
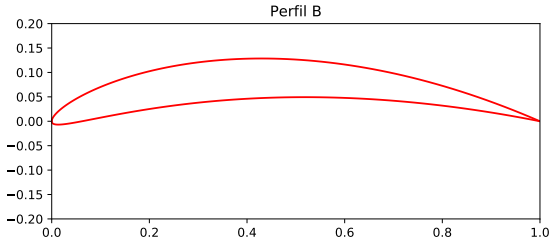


Figura 7.7: Frente de Pareto para ángulo de ataque de 6º

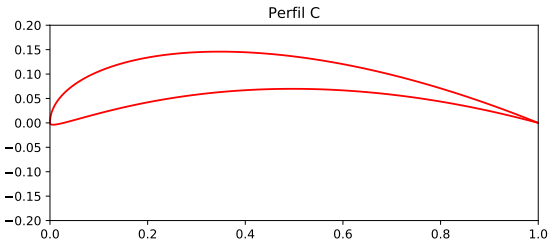
Figura 7.8: Seleccion de perfiles para optimización con AoA: 6º



(a) Perfil A: C_L : 1.1966, C_D : 0.00537



(b) Perfil B: C_L : 1.5384, C_D : 0.00763



(c) Perfil B: C_L : 1.7377, C_D : 0.01071

Tabla 7.2: Parámetros de generación de curvas de cada uno de los perfiles escogidos

Parámetro	Perfil A - AoA 0º	Perfil B - AoA 0º	Perfil C - AoA 0º	Perfil A - AoA 6º	Perfil B - AoA 6º	Perfil C - AoA 6º
$\mathbf{T}_A^{\text{upper}}$	0.105409366651556	0.129186840708512	0.31937152519031003	0.3716572996874833	0.13962720182698604	0.3954328393449785
$\mathbf{T}_A^{\text{lower}}$	0.11703052394598852	0.10889381267181782	0.10385049749271366	0.10049635762489759	0.126613041489691	0.10028822026802077
$\mathbf{T}_B^{\text{upper}}$	0.18140100009992322	1.303974452335608	1.859350670101537	1.0438067104396618	1.9948694224791705	1.9984463720997552
$\mathbf{T}_B^{\text{lower}}$	0.1713219981799387	1.8736619800238017	1.9538093714286549	0.534751804378726	1.99984875303454	1.999911987755029
α_b	8.808335358696924	15.712163404656364	14.508778175945919	16.693634956834632	12.20473413707831	6.921863676946719
α_c	-6.42558974354767	14.902834893934788	14.95922222446378	12.033506074059748	11.223811238127094	14.999528546882088

Capítulo 8

Conclusiones

En este trabajo se demostró cómo se puede implementar un mapa auto-organizado en un algoritmo evolutivo multi-objetivo. Esto es posible gracias a el frente de Pareto es un espacio topológico de dimensión $(m-1)$ para un problema multi-objetivo de m objetivos. Un mapa auto-organizado con una estructura regular de $(m-1)$ dimensiones permite la aproximación del frente de Pareto mediante el mapeo de grupo de datos de altas dimensiones (variables de decisión) en los elementos del mapa en un proceso de aprendizaje no supervisado. Este algoritmo propuesto fue implementado en el lenguaje de programación python y fue probado contra algoritmos de vanguardia en bancos de prueba estándar. A pesar de tener dificultades con frentes de Pareto complicados como lo son WFG1, WFG2, WFG4, el algoritmo propuesto se desempeñó relativamente mejor que sus contrapartes en la mayoría de problemas. Aunque se realizaron estudios de sensibilidad de parámetros, dado que este tipo de algoritmo suele ser problema-dependiente, es necesario realizar aún más estudios para el ajuste del desempeño en problemas con frentes de Pareto aún más complicados o con mayor número de objetivos.

Por otro lado, para validar el algoritmo, este fue probado en un problema de diseño de interés del autor. Dado un número de Reynolds, un ángulo de ataque y un método de parametrización de perfiles aerodinámicos, se realizó la exploración del espacio de diseño. Realizando más de 9500 evaluaciones de perfiles generados por el algoritmo, se encontraron frente de Pareto con soluciones no dominadas, tal como se esperaba. De aquí, seis perfiles fueron tomados y graficados. Cabe notar que existen dificultades a la hora de calcular correctamente valores de arrastre por medio de simulaciones. Los resultados aquí mostrados deben ser tomados con precaución. Si bien el cálculo de arrastre es impreciso y debe ser verificado mediante pruebas en túnel de viento, esta aproximación sirva para realizar una exploración del espacio de diseño, con el objetivo de tener las herramientas para tomar una decisión de ingeniería a la hora de comenzar un diseño detallado de un perfil aerodinámico basado en un método de parametrización de curvas.

Existe mucho trabajo respecto a esta implementación que vale la pena realizar. Aumentar el número de bancos de pruebas; comparar con otros algoritmos de vanguardia (existen algoritmos basados en SOMs, una comparación directa sería muy interesante); realizar un ajuste mucho más preciso de los hiperparámetros para obtener mejores resultados; eliminar el costo computacional de re-entrenar la red cada generación.

Bibliografía

[1] Oscar Brito Augusto, Fouad Bennis y Stephane Caro. «Multiobjective engineering design optimization problems: a sensitivity analysis approach». En: *Pesquisa Operacional* 32.3 (dic. de 2012), págs. 575-596. ISSN: 1678-5142. DOI: 10.1590/S0101-74382012005000028. URL: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-74382012000300006&lng=en&tlng=en.

[2] Jasbir Singh Arora y Jasbir Singh Arora. «Multi-objective Optimum Design Concepts and Methods». En: *Introduction to Optimum Design* (ene. de 2017), págs. 771-794. DOI: 10.1016/B978-0-12-800806-5.00018-4. URL: https://www.sciencedirect.com/science/article/pii/B9780128008065000184.

[3] Martin Klammer y col. «Pareto Optimization Identifies Diverse Set of Phosphorylation Signatures Predicting Response to Treatment with Dasatinib». En: *PLOS ONE* 10.6 (jun. de 2015). Ed. por Andrew R. Dalby, e0128542. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0128542. URL: https://dx.plos.org/10.1371/journal.pone.0128542.

[4] Kalyanmoy Deb y col. «A fast and elitist multiobjective genetic algorithm: NSGA-II». En: *IEEE Transactions on Evolutionary Computation* (2002). ISSN: 1089778X. DOI: 10.1109/4235.996017.

[5] Eckart Zitzler, Marco Laumanns y Lothar Thiele. «SPEA2: Improving the Strength Pareto Evolutionary Algorithm». En: *Igarss 2014* (2014). ISSN: 0717-6163. DOI: 10.1007/s13398-014-0173-7.2.

[6] N. Hansen y A. Ostermeier. *Completely derandomized self-adaptation in evolution strategies.* 2001. DOI: 10.1162/106365601750190398.

[7] Trevor Hastie, Robert Tibshirani y J. H. (Jerome H.) Friedman. *The elements of statistical learning : data mining, inference, and prediction*, pág. 745. ISBN: 9780387848570.

[8] Dirk Büche y Michele Milano. *Self-Organizing Maps for Multi-Objective Optimization*. Inf. téc. URL: https://pdfs.semanticscholar.org/3dae/2577c83a13f34376dbaa3195dffa00a5748f.pdf.

[9] Dirk Büche y col. «Self-organizing Maps for Pareto Optimization of Airfoils». En: Springer, Berlin, Heidelberg, 2002, págs. 122-131. DOI: 10.1007/3-540-45712-7{_}12. URL: http://link.springer.com/10.1007/3-540-45712-7_12.

[10] S. Kan, Z. Fei y E. Kita. «Application of self-organizing maps to genetic algorithms». En: *WIT Transactions on The Built Environment*. Vol. 106. WIT Press, mayo de 2009, págs. 3-11. ISBN: 978-1-84564-185-6. DOI: 10.2495/OP090011. URL: http://library.witpress.com/viewpaper.asp?pcode=OP09-001-1.

[11] Teuvo Kohonen. *Self-Organizing Maps*. Vol. 30. Springer Series in Information Sciences. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. ISBN: 978-3-540-67921-9. DOI: 10.1007/978-3-642-56927-2. URL: http://link.springer.com/10.1007/978-3-642-56927-2.

[12] Bob Allen. «NACA Airfoils». En: (2017). URL: https://www.nasa.gov/image-feature/langley/100/naca-airfoils.

[13] Mark Drela. «XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils». En: 1989. DOI: 10.1007/978-3-642-84010-4{_}1.

[14] Kalyanmoy. Deb y Deb Kalyanmoy. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, pág. 497. ISBN: 047187339X. URL: https://dl.acm.org/citation.cfm?id=559152.

[15] Teuvo Kohonen. «The self-organizing map». En: *Neurocomputing* 21.1-3 (nov. de 1998), págs. 1-6. ISSN: 0925-2312. DOI: 10.1016/S0925-2312(98)00030-7. URL: https://www.sciencedirect.com/science/article/pii/S0925231298000307.

[16] Sunith Bandaru, Amos H.C. Ng y Kalyanmoy Deb. «Data mining methods for knowledge discovery in multi-objective optimization: Part B - New developments and applications». En: *Expert Systems with Applications* 70 (mar. de 2017), págs. 119-138. ISSN: 09574174. DOI: 10.1016/j.eswa.2016.10.016. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417416305474.

[17] Hu Zhang y col. «A Self-Organizing Multiobjective Evolutionary Algorithm». En: *IEEE Transactions on Evolutionary Computation* 20.5 (oct. de 2016), págs. 792-806. ISSN: 1089-778X. DOI: 10.1109/TEVC.2016.2521868. URL: http://ieeexplore.ieee.org/document/7393537/.

[18] Kalyanmoy Deb y col. «Scalable multi-objective optimization test problems». En: *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002*. 2002. DOI: 10.1109/CEC.2002.1007032.

[19] Simon Huband y col. *A review of multiobjective test problems and a scalable test problem toolkit*. 2006. DOI: 10.1109/TEVC.2005.861417.

[20] Aimin Zhou y col. «A Model-Based Evolutionary Algorithm for Bi-objective Optimization». En: *2005 IEEE Congress on Evolutionary Computation*. Vol. 3. IEEE, págs. 2568-2575. ISBN: 0-7803-9363-5. DOI: 10.1109/CEC.2005.1555016. URL: http://ieeexplore.ieee.org/document/1555016/.

[21] E. Zitzler y L. Thiele. «Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach». En: *IEEE Transactions on Evolutionary Computation* 3.4 (1999), págs. 257-271. ISSN: 1089778X. DOI: 10.1109/4235.797969. URL: http://ieeexplore.ieee.org/document/797969/.

[22] Qingfu Zhang y Hui Li. «MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition». En: *IEEE Transactions on Evolutionary Computation* 11.6 (dic. de 2007), págs. 712-731. ISSN: 1941-0026. DOI: 10.1109/TEVC.2007.892759. URL: http://ieeexplore.ieee.org/document/4358754/.

[23] Kalyanmoy Deb, Manikanth Mohan y Shikhar Mishra. «Towards a Quick Computation of Well-Spread Pareto-Optimal Solutions». En: Springer, Berlin, Heidelberg, 2003, págs. 222-236. DOI: 10.1007/3-540-36970-8{_}16. URL: http://link.springer.com/10.1007/3-540-36970-8_16.

[24] D. Anitha y col. «Air foil Shape Optimization Using Cfd and Parametrization Methods». En: *Materials Today: Proceedings*. 2018. DOI: 10.1016/j.matpr.2017.12.122.

[25] Wenbin Song y Andrew Keane. «A Study of Shape Parameterisation Methods for Airfoil Optimisation». En: 2012. DOI: 10.2514/6.2004-4482.

[26] Eric Besnard y col. «Two-dimensional aircraft high lift system design and optimization». En: *36th AIAA Aerospace Sciences Meeting and Exhibit*. Reston, Virigina: American Institute of Aeronautics y Astronautics, ene. de 1998. DOI: 10.2514/6.1998-123. URL: http://arc.aiaa.org/doi/10.2514/6.1998-123.

[27] Andras Sobester y Andy Keane. «Airfoil Design via Cubic Splines - Ferguson's Curves Revisited». En: *AIAA Infotech@Aerospace 2007 Conference and Exhibit*. Reston, Virigina: American Institute of Aeronautics y Astronautics, mayo de 2007. ISBN: 978-1-62410-017-8. DOI: 10.2514/6.2007-2881. URL: http://arc.aiaa.org/doi/10.2514/6.2007-2881.

[28] Andras Sobester y Tom Barrett. «Quest for a Truly Parsimonious Airfoil Parameterization Scheme». En: *The 26th Congress of ICAS and 8th AIAA ATIO*. Reston, Virigina: American Institute of Aeronautics y Astronautics, sep. de 2008. ISBN: 978-1-60086-997-6. DOI: 10.2514/6.2008-8879. URL: http://arc.aiaa.org/doi/10.2514/6.2008-8879.