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

Técnicas de Auto-Adaptación para Algoritmos Evolutivos Multi-objetivo

Tesis que presenta:

Edgar Gerardo Yáñez Oropeza

Para obtener el grado de

Maestro en Ciencias

de la Computación

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

México, D. F. Noviembre 2009

Abstract

Nowadays, the use of evolutionary computation (EC) techniques has become a very popular tool to solve a variety of real-world problems. Within multi-objective optimization problems, several EC-based proposals currently exist, and are very popular in the specialized literature. However, the different EC tools currently available have an important drawback: they require a careful fine-tuning of their parameters, and such fine-tuning is normally done in an empirical way, being different for each problem to be solved. Such empirical fine-tuning of parameters aim to balance the exploration and exploitation phases of the evolutionary algorithm being used, such that it achieves the best possible results.

The main goal of this thesis was to define a multi-objective evolutionary algorithm (MOEA) that does not require any user-defined parameters. In order to achieve such goal, it was necessary to define different techniques to self-adapt the parameters of a state-of-the-art MOEA, the NSGA-II. Such self-adaptation techniques allow that the MOEA defines, by itself, and during its execution, the most appropriate values for its most important parameters.

In order to assess the performance of the proposed approach, its behavior was evaluated using twelve test problems taken from the specialized literature. The results obtained by the proposed approach were compared with respect to those produced by the original NSGA-II, concluding that the proposed approach is a viable alternative to perform parameterless evolutionary multi-objective optimization.

Resumen

En la actualidad, el uso del cómputo evolutivo se ha vuelto muy popular como herramienta para dar soluciones a diferentes problemas del mundo real. En el ámbito de los problemas multi-objetivo, existen diversas propuestas basadas en cómputo evolutivo que son muy populares en la literatura especializada. Sin embargo, las diferentes herramientas propuestas en el campo del cómputo evolutivo multi-objetivo tienen una importante desventaja: requieren un ajuste cuidadoso de sus parámetros, el cual suele realizarse de forma empírica, siendo además distinto para cada problema a resolverse. Dicho ajuste empírico de parámetros busca balancear la fase de exploración con la de explotación del algoritmo evolutivo utilizado, a fin de que éste logre los mejores resultados posibles.

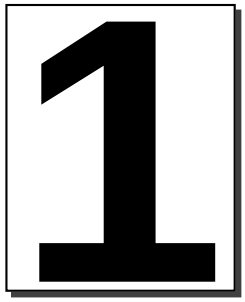
El objetivo principal de esta tesis fue definir un algoritmo evolutivo multi-objetivo que no requiriera parámetros a ser definidos por el usuario. Para alcanzar dicho objetivo, fue necesario definir diferentes técnicas de auto-adaptación de parámetros para un algoritmo evolutivo multi-objetivo del estado del arte, el NSGA-II. Dichas técnicas de auto-adaptación permiten que el algoritmo evolutivo defina, por sí mismo y durante su ejecución, los valores más adecuados de sus parámetros más importantes.

Para analizar el desempeño del esquema propuesto, se evaluó su comportamiento usando doce problemas de prueba, tomados de la literatura especializada. Los resultados obtenidos por el esquema propuesto se compararon con respecto a los del NSGA-II original, concluyéndose que la propuesta es una alternativa viable para realizar optimización evolutiva multi-objetivo libre de parámetros.

Índice general

Índice de figuras

Índice de tablas



Introducción

En las diferentes áreas de la ingeniería, la arquitectura, la economía, la física, la química, etc. se cuenta con problemas de optimización sumamente complejos (alta dimensionalidad, discontinuidad, multimodalidad), donde las técnicas clásicas [63] no han resultado del todo efectivas. Es por ello que surge el cómputo evolutivo como un método alternativo de optimización.

Los algoritmos evolutivos multi-objetivo (AEMOs) han mostrado ser sumamente efectivos en varios de los diferentes problemas del mundo real. Sin embargo, requieren un conocimiento previo para establecer los parámetros iniciales y éstos se deben definir de manera empírica. Lo que le pretendemos, en esta tesis, es obtener un AEMO más sencillo en cuanto a su uso, evitándole al usuario la compleja tarea de establecer manualmente sus parámetros.

En esta tesis se busca proponer diferentes técnicas de auto-adaptación que permitan a un AEMO prescindir de los parámetros iniciales que deben ser establecidos por el usuario antes de aplicar el algoritmo a un problema en específico.

1.1 Antecedentes y motivación

El uso del cómputo evolutivo como herramienta para resolver problemas de optimización se ha vuelto muy popular. Sin embargo, no existe una propuesta capaz de eliminar totalmente la carga que representa la definición de sus diferentes parámetros. Existen varias propuestas como el micro–GA2 (μGA^2) [89], el Incrementing Multiobjetive Evolutionary Algorithm (IMOEA [87]) y el Self-Adaptive Pareto Differential Evolution (SPDE [1]), entre otras.

En cuanto a los algoritmos evolutivos multi-objetivo tenemos al Nondominated Sorting Genetic Algorithm II (NSGA-II) [24] que se ha mostrado muy competitivo con respecto a otros algoritmos evolutivos multi-objetivo modernos tales como la Pareto Archived Evolution Strategy (PAES) [47] y el Strength Pareto Evolutionary Algorithm (SPEA) [101].

El NSGA-II es un algoritmo muy rápido y efectivo. La cantidad de parámetros que requiere es menor que la requerida por otros algoritmos del estado del arte, por lo cual cuenta con una gran popularidad entre los investigadores y se puede considerar el algoritmo a vencer cuando se propone un nuevo AEMO.

Para lograr que un AEMO implemente diferentes operadores que realicen una búsqueda efectiva, éstos deben ajustarse de manera adecuada, a fin de tener un balance entre las dos fases de un algoritmo de optimización: la explotación y la exploración. Una buena exploración del espacio de búsqueda impide converger a óptimos locales. Por otro lado, el tener un mayor poder explotativo ayudará a encontrar buenas soluciones dentro de una cierta zona promisoria del espacio de búsqueda.

Un algoritmo evolutivo puede comportarse de manera muy diferente cuando los parámetros son diferentes de una ejecución a otra, aunque los cambios realizados hayan sido muy pequeños. Además, conforme se va avanzando en las generaciones, es conveniente utilizar otros parámetros, a fin de poder realizar una búsqueda más fina.

Para realizar ajustes en los parámetros de un algoritmo evolutivo es sumamente recomendable tener conocimiento de éstos y la manera en que afectan

el desempeño del algoritmo. Es por ello que a muchas personas no especializadas en el área les cuesta trabajo utilizar de manera eficaz un algoritmo evolutivo, y frecuentemente se requieren análisis empíricos de sus parámetros para cada problema en particular.

Todo esto motivó la idea de contar con un algoritmo evolutivo multiobjetivo con mecanismos de auto-adaptación, con el cual el usuario se despreocupe y no tenga que intervenir en el proceso de ajustar o brindar un conjunto de parámetros iniciales que se requieran para resolver su problema.

1.2 Planteamiento del problema

Para poder utilizar un algoritmo evolutivo es necesario definir varios parámetros:

- Tamaño de población
- Número máximo de generaciones
- Probabilidad de cruza
- Probabilidad de mutación
- Tipo de cruza
- Tipo de mutación
- Tipo de representación
- Funciones objetivo
- Número de variables
- Límite de las variables

Los últimos tres son parámetros que dependen de la naturaleza del problema y por ende no pueden ser auto-adaptados. Sin embargo, los restantes son parámetros que deben ser definidios empíricamente, y su ajuste dista de ser trivial. El problema es desarrollar e implementar mecanismos que auto-adapten los diferentes parámetros de un AEMO y lo mantengan competitivo

evitándose la intervención del usuario.

Aunque se han realizado varias investigaciones acerca del comportamiento de los algoritmos evolutivos, no se tiene una idea muy clara de la forma en que sus parámetros se correlacionan con su desempeño, ya que éstos interactúan de manera altamente no lineal.

La tarea de encontrar un conjunto de parámetros que proporcionen a un AEMO un buen desempeño en un problema arbitrario no es algo sencillo y de ahí la relevancia de este trabajo.

1.3 Propuesta

En esta tesis se presenta un conjunto de técnicas que fueron implementadas en el algoritmo NSGA-II [24], las cuales evitan la definición por parte del usuario de todos los parámetros que no son parte de la naturaleza del problema. El algoritmo resultante se muestra bastante competitivo con respecto al NSGA-II original y, en algunos casos, logra incluso superarlo.

El algoritmo propuesto se describe a detalle capítulo 5. El diseño experimental y el análisis de resultados de la propuesta se detalla en el capítulo 6.

1.4 Objetivos

Diseñar mecanismos que permitan a un algoritmo evolutivo multi-objetivo auto-adaptar los diferentes parámetros con los que éste cuenta, evitando al usuario lidiar con la difícil tarea de ajustar manualmente sus parámetros iniciales, v.g., se busca contar con un algoritmo que pueda ajustar automáticamente sus parámetros sin la intervención del usuario.

1.5 Contribuciones

Lo que se pretende obtener son mecanismos de auto-adaptación que puedan ser implementados en un algoritmo evolutivo multi-objetivo, de forma

que el usuario no tenga que ser partícipe de la definición o ajuste de sus parámetros, y sólo se dedique a delimitar su problema y aplicar el algoritmo que lo resolverá.

Se llevó a cabo un análisis de sensibilidad del NSGA-II a sus parámetros principales a fin de determinar cuáles afectaban más su desempeño. Esto se detalla en el capítulo 4.

También se buscó definir un criterio de paro, que indique el momento en que se deben detener las iteraciones, evitando al usuario la definición del número máximo de generaciones.

1.6 Estructura de la tesis

La forma en que está organizada la presente tesis es la siguiente:

- El capítulo 2 muestra los antecedentes biológicos y computacionales de los algoritmos evolutivos. Además, describe de forma breve los paradigmas más relevantes dentro de la computación evolutiva.

- El capítulo 3 contiene los antecedentes sobre la definición, adaptación y auto-adaptación de parámetros en algoritmos evolutivos y muestra el estado del arte específicamente en el área de auto-adaptación en algoritmos evolutivos multi-objetivo.

- En el capítulo 4 se describen los parámetros sobre los que se va a trabajar, y se presenta el análisis de sensibilidad del NSGA-II a sus parámetros.

- En el capítulo 5 se explica a detalle el algoritmo propuesto.

- En el capítulo 6 se presenta el diseño experimental y el análisis de los resultados arrojados de la validación de nuestra propuesta.

- El capítulo 7 contiene las conclusiones desprendidas de la presente tesis así como algunas rutas posibles de trabajo futuro.

Adicionalmente, se proporcionan varios apéndices organizados de la siguiente forma:

- Apéndice A: contiene la definición y características de cada una de las funciones de prueba utilizadas para analizar el desempeño de la propuesta.
- Apéndice B: muestra de forma gráfica los diferentes frentes verdaderos de Pareto para cada una de las funciones de prueba utilizadas.
- Apéndice C: muestra las medidas de desempeño con las cuales se realizó la validación de resultados.
- Apéndice D: contiene la definición de los diferentes tipos de cruza y mutación adoptadas en el algoritmo propuesto.

2

Cómputo Evolutivo Multi-Objetivo

El cómputo evolutivo está inspirado principalmente en la teoría Neo-Darwiniana y en un conjunto de procesos biológicos presentes en la naturaleza. El cómputo evolutivo realiza una simulación de estos procesos para aplicarlos en la solución de problemas de optimización.

En este capítulo se presentan los antecedentes biológicos e históricos de la teoría Neo-Darwiniana, también se presenta la forma en que el cómputo evolutivo adopta estas ideas y, por último, se explican los paradigmas principales que de él se desprenden.

2.1 Antecedentes biológicos e históricos

El paradigma conocido como Neo-Darwinismo es la fusión de la teoría de la evolución propuesta por Darwin en 1859, la teoría genética descrita por Mendel en 1865 y la teoría del plasma germinal presentada por Weismann en 1883. En la actualidad, el pensamiento evolutivo está basado en el paradigma del Neo-Darwinismo, que a grandes rasgos nos señala que las distintas formas

de vida existentes son el resultado de cuatro procesos estadísticos que se efectúan sobre las especies. A continuación, se proporcionan más detalles de estos procesos estadísticos y de las teorías que los sustentan: la reproducción, la mutación, la competencia y la selección.

2.1.1 Teoría de la evolución

El naturalista británico Charles Robert Darwin (figura 2.1) publicó en 1859 su obra titulada *El origen de las especies por medio de la selección natural o la preservación de las razas favorecidas en su lucha por la vida* [18], donde gracias a sus ideas de la selección natural, sentó las bases de lo que hoy en día conocemos como evolución y los conceptos que la conforman.

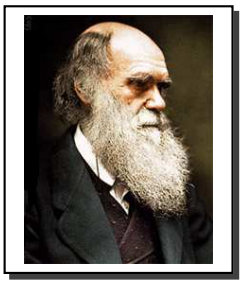


Figura 2.1: Charles Robert Darwin (1806-1882)

La teoría de la evolución moderna nos señala que los seres y las especies existentes en la actualidad son descendientes de otras diferentes que existieron en el pasado, y que se produjeron mediante un conjunto de mecanismos que generan nuevos individuos similares pero con algunas modificaciones, donde el proceso natural más relevante es la selección y por medio del principio de herencia es como se transmiten las características físicas a su descendencia.

Darwin señala que los nuevos individuos con características que los favorezcan en su ambiente tendrán una mayor probabilidad de ser preservados. A este principio de preservación lo llamó selección natural y en resumen dice:

- En cualquier generación existen individuos que no logran reproducirse.
- Los individuos de una población o los miembros de una especia difieren en sus características, es decir, tienen modificaciones entre uno y otro.
- La capacidad de tener una mayor descendencia está asociada con las características heredadas.

- Las variaciones de cada individuo son, en su mayoría, heredadas de los padres.

2.1.2 Teoría genética

El monje austríaco Gregor Johann Mendel (figura 2.2) publicó en 1865 una obra titulada *Experimentos de Hibridación en Plantas* [60]. En esta obra Mendel plantea, desarrolla y muestra sus conclusiones en los experimentos que realizó con chícharos. Como resultado de sus experimentos demostró que la herencia se transmite por elementos particulares (refutando la herencia de las mezclas) y que además siguen normas estadísticas.

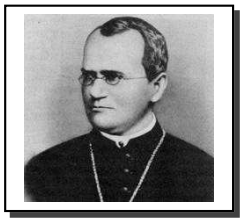


Figura 2.2: Gregor Johann Mendel (1822-1884)

Además, como otro fruto de su investigación, estableció tres leyes que hoy en día son fundamentales en el campo de la genética, las cuales son:

- **Ley de la Independencia**: los pares de alelos se separan durante la formación de gametos.
- **Ley de la Segregación**: los genes recibidos de los padres se separan durante la producción de los gametos.
- **Ley de la Uniformidad**: las características heredadas son determinadas mediante dos factores provenientes de los padres, lo cual decide si un gene es dominante o recesivo.

2.1.3 Teoría del plasma germinal

El biólogo alemán austríaco August Friedrich Leopold Weismann (figura 2.3) publicó en 1893 una obra titulada *Plasma Germinal: Una Teoría de la Herencia* [60] donde establece la teoría de la continuidad del plasma germinal. El autor indica que los individuos están formados por células germinales

(germoplasma) y por células somáticas (somatoplasma); las primeras son una porción inmortal del individuo y contienen la información a heredar, mientras que las segundas forman el cuerpo y las funciones del individuo.

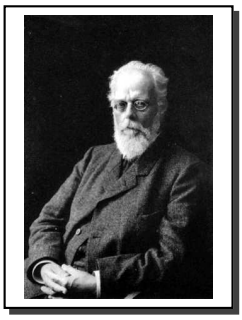


Figura 2.3: August Friedrich Leopold Weismann (1834-1914)

El germoplasma no puede ser alterado o modificado durante la existencia de un individuo; es decir, si el individuo aprende nuevas habilidades (lo que modifica el somatoplasma) estos conocimientos no serán transmitidos, ya que el germoplasma no los contempla.

2.1.4 Neo-Darwinismo

Se conoce como Neo-Darwinismo al conjunto de teorías presentada por Darwin, Mendel y Weismann en torno al origen de las especies. Este paradigma dicta que la enorme diversidad de especies que encontramos en el planeta se puede explicar mediante sólo cuatro procesos estadísticos que se aplican sobre las especies, los cuales son:

- **Competencia**: Es el proceso en el cual los individuos tienen una lucha constante por subsistir y reproducirse, heredando su código genético. Este proceso se da por el exceso poblacional de una especia que afecta a los individuos menos aptos al realizarse una selección estocástica.
- **Mutación**: Es el proceso donde se realiza una modificación de la información genética en el momento que se lleva a cabo la copia de ésta en el proceso de reproducción. Algunas mutaciones pueden resultar benéficas, esto significa que el cambio le otorgará al individuo una mejor adaptación al ambiente. Estas modificaciones pueden ser heredadas a la siguiente generación.

- **Reproducción**: Es el proceso por el cual se asegura que la información genética de un individuo sea parte de la siguiente generación. Existe la reproducción sexual que requiere la intervención de dos individuos, que generarán individuos con información genética diferente entre ellos. Y existe la reproducción asexual, donde un solo individuo es capaz de generar nuevos individuos con réplicas de su información genética, resultando en individuos iguales entre ellos.

- **Selección**: Es el proceso en el cual los individuos que, gracias a sus características peculiares son más aptos que otros individuos, tendrán una mayor oportunidad de supervivencia y reproducción.

2.1.5 Conceptos biológicos

Al tener el cómputo evolutivo una estrecha relación con el Neo-Darwinismo, la genética y la biología, se hace necesario presentar conceptos de gran utilidad para una mejor comprensión del tema.

- **Gene**: es una sección de ácido desoxirribonucleico (ADN) que es esencial para llevar a cabo cierta función bioquímica definida. Es la unidad fundamental de herencia (figura 2.4).

- **ADN**: es el componente básico del material genético (figura 2.4).

- **Cromosoma**: es una de las cadenas de ADN, y está formado por genes (figura 2.4).

- **Alelo**: se denomina así a las formas alternativas que puede presentar un gene dentro de una posición específica del cromosoma.

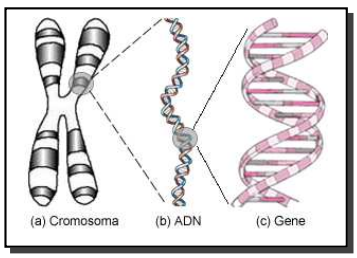


Figura 2.4: Relación cromosoma-ADN-gene

- **Genoma**: es el conjunto completo de genes que forman al individuo, es decir, el conjunto total de cromosomas.
- **Genotipo**: es la información genética del individuo, no observable a simple vista (figura 2.5).
- **Fenotipo**: es el resultado de decodificar al genotipo, mostrando sus rasgos observables a simple vista (figura 2.5).

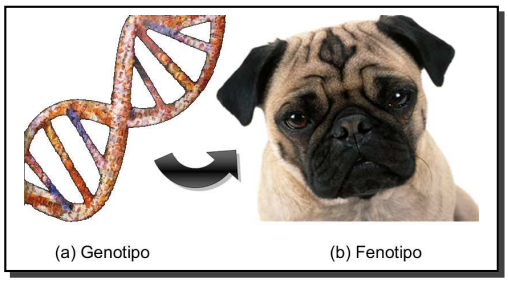


Figura 2.5: Relación genotipo-fenotipo.

- **Individuo**: se le denomina así a un solo individuo de una población (figura 2.6).
- **Población**: un conjunto de individuos que pueden interactuar juntos, particularmente con fines reproductivos (figura 2.6).

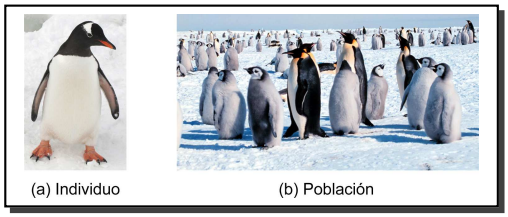


Figura 2.6: Ejemplo de individuo y población.

- **Aptitud**: la aptitud de un individuo se define como la probabilidad de éste para sobrevivir y reproducirse, o como una función del número de descendientes que tiene.
- **Ambiente**: es todo aquello que rodea al organismo, y que condiciona su evolución.

2.2 Cómputo evolutivo

La computación evolutiva comprende un conjunto de técnicas de búsqueda y optimización que simulan el proceso de la evolución de un conjunto de individuos (las soluciones potenciales a un problema) sujetos a selección, reproducción y mutación.

Para poder implementar con éxito un algoritmo evolutivo es necesario contar con los elementos siguientes [61]:

- Tener una forma de codificar las estructuras que se replicarán (representar las soluciones al problema)
- Operadores que actúen sobre los individuos
- Una función de aptitud, que juegue el papel del ambiente
- Un mecanismo de selección

El cómputo evolutivo en la actualidad ha tenido gran éxito, principalmente porque cuenta con ciertas características que lo hacen una buena alternativa en comparación con otras técnicas de optimización. Entre estas características destacan las siguientes:

- No requiere conocimientos específicos del problema, resolviendo en ocasiones problemas para los cuales no se conocía solución alguna
- Uso de un conjunto de soluciones en vez de utilizar solamente una, lo cual hace al algoritmo menos susceptible a quedar atrapado en mínimos o máximos locales
- Su simplicidad conceptual y amplia aplicabilidad
- Fácilmente adaptable a arquitecturas en paralelo
- Utilizan operadores probabilísticos, en comparación con las técnicas tradicionales que utilizan operadores determinísticos
- Pueden hibridizarse con otras técnicas de optimización o búsqueda
- Son robustos a los cambios dinámicos
- Generalmente pueden auto-adaptar sus parámetros

Dentro del campo de la computación evolutiva se pueden encontrar principalmente tres paradigmas que la conforman:

- Algoritmos genéticos
- Estrategias evolutivas
- Programación evolutiva

El pseudo-código de un algoritmo evolutivo se muestra en el Algoritmo 1. Básicamente así es como funciona un algoritmo evolutivo, aunque los diferentes paradigmas realizan modificaciones o adecuaciones específicas. Más adelante se mostrará el funcionamiento de los diferentes paradigmas y sus peculiaridades.

Algoritmo 1 Esquema general de un algoritmo evolutivo

```
1: Generar los individuos de una población P de manera aleatoria;
2: Se evalúa en la función objetivo a cada individuo;
3: repeat
4:     Seleccionar a los padres;
5:     Cruzar padres;
6:     Mutar individuos nuevos (hijos);
7:     Evaluar hijos;
8:     Aplicar elitismo;
9: until La condición de paro se satisfaga
```

2.2.1 Elementos de un algoritmo evolutivo

En esta sección se definirán los componentes más representativos del algoritmo evolutivo, que surgen de su contraparte genética. Estos componentes son:

- Representación
- Población
- Función objetivo
- Mecanismo de selección

- Operador de cruza
- Operador de mutación
- Elitismo

2.2.1.1 Representación

Antes de comenzar a tratar el problema, debemos encontrar una manera de representar las variables de tal forma que podamos constituir cromosomas que serán los que manipule el algoritmo evolutivo. Un cromosoma consta de varios genes, cada uno de los cuales corresponde a una de las variables de decisión (variables del problema planteado). La representación será nuestro genotipo, mientras que la decodificación del genotipo nos arrojará el valor de los parámetros de entrada usados en la función objetivo. Esto es conocido como el fenotipo, como se muestra gráficamente en la figura 2.7).

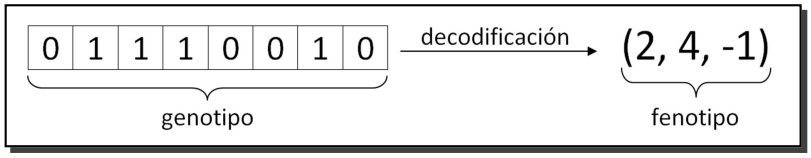


Figura 2.7: Representación del genotipo y fenotipo.

Las codificaciones más usuales son la binaria, la real y la entera; la elección de alguna depende del tipo de problema que se esté enfrentado. En la figura 2.8 se muestra un ejemplo de las tres diferentes codificaciones listadas con anterioridad.

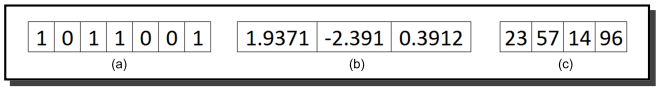


Figura 2.8: Representación binaria (a), real (b) y entera (c).

2.2.1.2 Población

Un individuo se define como una solución potencial al problema planteado, el cual contiene un cromosoma donde se codifican los valores de la solución potencial. La población será un conjunto de individuos, donde la

aptitud de cada uno corresponderá a la calidad de la solución, es decir, al ser evaluado el individuo en la función objetivo y comparado con los demás individuos se sabrá qué tanto es mejor o peor que los demás individuos de la población.

Los algoritmos evolutivos cuentan con un tamaño de población fijo. Sin embargo, existen propuestas que utilizan tamaños de población variable [87]. Además, la población de hijos puede reemplazar totalmente a la de los padres o se puede realizar la unión de ambas poblaciones y elegir a sólo a los mejores con el objetivo de tener un tamaño de población fijo de manera que no se vaya incrementando con el paso de las generaciones.

2.2.1.3 Función objetivo

La función objetivo es la que queremos minimizar o maximizar, o sea, la función a optimizar. Así, la función objetivo tomará el papel del ambiente, que nos dirá qué individuos son más aptos, y éstos tendrán mayor probabilidad de sobrevivir y tener descendencia. El valor que es arrojado por la función objetivo define la aptitud de un individuo.

2.2.1.4 Mecanismos de selección

El objetivo de la selección es obtener un conjunto de padres que participen en el proceso de reproducción para crear la nueva generación, es decir, para la creación de los hijos. La selección de los padres se basa en la aptitud de cada uno de ellos y en la calidad de ésta con respecto a la de los demás individuos (los otros padres). Así, lo que se pretende es que los mejores padres tengan una mayor probabilidad de ser seleccionados (como nos indica la selección natural). El objetivo es agregar presión para mejorar la calidad de los individuos, es decir, que la siguiente generación (los hijos) superen a la generación actual (los padres).

Entre las diferentes propuestas que existen en la literatura para realizar este proceso, se pueden distinguir las siguientes:

- **Selección proporcional** [40]: se eligen a los individuos de acuerdo a la contribución de aptitud que tengan en comparación con la aptitud total de la población. Algunas propuestas de este tipo de selección son: la ruleta [25], muestreo determinístico [25] y el sobrante estocástico [7, 9].

- **Selección mediante torneo** [94]: estos métodos se basan en la comparación directa entre individuos, ya sean dos o más participantes en cada torneo. Esto se realiza tomando una muestra aleatoria de individuos, la cual representa a los participantes del torneo, donde el ganador (el individuo seleccionado) será el de mejor calidad, el de mayor aptitud. Existen dos variantes, la determinística donde siempre se elige al más apto, y la probabilística donde se elige con una cierta probabilidad al individuo más apto, y en el caso contrario, se elige al menos apto.
- **Selección de estado uniforme** [95]: en este caso solamente algunos individuos serán reemplazados por los nuevos (hijos) en cada generación. Esta técnica es utilizada en algoritmos genéticos no generacionales.

2.2.1.5 Operador de cruza

La finalidad de este operador es combinar dos o más padres para obtener uno o más hijos tomando las mejores características de los padres para crear mejores hijos. Entre las formas más usuales de realizar la cruza se encuentra la de n puntos [25] propuesta para representaciones binarias, de la cual se muestra un ejemplo en la figura 2.9. Sin embargo, existen cruzas para representaciones reales y enteras, que en muchos casos intentan emular la cruza de n puntos. Este operador se aplica con cierta probabilidad, es decir, no siempre se va a realizar el intercambio de genes. En los casos en que la cruza no se lleva a cabo, los padres participantes generan hijos que son una copia idéntica de ellos.

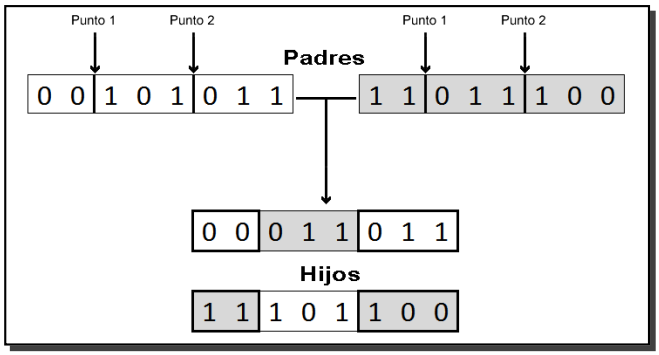


Figura 2.9: Cruza de dos puntos.

2.2.1.6 Operador de mutación

Este operador afecta el cromosoma de un solo individuo a la vez, y su objetivo es permitir la generación de soluciones que la cruza no puede producir. Lo que busca es emular la mutación que existe en el ámbito de la biología cuando se realiza la copia de genes y parte de la información sufre alteraciones. Después de realizar la cruza, el nuevo individuo sufre modificaciones que en la mayoría de los casos son muy leves, es decir, se modifica una parte mínima de todo el cromosoma.

Al igual que la cruza, la mutación se realiza con cierta probabilidad, pero a diferencia de la cruza, esta probabilidad indica si el alelo o el gene cambiarán de valor (serán mutados). Una de las técnicas de mutación más utilizadas es la uniforme, que recorre el cromosoma completo con alguna probabilidad de realizar el cambio, un ejemplo se muestra en la figura 2.10.

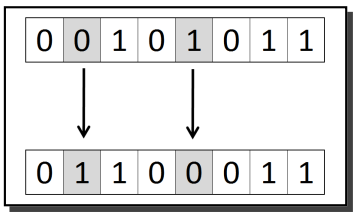


Figura 2.10: Mutación con $p = 1/4$.

2.2.1.7 Elitismo

Este mecanismo pretende asegurar que aquel o aquellos individuos que son los más aptos de la población actual sobrevivan y continúen participando en el proceso evolutivo, pasando a la siguiente generación de manera intacta (sin recombinarse ni mutarse). Implementar este mecanismo asegura que la mejor aptitud encontrada hasta el momento (el mejor individuo hasta el momento) no se reducirá en la siguiente generación. El elitismo es importante, pues existe una prueba matemática que garantiza la convergencia global de un algoritmo genético [73].

2.3 Principales paradigmas

Dentro del cómputo evolutivo existen diferentes paradigmas. Aunque todos los paradigmas se basan en la misma idea del neo-darwinismo y en el uso de una población de soluciones, difieren entre ellos por la forma de implementar los mecanismos de selección, cruza, mutación y elitismo. Los principales paradigmas son: los algoritmos genéticos, las estrategias evolutivas y la programación evolutiva. Aunque también existen otras técnicas como la programación genética [51], la evolución diferencial [86], la optimización mediante cúmulos de partículas [46, 58], la optimización por colonia de hormigas [26], los algoritmos culturales [71], los sistemas inmunes artificiales [19] y la búsqueda dispersa [54]. A continuación sólo se explicarán muy brevemente los paradigmas principales.

2.3.1 Algoritmos genéticos

En la actualidad, este paradigma se presume como el más popular. Aunque existen propuestas similares [83], como la de Fraser [36], y la de Bremermann [8], se considera a John Holland [4, 72, 40] como el que definió las bases de los algoritmos genéticos modernos.

En cuanto a la representación, tradicionalmente hace uso de una cadena binaria, es decir, trabajan a nivel de genotipo, lo que equivale a transformar el problema de un espacio cualquiera al binario. Por ello, en el algoritmo genético una representación adecuada será un factor importante para obtener bueno resultados.

Tradicionalmente, hace uso de la selección proporcional [40] con base en aptitud. Le da mayor importancia al operador de cruza que al de mutación, y no cuenta con mecanismos de auto-adaptación. En cuanto a la probabilidad de cruza se utilizan valores altos, al contrario de la probabilidad de mutación donde los valores usualmente son bajos.

El pseudocódigo de un algoritmo genético simple se muestra en el Algoritmo 2.

2.3.2 Estrategias evolutivas

Este esquema fue propuesto por Peter Bienert [6, 82], Ingo Rechenberg [70, 82] y Hans-Paul Schwefel [78, 82, 81] y consiste en un método de ajustes discretos aleatorios inspirado en el proceso de mutación biológico [88].

Algoritmo 2 Algoritmo Genético Simple

1: Inicializar población P aleatoriamente;
2: Evaluar la aptitud de los individuos en P;
3: **repeat**
4: Seleccionar padres;
5: Cruzar padres;
6: Aplicar mutación a los hijos (generados de la cruza);
7: Evaluar la aptitud de los hijos;
8: Seleccionar la nueva población entre los padres e hijos creados;
9: **until** La condición de paro se satisfaga

En la propuesta inicial, el esquema sólo usaba un individuo que era mutado para producir un descendiente. Sin embargo, Schwefel introdujo más adelante el uso de poblaciones [81].

La forma de representación es el uso de vectores reales (a nivel fenotipo), donde la mutación es el operador principal y se realiza de forma Gaussiana [5]. Se utiliza un esquema de selección determinística y la cruza suele ser discreta o intermedia.

En cuanto a la selección, presenta dos esquemas, el primero basado en la aptitud donde se toman en cuenta ambas poblaciones (padres e hijos) y el segundo basado en la generación, el cual sólo se enfoca en la nueva población (hijos).

La primera versión, sin población, es conocida como $(1+1)-EE$, donde con un solo padre genera un solo hijo y el mejor sobrevive a la siguiente generación. La forma de generarlo era la siguiente: a partir de un padre $\vec{x} = (x_1, x_2, \ldots, x_n)$ se obtiene un hijo $\vec{x}' = (x'_1, x'_2, \ldots, x'_n)$ por medio de la siguiente ecuación:

$$x'_i = x_i + N_i(0, \sigma_i)$$

donde $N_i(0, \sigma_i)$ es una función que genera un número aleatorio Gaussiano con media cero y desviación estándar σ_i. Se genera un número aleatorio independiente para cada componente del vector.

Rechenberg [70] propone la $(\mu+1)-EE$, la cual cuenta con μ padres que generan un solo hijo y se seleccionan μ individuos para la siguiente generación, es decir, el hijo puede o no reemplazar a algún padre.

Después aparecen dos propuestas de Schwefel [79] que son $(\mu+\lambda)-EE$ y $(\mu,\lambda)-EE$, donde se tienen μ padres que generan λ hijos. La diferencia

entre ambas radica en la estrategia de selección. La primera selecciona de ambas poblaciones a los mejores y la segunda selecciona μ individuos de los λ hijos generados (obviamente se requiere que $\mu \leq \lambda$).

La desviación estándar que se utiliza para la generación de los nuevos individuos es un parámetro sumamente importante en la mutación Gaussiana, y el valor de ésta será un factor para el éxito del paradigma. Sin embargo, el valor óptimo de este parámetro depende de la dimensionalidad y naturaleza del problema [83].

Del hecho anterior, surgieron los estudios realizados por Schwefel con la estrategia $(\mu + 1) - EE$, donde logró adaptar de manera automática el valor de la desviación estándar σ_i, lo que se conoce como auto-adaptación. Ésta es una de sus principales características y lo realizan mediante la codificación de sus parámetros que son tratados como variables adicionales que se desean optimizar [80].

2.3.3 Programación evolutiva

Este paradigma fue propuesto por Lawrence J. Fogel [33], quien planteó que el comportamiento inteligente requiere de la capacidad de hacer predicciones correctas y de traducir dichas predicciones en una respuesta adecuada. En este esquema la inteligencia se ve como un comportamiento adaptativo y le da más importancia al vínculo entre padres e hijos que a los operadores genéticos.

Fogel define a una población de autómatas de estado finito donde el ambiente era una secuencia de símbolos. De tal forma, cada padre recibía un símbolo de entrada y realizaba la predicción pertinente sobre el siguiente símbolo la cual se comparaba con el símbolo real proporcionado por el ambiente. El objetivo final era que el autómata fuera capaz de predecir la secuencia de símbolos proporcionada por el ambiente con la máxima precisión posible.

Para la representación, utilizaba números reales para los individuos, una mutación de tipo Gaussiana e implementaba un esquema de auto-adaptación (similar a las estrategias evolutivas). Este esquema no implementa un operador de cruza, ya que la simulación del proceso evolutivo se realiza a nivel de las especies y especies distintas no pueden recombinarse [31, 32].

En cuanto a la mutación, se pueden considerar cinco tipos [57]: cambiar el símbolo de salida, cambiar la transición a un estado, agregar un estado, eliminar un estado y cambiar el estado inicial.

El pseudocódigo básico de la programación evolutiva [88] se muestra en el Algoritmo 3.

Algoritmo 3 Algoritmo Básico de la Programación Evolutiva

```
1: Generar población aleatoriamente;
2: repeat
3:     Aplicar mutación;
4:     Calcular aptitud de cada hijo;
5:     Seleccionar las soluciones para la nueva población;
6:     Reemplazar la población vieja por la nueva;
7: until La condición de paro se satisfaga
```

2.4 Conceptos de optimización multi-objetivo

En la actualidad, en las diferentes áreas del conocimiento se requiere de la toma de decisiones, y muchas veces no sólo son de tipo técnico sino que involucran temas administrativos o sociales, por lo que los problemas deben resolverse de forma eficiente (de buena forma) y eficaz (en un tiempo razonable). De ahí la importancia de la optimización, que permite minimizar (o maximizar) una función de costo, sujeta a diversas restricciones impuestas sobre un problema. Sin embargo no existe ningún método de optimización capaz de solucionar cualquier tipo de problema.

La optimización se puede dividir en dos tipos: mono-objetivo y multi-objetivo. La primera abarca problemas con una sola función objetivo (más adelante aclararemos este concepto) y la segunda se refiere a problemas donde existen dos o más funciones objetivo (que en la mayoría de los casos se encuentran expresadas en unidades diferentes y en conflicto entre sí).

Para la optimización mono-objetivo existen diferentes propuestas que se han desarrollado a lo largo de los años y que resultan útiles para diferentes tipos de problemas [69].

En el campo de la investigación de operaciones, se han desarrollado varias técnicas para lidiar con problemas de optimización multi-objetivo [63]. Sin embargo, estos métodos muestran serias limitaciones y en muchas ocasiones son inútiles para resolver el problema planteado. Los algoritmos evolutivos y su carácter poblacional los convierten en una técnica sumamente adecuada para la resolución de problemas multi-objetivo, ya que pueden generar un

conjunto de soluciones compromiso en una sola ejecución, además de ser menos susceptibles a las diferentes características que pueda tener el frente de Pareto (p. ej., discontinuidades, concavidades y convexidades) [57].

En este capítulo introduciremos varios conceptos de optimización en general y de la optimización multi-objetivo en particular, así como las diferentes técnicas que existen para dar solución al problema de la optimización multi-objetivo, ya que éste es el tipo de optimización de interés para esta tesis.

2.4.1 Conceptos de optimización

Para comprender mejor lo que es un problema de optimización y cómo trabajan las diferentes técnicas disponibles para resolverlos, es necesario definir algunos términos.

- **Variables de decisión**: Son el conjunto de n parámetros (de entrada para la función objetivo) cuyos valores arrojan una posible solución al problema. Estas variables pueden ser enteras, reales, o cualquier combinación de ellas. Cada uno de estos parámetros es denotado como $x_i, i = 1, 2, \ldots, n$. El vector $\vec{x}$ de n variables se representará de la siguiente forma:
$$\vec{x} = [x_1, x_2, \ldots, x_n]$$

- **Restricciones**: En la mayoría de los problemas del mundo real existen limitaciones arrojadas por las condiciones del ambiente y los recursos con los que se cuenta, las cuales deben satisfacerse. A estas limitaciones se les conoce como restricciones, que delimitan el problema y dan validez a las soluciones. Éstas pueden ser restricciones de desigualdad:
$$g(\vec{x}) \leq 0$$
o de igualdad:
$$h(\vec{x}) = 0$$

- **Función objetivo**: Forma el criterio de evaluación de las variables de decisión, que nos dirá la calidad de la solución. La definición de la función objetivo está dada por la naturaleza del problema. La función objetivo se define como $f(\vec{x})$. Los problemas pueden tener una sola función o varias. El primer tipo de problema se denomina mono-objetivo y el segundo es conocido como multi-objetivo. Para fines de esta tesis, abordaremos problemas multi-objetivo.

- **Problema de optimización mono-objetivo**: se define como un conjunto de n variables de decisión, que optimizan (minimizan o maximizan) una función objetivo, satisfaciendo m restricciones de desigualdad y p restricciones de igualdad. Debido a que el problema de maximizar $f(\vec{x})$ es equivalente a minimizar $-f(\vec{x})$, el problema de optimización mono-objetivo se puede generalizar al tener un vector de variables de decisión $\vec{x}$ tal que minimice $f(\vec{x})$ y cumpla con:

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

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

  Es decir, que minimice la función objetivo y que cumpla con las restricciones dadas.

- **Problema de optimización multi-objetivo**: se define como un conjunto de n variables de decisión, que optimizan (minimizan o maximizan) k funciones objetivo, donde $k \geq 2$, satisfaciendo m restricciones de desigualdad y p restricciones de igualdad. Formalmente, buscamos el vector de variables de decisión $\vec{x}$ tal que minimice:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]$$

  y cumpla con:

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

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

  Es decir, que minimice las funciones objetivo y que cumpla con las restricciones dadas. Para fines de esta tesis, sólo se tratarán problemas multi-objetivo sin restricciones.

2.4.2 Optimización multi-objetivo

Los problemas de optimización multi-objetivo están formados por diferentes funciones objetivo que están expresadas en unidades diferentes y se encuentran en conflicto entre sí (al menos parcialmente).

Los problemas multi-objetivo tienen la particularidad de no tener una solución única, sino un conjunto de soluciones compromiso; a este conjunto se le conoce como conjunto óptimo de Pareto (más adelante se explicará a más detalle dicho concepto y de dónde se desprende).

El problema de optimización multi-objetivo es definido por Osyczka como [68, 88]:

> *El problema de encontrar un vector de variables de decisión que satisfaga las restricciones y que optimice una función vectorial cuyos elementos representen las funciones objetivo. Estas funciones forman una descripción matemática de criterios de desempeño que están usualmente en conflicto entre sí. Por lo tanto, el término optimizar significa encontrar aquella solución que daría un valor aceptable al diseñador en todas las funciones objetivo.*

Para los problemas multi-objetivo, el concepto de óptimo tiene otro enfoque que en el caso de la optimización global. La primera noción de optimalidad en un contexto multi-objetivo fue propuesta por Francis Ysidro Edgeworth [27] en 1881, sin embargo, fue Vilfredo Pareto [93] en 1896 quien la generalizó.

Entre los conceptos más relevantes y que serán de utilidad para esta tesis, encontramos los siguientes:

- **Dominancia**: Para el caso de minimización, decimos que con dos vectores $\vec{x_1}, \vec{x_2} \in F$, donde F es la zona factible (o sea, la región del espacio de búsqueda donde se satisfacen todas las restricciones del problema), tenemos que $\vec{x_1}$ domina a $\vec{x_2}$ (denotado mediante $\vec{x_1} \prec \vec{x_2}$) si y sólo si $\vec{x_1}$ es parcialmente menor que $\vec{x_2}$ [57]. Es decir, en un problema con k funciones objetivo,

  $$\vec{x_1} \prec \vec{x_2}$$

  si y sólo si

  $$\begin{aligned} \vec{f_i}(\vec{x_1}) \leq \vec{f_i}(\vec{x_2}), &\quad \forall i \in [1, 2, \ldots, k] \text{ y} \\ \exists j \in [1, 2, \ldots, k], &\quad \text{tal que } \vec{f_j}(\vec{x_1}) < \vec{f_j}(\vec{x_2}) \end{aligned}$$

- **Conjunto de óptimos de Pareto**: El conjunto de óptimos de Pareto (P^*) se define como:

  $$P^* = \left\{ \vec{x_*} \in F \,|\neg \exists \vec{x} \in F|\, \vec{f}(\vec{x}) \prec \vec{f}(\vec{x_*}) \right\}$$

  Es decir, es el conjunto de vectores solución tal que no existe ningún otro vector dentro que domine a cualquiera de éstos.

- **Frente de Pareto**: El frente de Pareto FP^* se define como:

$$\mathrm{FP}^* = \left\{ \vec{u} = \vec{f} = (f_1(\vec{x_*}), \ldots, f_k(\vec{x_*})) \,|\vec{x_*} \in P^* \right\}$$

  Existen problemas que contienen diferentes frentes de Pareto locales, y que suelen atraer al algoritmo de búsqueda. A estos problemas se les conoce como multifrontales [88].

  Los frentes de Pareto pueden tener diferentes características geométricas tales como concavidad, convexidad, discontinuidades, etc.

Tomando en cuenta los conceptos anteriores, el objetivo de la optimización multi-objetivo se vuelve complejo ya que no se puede dar mayor importancia a alguna solución no dominada en particular si no se cuenta con información adicional del problema, ya que todas las soluciones que pertenezcan al conjunto de óptimos de Pareto tienen la misma importancia entre sí. Así, el objetivo principal se concentra en dos aspectos: el primero en encontrar un conjunto de soluciones que sean parte o por lo menos se encuentran muy cercanas al verdadero frente de Pareto, y el segundo aspecto es tener un conjunto de soluciones lo más diversas entre sí, es decir, que no se concentren en una sola parte del frente, sino que estén uniformemente distribuidas a lo largo de éste.

2.5 Algoritmos multi-objetivo

En cuanto a la optimización mono-objetivo se cuenta con diferentes técnicas como son [69, 52, 59]: técnicas estocásticas, métodos estadísticos, métodos de cálculo diferencial, método de los multiplicadores de Lagrange, el método de Nelder-Mead, la programación lineal, cuadrática, no lineal, geométrica, entera, etc. Sin embargo, estas técnicas tienen diversas limitantes. Por ejemplo, algunas sólo son aplicables a cierta clase de funciones; otras, requieren información específica del problema. En contraste, los algoritmos evolutivos proporcionan mayor flexibilidad y facilidad de uso, aunque también adolecen de ciertas desventajas (por ejemplo, requieren la definición de algunos parámetros que suelen afectar su desempeño).

En el área de la investigación de operaciones se han desarrollado un gran número de técnicas capaces de lidiar con los problemas multi-objetivo [68, 63]

y también se han desarrollado técnicas alternativas a éstas como los algoritmos evolutivos, que son los de interés en la presente tesis. La motivación para el uso de algoritmos evolutivos multi-objetivo ha sido similar al caso mono-objetivo.

2.5.1 Técnicas tradicionales

Existen diversas propuestas de técnicas evolutivas que se han propuesto para optimización multi-objetivo. Según la clasificación de técnicas multiobjetivo propuesta por Cohon y Marks [15] se dividen en:

- **Técnicas a priori [45, 30, 16]**: Las preferencias del usuario tienen que ser conocidas antes de comenzar la búsqueda. Se especifican las características de la solución deseada, pero, el inconveniente que presentan es que no siempre se pueden saber de antemano dichas características.
- **Técnicas progresivas [97, 99, 64]**: Las preferencias se van dando conforme la búsqueda avanza y el tomador de decisiones indica si una solución le parece adecuada o no y el proceso actualiza las preferencias conforme el tomador de decisiones lo va indicando, guiando así el proceso de búsqueda.
- **Métodos a posteriori [37, 98, 55]**: Las preferencias se expresan al final y el tomador de decisiones recibe la información completa de los resultados para así entonces tomar la decisión que mejor le convenga. Es decir, los resultados intentan mostrar todos los compromisos posibles entre las funciones objetivo tratando de generar el verdadero frente de Pareto o al menos una aproximación razonablemente buena.

Estas técnicas presentan diversas dificultades, tales como [11, 20, 34, 62, 57]:

- Requieren varias ejecuciones para encontrar varios elementos del conjunto de óptimos de Pareto.
- Las soluciones encontradas suelen ser escasas.
- Suelen requerir información adicional sobre el problema a resolverse.
- Suelen ser sensibles a ciertas características del verdadero frente de Pareto (por ejemplo, discontinuidades).

- Las soluciones obtenidas suelen estar mal distribuidas a lo largo del frente de Pareto verdadero.
- Resultan inadecuadas para problemas con incertidumbre o ruido.

2.5.2 Algoritmos evolutivos multi-objetivo

La primera implementación formal que sentó las bases para los algoritmos evolutivos multi-objetivo (AEMO) fue propuesta por David Schaffer con su algoritmo genético de evaluación vectorial (VEGA) [75, 74]. Posteriormente, diversos investigadores han realizado diferentes propuestas algorítmicas [14, 84, 105, 41, 35, 47, 12], que en general, han mostrado una clara superioridad en su desempeño con respecto a las técnicas clásicas.

Las características fundamentales que contiene un AEMO son [104]:

- Operan sobre un conjunto de soluciones potenciales al problema, lo cual les permite generar varios elementos del conjunto de óptimos de Pareto en una sola ejecución.
- Adoptan un mecanismo de selección cuyo objetivo es preservar las soluciones no dominadas de cada generación.
- Cuentan con un mecanismo que busca preservar la diversidad en la población (el denominado estimador de densidad).
- Cuenta con un mecanismo elitista que preserva las soluciones no dominadas globales.

Las últimas tres características hacen al AEMO diferente al algoritmo evolutivo mono-objetivo. El mecanismo de selección de un AEMO debe incorporar información sobre las (varias) funciones objetivo del problema, lo cual dificulta el uso de un esquema tradicional, basado en aptitud. Para resolver dicho problema se tienen tres criterios principales [105, 57]:

- **Ordenamiento lexicográfico**: Se toma en cuenta una sola función objetivo, y se optimiza en la mayor medida de lo posible. Posteriormente, se toma otra función objetivo y, a partir del resultado de la primera, se le optimiza manteniendo el valor obtenido previamente para los objetivos anteriores. Este procedimiento continúa hasta haber procesado todos los objetivos del problema.

- **Basados en agregación**: El conjunto de funciones objetivo es combinado para obtener una sola función escalar a optimizarse.
- **Basados en jerarquización de Pareto**: Asignan la aptitud con base en la dominancia de Pareto. Este esquema es el más popular en la actualidad, y existen muchos propuestas que se basan en él [35, 41, 85].

En cuanto al elitismo, en un algoritmo evolutivo mono-objetivo se conserva a la mejor solución para que ésta participe en la siguiente generación dentro del proceso evolutivo. Sin embargo, en los AEMO no existe una solución única que sea la mejor en cualquier generación, sino que existe un conjunto de soluciones no dominadas, que son igualmente buenas entre sí. Para implementar el elitismo en un AEMO existen dos mecanismos principales:

- Realizar una selección + como en las estrategias evolutivas, donde la población de padres e hijos se juntan para elegir de ahí a los mejores individuos con base en la dominancia de Pareto [23].
- Mantener una población secundaria conocida como archivo histórico, donde se guardarán las soluciones no dominadas encontradas a lo largo del proceso evolutivo [105].

En sus orígenes, los AEMO no solían emplear elitismo, por lo cual, históricamente, se suelen considerar dos generaciones de AEMOs. Los AEMOs de primera generación son los no elitistas y los de segunda son los elitistas.

A continuación se dará una pequeña descripción de los AEMO más representativos del estado del arte y sus características principales, comenzando con los de primera generación:

- **VEGA (Vector Evaluated Genetic Algorithm) [75]**: Propuesto por David Schaffer, quein usó como base el *GENESIS* [39], al que modificó la forma de realizar la selección a fin de poder manejar varias funciones objetivo.
- **MOGA (Multi-Objective Genetic Algorithm) [35]**: Esta propuesta jerarquiza a los individuos basándose en la dominancia de Pareto. Al individuo o individuos que no son dominados por ningún otro se les asigna la mejor jerarquización y para los demás, el valor de su jerarquía es igual al número de soluciones que lo dominan.

- **NSGA (Non-dominated Sorting Genetic Algorithm) [84, 85]**: Realiza una jerarquización basada en la dominancia de Pareto, pero, clasifica a las individuos en diferentes capas. Los individuos no dominados de la primera capa son los que tendrán una mayor aptitud y por medio de la selección proporcional tendrán una mayor probabilidad de ser seleccionados. Con esto se pretende que la búsqueda se enfoque en los individuos no dominados globales (con rspecto a la población actual).

- **NPGA (Niched-Pareto Genetic Algorithm) [43, 42]**: Hace uso de la selección mediante torneo y la jerarquización basada en la dominancia de Pareto. Lo que realiza es comparar dos individuos contra un subconjunto de la población. El ganador es aquél que no sea dominado. En caso de empate se decide al ganador por medio de la compartición de aptitud [38](el individuo que se encuentre en la región menos poblada es el vencedor).

Como parte de la segunda generación tenemos a los siguientes algoritmos:

- **SPEA (Strength Pareto Evolutionary Algorithm) [105]**: Intenta integrar diferentes AEMOs de la primera generación. Usa un archivo externo para almacenar las soluciones no dominadas encontradas hasta el momento. Utiliza una jerarquización similar al MOGA y se auxilia del archivo externo para realizar la comparación de jerarquías. Utiliza un método de agrupamiento [65] para mantener la diversidad de las soluciones.

- **PAES (Pareto Archived Evolution Strategy) [47, 48]**: Es una estrategia evolutiva del tipo $(1 + 1)$ y hace uso de un archivo histórico que almacena las soluciones no dominadas encontradas. Todo nuevo individuo es comparado con el archivo histórico. Implementa un malla adaptativa para mantener la diversidad de las soluciones en el archivo y divide el espacio de las funciones objetivo para agrupar las soluciones encontradas.

- **SPEA2 (Strength Pareto Evolutionary Algorithm 2) [103]**: Cuenta con cuatro características que lo diferencian de su antecesor: 1) incorpora una asignación de aptitud de grano fino (toma en cuenta el número de individuos que dominan y a los que domina cada individuo); 2) estima la densidad del vecino más cercano para guiar la

búsqueda; 3) mejora el método de truncado del archivo para preservar las soluciones en la frontera del frente de Pareto; 4) utiliza un algoritmo de agrupamiento menos complejo computacionalmente que el de SPEA.

- **NPGA2 (Niched Pareto Genetic Algorithm [29])**: Utiliza una jerarquización basada en dominancia de Pareto y mantiene la selección por torneo, pero en caso de empate se basa en la aptitud para desempatar. El mecanismo de elitismo es parecido a la selección +. Además, utiliza un mecanismo llamado compartición de aptitud continuamente actualizada [67], para el conteo de los nichos.

- **PESA (Pareto Envelope-based Selection Algorithm) [49]**: Hace uso de dos poblaciones, una interna de tamaño reducido y otra externa de tamaño mayor. Utiliza el mismo mecanismo de malla adaptativa que el PAES. El mecanismo de selección se basa en la métrica de densidad de población usada por la malla adaptativa. El archivo externo determina la selección realizada por el método.

- **μ-GA (Micro-Genetic Algorithm) [13, 14, 89]**: Utiliza una población muy pequeña (cuatro individuos), y por medio de una memoria externa almacena las soluciones no dominadas encontradas a lo largo del proceso. Esta memoria externa es una versión modificada de la malla adaptativa de PAES. Existe una porción que es reemplazable con individuos generados aleatoriamente, y la otra porción nunca se reemplaza.

- **PESA-II (Pareto Envelope-based Selection Algorithm - II) [17]**: Muy similar al PESA, pero utiliza una selección basada en regiones, es decir, utiliza un hipercubo como unidad de selección en vez de utilizar a un individuo. El objetivo final de la propuesta es reducir el costo computacional de los AEMOs basados en la dominancia de Pareto.

- **NSGA-II (Non-dominated Sorting Genetic Algorithm - II) [24]**: Utiliza la selección + para llevar a cabo el elitismo, es más eficiente que el NSGA original (computacionalmente hablando) y utiliza un operador que estima la densidad de soluciones que rodean a un individuo, como mecanismo para mantener la diversidad de las soluciones no dominadas.

Dentro de las propuestas anteriores el NSGA-II es por mucho el AEMO más popular y citado dentro de la literatura [22]. Una de las principales características por las que es tan popular es por la reducida cantidad de parámetros de entrada que requiere, así como su simplicidad de uso y su robustez. Por ello elegimos al NSGA-II realizar la implementación de las diferentes técnicas de auto-adaptación propuestas en esta tesis.

3

Ajuste de Parámetros

Un AEMO cuenta con diferentes parámetros, que normalmente interactúan entre sí de forma no lineal, por lo que no pueden optimizarse de forma independiente. Aunque se han llevado a cabo diversas investigaciones para definir el conjunto óptimo de parámetros o la forma para encontrar dichos valores, no existe ninguna propuesta que sea aceptada por un sector amplio de la comunidad de computación evolutiva.

En este capítulo se definirá en qué consiste la definición de parámetros, los diferentes enfoques que existen y se hará una revisión de las propuestas actuales sobre la adaptación de parámetros en algoritmos evolutivos multiobjetivo.

3.1 Introducción

Una de las características principales de algoritmo evolutivo es la de contar con parámetros que controlan su funcionamiento. Establecer los valores de los diferentes parámetros de un algoritmo evolutivo es crucial para su buen desempeño.

El valor de los diferentes parámetros determinará en gran medida si el

algoritmo encontrará el óptimo del problema (o al menos una buena aproximación de éste), así como la consistencia con que lo hará. Sin embargo, elegir los parámetros más adecuados para un problema arbitrario es una tarea difícil. Encontrar un conjunto de valores para los parámetros de un algoritmo evolutivo es un problema complejo, que no ha sido bien estructurado e incluso no se ha logrado definir claramente. Para ello, muchos de los investigadores se han planteado un conjunto de preguntas:

- ¿Existe un conjunto de parámetros óptimos para cualquier algoritmo evolutivo?
- ¿Existe un conjunto de parámetros óptimos para algún problema en específico?
- ¿Es conveniente modificar los parámetros durante la ejecución de un algoritmo evolutivo?
- ¿Cómo afecta el valor de cada parámetro el desempeño de un algoritmo evolutivo?
- ¿Cómo afectan las características del problema la elección de los parámetros?

Como punto de partida podemos tomar en cuenta el *No Free Lunch Theorem* (NFLT) [96], que nos dice que no existe ningún algoritmo estocástico de búsqueda que sea mejor a todos los demás en todas las clases de problemas. Esto se debe a que todas las técnicas de búsqueda heurística son matemáticamente equivalentes en general, por lo que no hay una técnica que supera a las demás en todos los casos.

La principal consecuencia de este teorema es que no se debe buscar realizar una propuesta algorítmica que sea capaz de resolver todos los tipos de problemas dado que eso es imposible. En vez de eso, debiéramos buscar entender mejor el comportamiento de un algoritmo evolutivo en particular, al usarse en una cierta clase de problemas.

Eso es precisamente lo que se persigue en esta tesis, ya que propondremos un esquema de ajuste automático de parámetros para problemas multiobjetivo cuyas variables se expresan mediante números reales. Dicho ajuste automatizado se basará en explotar el uso de conocimiento generalizado sobre los problemas de nuestro interés.

3.2 Tipos de ajuste de parámetros

En cuanto al ajuste o determinación de parámetros, podemos dividir a las técnicas en dos grupos:

- **Afinación de parámetros**: Consiste en encontrar un conjunto de buenos valores para los parámetros antes de ejecutar el algoritmo, para después ejecutarlo con los valores encontrados, y estos valores se mantienen sin cambios durante toda la ejecución.
- **Control de parámetros**: Consiste en comenzar la ejecución del algoritmo con ciertos valores en los parámetros, los cuales cambian durante la ejecución.

Para encontrar los mejores valores en el proceso de ajuste de parámetros, es necesario llevar a cabo varios experimentos con diferentes combinaciones de parámetros y después seleccionar la combinación que mostró los mejores resultados. Las desventajas principales de este esquema son:

- Los parámetros no son independientes, no tienen un comportamiento lineal, y probar todas las diferentes combinaciones posibles es, prácticamente imposible.
- El proceso de ajustar los parámetros consume mucho tiempo, incluso si los parámetros son optimizados uno por uno, a pesar de que interactúan entre ellos.
- Para un problema en específico los valores seleccionados para los parámetros no necesariamente son los óptimos, inclusive si se llevaron a cabo experimentos con una gran cantidad de combinaciones de éstos.

Existen varios estudios que han intentado encontrar estos valores óptimos. Los primeros investigadores que buscaron los valores óptimos de los parámetros de un algoritmo evolutivo fueron De Jong [25], Grefenstette [39] y Schaffer [76]. Sin embargo, los algoritmos evolutivos son dinámicos en el sentido de que una ejecución sigue un proceso de adaptación. Por ello, establecer parámetros que no cambien durante toda la ejecución va en contra de la naturaleza del algoritmo. Además, existe evidencia que usar diferentes valores de los parámetros en las diversas etapas del proceso evolutivo parece ser lo más adecuado [28].

Un ejemplo se puede observar con la mutación, donde variaciones grandes del porcentaje de mutación en las primeras generaciones ayuda en el proceso de exploración, mientras que cambios pequeños son más útiles en las últimas generaciones (etapa de explotación) para mejorar al mejor individuo.

3.3 Control de parámetros

Para llevar a cabo el control de parámetro, se debe tener claro los siguientes puntos [28]:

- **El parámetro que se va a controlar**: por ejemplo, la probabilidad de mutación, la probabilidad de cruza, el tamaño de población, etc.
- **La forma en que se realizarán los cambios**: si se realizarán de forma determinista, con adaptación o se utilizará auto-adaptación
- **Las pruebas con las cuales se decidirá realizar el cambio del parámetro**: por ejemplo, midiendo la diversidad de la población, el desempeño de algún operador, etc.
- **El alcance que tendrá el cambio**: por ejemplo, a nivel poblacional, nivel individual, etc.

En cuanto al parámetro que se va a controlar, está claro que, dependiendo del algoritmo, los parámetros van a ser diferentes. Sin embargo, se tiene que tener muy claro cuál o cuáles serán los parámetros sobre los que el control se efectuará.

Los cambios para llevar a cabo el control se pueden hacer de la siguiente forma:

- **Determinista**: Los cambios se harán con alguna regla predeterminada, sin utilizar información arrojada por el proceso evolutivo. Un ejemplo, es el utilizar una regla que siga el número de generaciones, y según la generación actual, determine el valor que se adopte para el parámetro adaptado.
- **Adaptación**: Cuando se utiliza información del proceso evolutivo que sirve para conocer la dirección o la magnitud del cambio en el parámetro. También se debe determinar en qué momento se detendrá o si

mantendrá el nuevo valor del parámetro. Este mecanismo se maneja de forma externa, es decir, no forma parte del ciclo evolutivo.

- **Auto-adaptación**: En este esquema, los parámetros a controlar son codificados dentro del individuo como variables adicionales del problema y son modificados directamente por los operadores genéticos.

En cuanto a la información que se tomará en cuenta para llevar a cabo el cambio, lo más común es monitorear el progreso de la búsqueda, por ejemplo, mediante la diversidad de la población. La información recolectada es usada para realizar el ajuste del o los parámetros requeridos. Se pueden observar dos casos:

- **Información absoluta**: Sucede cuando el valor del parámetro se modifica por alguna regla preestablecida que se cumple cuando cierta condición se satisface. Ejemplos son: cambiar la probabilidad de mutación por medio de alguna regla difusa usando la variedad de las estadísticas poblacionales, ó modificar la probabilidad de mutación cuando la diversidad de la población tenga cierto valor.

- **Información relativa**: Los valores de los parámetros se comparan de acuerdo con la aptitud de la nueva población producida, y los mejores valores son recompensados. La dirección o magnitud del cambio es guiada de forma relativa por otros valores, es decir, se debe tomar en cuenta más de un valor en cualquier momento.

Se debe tener también muy claro el alcance que tendrá el cambio, ya que se pueden modificar genes, individuos, poblaciones, operadores y hasta la función de evaluación.

3.4 Auto-adaptación multi-objetivo

Aunque el uso de mecanismos de ajuste automático de parámetros ha sido menos común en algoritmos evolutivos multi-objetivo, existen algunas propuestas al respecto, las cuales serán discutidas brevemente a continuación.

3.4.1 El micro–AG2

Una de las principales propuestas que implementan adaptación en línea en un algoritmo evolutivo multi-objetivo es el denominado micro–AG2 (μAG^2) [89], que es una versión sin parámetros del micro-AG [14].

En esta técnica, se adapta el operador de cruza ejecutando varios μAGs en paralelo. Se cuenta con tres operadores de cruza y la elección del operador a ser usado se realiza por medio de la distancia entre cada variable correspondiente a cada padre. Después se analizan los resultados, y el operador con el peor desempeño es reemplazado por el operador con el mejor desempeño.

En una memoria externa se guardan las soluciones globales no dominadas, y las nuevas poblaciones se conforman usando dicha memoria externa. En cuanto al número máximo de generaciones, el μAG^2 utiliza un criterio de convergencia, el cual consiste en revisar si dentro de las soluciones arrojadas por los μAGs ejecutados en paralelo no existen mejores soluciones que las ya existentes dentro de la memoria externa. Cuando no se han encontrado nuevas soluciones después de un tiempo razonablemente grande, entonces el algoritmo se detiene, pues considera que no tiene caso continuar iterando.

En cuanto al manejo de las fases de explotación y exploración, el μAG^2 intenta mantener un balance entre ambas, cambiando las prioridades de los operadores evolutivos. En la fase de exploración se le da mayor importancia al operador de mutación sobre el operador de cruza. En la fase de explotación el operador de mayor importancia es la cruza y el de menor importancia es la mutación. Los autores dividen los parámetros en dos grupos: los que no pueden ser adaptados en línea y los que sí. Dentro de los primeros tenemos a las funciones objetivo, el número de variables, las amplitudes de las variables y el número de vectores no dominados. Para los parámetros que pueden adaptarse los autores presentan los siguientes:

- **Porcentaje de mutación:** se fija dividiendo uno entre el número de variables de decisión (o genes).

- **Porcentaje de cruza:** cambia desde 0.5 hasta 1. El primer valor es usado en la fase de exploración y el segundo es usado en la fase de explotación.

- **Tamaño de la memoria de población:** fijado a $T_{\text{pareto}} \div 2$. Donde T_{pareto} es el tamaño de la memoria externa.

- **Porcentaje de la memoria no reemplazable:** fijado al 10 % del tamaño de la memoria de población.

- **Número total de iteraciones:** cuando la memoria externa ha sido reemplazada $T_{\text{pareto}} \times 2$ veces sin tener ningún individuo dominado.

- **Ciclo de reemplazo:** se reemplaza la memoria reemplazable cada que T_{pareto} individuos han sido evaluados.

- **Número de subdivisiones de la rejilla adaptativa:** No excede en promedio tres individuos por región y nunca menos de 1.5 individuos por región.

Para fines de su evaluación se adoptaron tres métricas: distancia generacional [92], tasa de error [91] y espaciado [77]. Los autores compararon sus resultados con respecto a los del NSGA-II [24], el PAES [48] y el micro-AG [14]. En conclusión, el μAG^2 se mantuvo competitivo con respecto al NSGA-II y al PAES, y en cuanto al micro-AG original normalmente mejora los resultados de éste sin la necesidad del ajuste manual de parámetro alguno.

3.4.2 IMOEA

Tan et. al. [87] propusieron el algoritmo llamado IMOEA (Incrementing Multiobjective Evolutionary Algorithm) que implementa un tamaño de población variable o dinámico que se obtiene de forma adaptativa de acuerdo con las soluciones encontradas en línea y con la distribución de densidad poblacional deseada.

Este algoritmo incorpora un método difuso de perturbación local en la frontera que altera a los individuos no dominados, para producir los individuos necesarios que incrementen la población a fin de alcanzar el tamaño deseable manteniendo además una buena distribución a lo largo del frente.

Este algoritmo cuenta con un mecanismo para medir la convergencia, el cual se basa en la población de dominados y en la tasa de progreso. Otra característica importante es que hace uso de nichos dinámicos, es decir, no requiere que el usuario defina ningún factor de compartición.

3.4.3 Adaptación de parámetros en optimización mediante cúmulo de partículas

En 2007, Zielinski y Laur [100] propusieron un esquema para la adaptación de tres parámetros en un algoritmo de optimización multi-objetivo basado en cúmulos de partículas.

La optimización mediante cúmulos de partículas se deriva de la vida artificial, en específico del comportamiento de grupos sociales como parvadas o bancos de peces. Los individuos en este tipo de optimización requieren conocimiento de un grupo social.

Cada una de las partículas se encuentra en un entorno dinámico, el cual es generado por la ecuación que les actualiza la velocidad y la posición de la siguiente forma:

$$\vec{v}_i(t+1) = w \cdot \vec{v}_i(t) + c_1 r_1 \left[\vec{p}_i(t) - \vec{x}_i(t)\right] + c_2 r_2 \left[\vec{g}_i(t) - \vec{x}_i(t)\right]$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1)$$

Donde: $\vec{x}_i$ es la posición actual en el espacio de búsqueda, $\vec{v}_i$ es la velocidad, que provee información de las búsquedas anteriores, $\vec{g}_i$ es la mejor posición encontrada hasta el momento en un vecindario específico y, $\vec{p}_i$ es la mejor posición encontrada hasta el momento por la partícula.

Estas ecuaciones causan movimiento hacia la mejor posición de la partícula (componente cognitivo) y la mejor posición del vecindario (componente social). En cuanto a las otras variables, tenemos a w que es la inercia, que se usa para regular el efecto de la velocidad previa. Así mismo, el impacto del componente cognitivo y social es influenciado por los parámetros c_1 y c_2 respectivamente, mientras que r_1 y r_2 agregan un elementos estocástico al movimiento de las partículas (su valor se escoge de manera aleatoria en el intervalo $[0, 1]$). Otros parámetros son NP (número de partículas o tamaño de población) y V_{max} que representa la velocidad máxima permisible.

El proceso está basado en un método del diseño de experimentos que es llamado Operación Evolutiva (Evolutionary Operation [EVOP]). Los autores señalan que EVOP es de gran utilidad para realizar un monitoreo continuo y mejorar los procesos con desempeño variable [66, 100].

La propuesta pretende adaptar tres parámetros (w, c_1, c_2). Para realizar el análisis se otorgan dos valores para cada parámetro, teniendo así $2^3 = 8$ combinaciones diferentes de parámetros.

El EVOP permite detectar tanto el comportamiento individual de los parámetros como su comportamiento cuando estos interactúan. Por medio del método estadístico ANOVA (*analysis of variance*) se evalúan las diferencias de desempeño de las diferentes combinaciones de parámetros.

Para obtener los datos a evaluar, cada una de las ocho combinaciones se aplica a un octavo de los miembros de la población, elegidos aleatoriamente.

Las parámetros son adaptados de acuerdo a su éxito, que se mide por la no dominancia de las partículas. Después de cada generación se revisa el éxito de cada combinación por medio del análisis de experimentos. Si existe alguna significativamente mejor, entonces el parámetro se ajusta con pasos de 0.05 para el mejor valor y de 0.1 para el otro (cada parámetro tiene 2 valores: para el menor valor se resta el paso y para el mayor se suma el paso).

Se vuelven a asignar los nuevos valores a un octavo de la población como al inicio. Si después de varios análisis no se han detectado cambios significativos, el análisis de diseño de experimentos es reinicializado. Esto se hace disminuyendo y aumentando los dos valores de cada parámetro, respectivamente, para hacer la distancia más grande entre los dos valores.

Los resultados arrojados no son del todo buenos en comparación con los de otros algoritmos. Los autores sugieren que esto sucede debido al uso de un algoritmo de cúmulo de partículas muy básico sin el uso de mecanismos más sofisticados.

3.4.4 Criterio de convergencia

En 2008, Trautmann et al. [90] propusieron un esquema para determinar en qué generación ha convergido un algoritmo evolutivo multi-objetivo (AEMO). Básicamente, el AEMO se ejecuta varias veces con diferentes semillas de generación de aleatorios y se analiza la varianza de los resultados de tres métricas para saber si éstos han cambiado o no significativamente.

El funcionamiento es el siguiente: con una cota del número de generaciones definida como $G^* \in [G_L, G_U]$ y con un paso de S (los autores recomiendan $S = 2$) se realizan cada una de las m ejecuciones (es decir m semillas diferentes, para tener m algoritmos ejecutándose de forma paralela, donde los autores recomiendan $m = 50$). La ejecución inicia en G^* y se pausa en $G^* + S$ con cada una de las m ejecuciones diferentes.

Después, se obtienen los valores de las métricas distancia generacional [92], hipervolumen [101] y dispersión [21], para cada una de las m ejecuciones. Luego, continúa la ejecución para detenerla posteriormente en $G^* + 2S$

realizando el mismo procedimiento de las métricas, y así sucesivamente hasta llegar a G_U.

Se analizan los valores de las métricas de las m ejecuciones en grupos de tres; es decir, se realiza el ANOVA (análisis de varianza) con los datos de las generaciones $G_L, G_L + S, G_L + 2S$ para verificar si no hay una varianza significativa [56].

Si ya no existe varianza significativa; es decir, si los resultados no varían (o varían muy poco), se puede concluir que la última de estas tres generaciones corresponde al valor óptimo de generaciones requerido para lograr convergencia.

En caso de que sí exista varianza en los resultados, se continúa analizando otro grupo de tres generaciones, pero ahora se analizarán $G_L + S, G_L + 2S, G_L + 3S$. Se sigue el mismo procedimiento hasta encontrar la generación óptima o haber revisado todas las generaciones; es decir, revisar la última tercia de generaciones $(G_U - 2S, G_U - S, G_U)$.

Si no se ha encontrado la generación óptima, entonces se actualizan los límites con $G_L := G_U - 5$ y a $G_U := G'_U$ donde $G'_U > G_U$ y se define de manera empírica.

Después, se retoman las m ejecuciones que estaban detenidas, y se continúa ejecutando hasta llegar al nuevo G_U. Luego, se sigue con el procedimiento de la misma manera, hasta encontrar la generación óptima.

La propuesta resulta muy costosa computacionalmente hablando, ya que realiza muchas ejecuciones para realizar la comparación estadística.

3.4.5 SPDE

Hussein Abbass propuso un algoritmo multi-objetivo llamado *Self-Adaptive Pareto Differential Evolution* (SPDE) [1], el cual auto-adapta sus porcentajes de cruza y mutación. Este algoritmo trabaja de manera muy similar a como lo hace el PDE original [2].

El porcentaje de cruza y el de mutación es heredado por los padres de la misma forma como se lleva a cabo con las variables de decisión, es decir, como si los porcentajes de cruza y mutación fuesen variables de decisión.

Para el caso de la cruza el hijo quedaría de la siguiente forma:

$$x_c = x_c^{\alpha_1} + N(0,1) \times (x_c^{\alpha_2} - x_c^{\alpha_3})$$

donde x_c es el porcentaje de cruza para el hijo, $x_c^{\alpha_n}$ es el porcentaje de cruza

del padre n y $N(0,1)$ es un número aleatorio con distribución Gaussiana, desviación estándar 0 y media 1.

En cuanto al valor de la mutación, quedaría:

$$x_m = x_m^{\alpha_1} + N(0,1) \times (x_m^{\alpha_2} - x_m^{\alpha_3})$$

donde x_m es el porcentaje de mutación para el hijo, $x_m^{\alpha_n}$ es el porcentaje de mutación del padre n y $N(0,1)$ es un número aleatorio con distribución Gaussiana, desviación estándar 0 y media 1.

Si x_c o x_m no se encuentran dentro del rango $[0,1]$, entonces se aplica una regla de ajuste, que trunca la parte entera dejando solamente la parte decimal, es decir, si tenemos un $x_c = 1.7$, con la regla de ajuste el nuevo valor sería $x_c = 0.7$ eliminando la parte entera (1).

El SPDE mejoró varias propuestas existentes al momento de ser publicado (2002).

Dentro de las conclusiones el autor notó que incrementar el tamaño de la población no tenía un gran efecto en el desempeño del algoritmo y que búsquedas más largas son más importantes que poblaciones grandes. Para los trabajos futuros, propone minimizar la carga al usuario con la elección e inserción de los diferentes parámetros, haciendo la propuesta más atractiva para los usuarios con problemas de la vida real.

3.4.6 Uso de mapas auto-organizados

Existe una propuesta de Dirk Büche et al. [10] que agrega una nueva forma de recombinación y un nuevo operador de mutación. Su principal objetivo era dar solución al problema del diseño de perfiles aerodinámicos de alas de avión.

El nuevo operador de mutación está basado en mapas auto-organizados de Kohonen [50], que aprenden del proceso evolutivo. Estos mapas adaptan la tasa de cambio de la mutación para enfocarse en áreas de soluciones prometedoras a fin de acelerar la convergencia. Las soluciones no dominadas son la base del conocimiento dentro de la red de los mapas.

Los autores realizaron un diseñador automático de perfiles aerodinámicos que se componía de dos herramientas usadas por los ingenieros especializados en el área y le agregaron el algoritmo evolutivo multi-objetivo basado en mapas auto-organizados.

3.4.7 PCGA

Kumar et. al [53] propusieron el PCGA (Pareto Converging Genetic Algorithm), que es un algoritmo evolutivo multi-objetivo que incluye un mecanismo de detección de convergencia.

El objetivo es obtener un frente de Pareto de un problema desconocido con soluciones diversas y obteniendo convergencia usando una cantidad mínima de parámetros y procedimientos. Su objetivo no es asegurar convergencia global, sino enfocar sus esfuerzos para llegar a ella.

La población es jerarquizada como lo hace el MOGA (Multi-Objective Genetic Algorithm) [35], es decir, el valor de jeraquización de cada individuo es igual al número de individuos que lo dominan. Se seleccionan dos padres por medio del método de la ruleta [25] de forma convencional. Se generan dos nuevos individuos por medio de la cruza, y estos son sometidos al operador de mutación. Con esto se obtiene una población de N individuos padres y 2 individuos hijos. Para mantener N individuos en la población se re-jerarquiza la población y se eliminan los dos peores individuos. Si existe empate en los peores individuos, se elige de entre los peores a alguno de forma aleatoria.

También se presenta el concepto de época, que es el proceso iterativo para formar dos nuevos individuos y reajustar el tamaño de población como se indicó anteriormente.

Los autores definen el concepto de islas PCGA, que son ejecuciones aisladas del algoritmo (no existe migración de individuos), que en un futuro serán comparadas entre sí, considerando dos o más islas. Sin embargo, dos o más islas pueden fusionarse para formar una sola a fin de realizar pruebas de convergencia, aunque esta fusión es opcional.

Otro concepto es el de histograma de jerarquización intra-islas que mide la frecuencia de distribución de jerarquías iguales en la población de una sola isla. También definen el histograma de jerarquización inter-islas que realiza una comparación entre las soluciones de diferentes islas, fusionando los individuos y re-jerquizando a la nueva población.

Los autores proponen realizar una comparación entre histogramas intra-islas en épocas sucesivas, para observar si se obtienen más individuos no dominados o en el peor de los casos se mantiene igual (por el hecho de sólo eliminar a las peores soluciones). Sin embargo, los autores resaltan que el tener toda la población con individuos no dominados y que no existan cambios en épocas consecutivas no significa haber logrado convergencia global. Sin embargo, en tal caso, es poco probable obtener mejoras significativas si

se continúa iterando.

En cuanto a los histogramas inter-islas, al unir poblaciones de islas se busca mantener la diversidad y realizar una comparación directa entre las diferentes islas. Eso se lleva a cabo de manera similar que con el histograma intra-islas, ya que se fusionan las poblaciones y se re-jerarquizan para obtener un histograma que nos indique si al agregar nuevos individuos, que también son no dominados en sus respectivas islas, la población mantiene la no dominancia. Para el análisis inter-islas sólo se toma en cuenta a los individuos no dominados.

La fusión de islas se realiza cuando el histograma intra-islas nos indica que en épocas sucesivas no han existido cambios, y se puede considerar poco fructífero el seguir iterando.

El modelo de islas busca realizar un mejor muestreo del frente de Pareto y trata de minimizar el procesamiento computacional. Además, el cómputo de una sola isla termina cuando se cumple el criterio de paro sin la necesidad de fijar un número máximo de generaciones.

El algoritmo logra diversidad sobre el frente de Pareto sin la necesidad de definir nichos y da un primer acercamiento a la forma en que se puede realizar la medición de la convergencia.

Los autores proponen, como trabajo futuro, analizar el comportamiento de los demás operadores como son: el porcentaje de cruza, el porcentaje de mutación, etc. Además, sugieren implementar pruebas estadísticas con los datos de los histogramas intra-islas e inter-islas

3.5 Sumario

El objetivo final de esta tesis es producir un algoritmo efectivo capaz de resolver diferentes tipos de problemas y que no contenga ningún parámetro visible al usuario, tomando en cuenta al *No Free Lunch Theorem*.

El hacer que un parámetro sea auto-adaptado, no necesariamente significa que obtendremos un algoritmo evolutivo con menos parámetros. Esto lo podemos observar en el algoritmo genético con tamaño de población variable [3], donde el parámetro de tamaño de población es eliminado a costa de aumentar dos nuevos parámetros (el tiempo de vida máximo y el mínimo).

Tomando en cuenta el *No Free Lunch Theorem*, debemos ser cuidadosos al momento de generalizar los resultados, y usar el ajuste de parámetros como un mecanismo para adaptar el algoritmo para problemas de alguna clase en

particular.

Muchas propuestas buscan auto-adaptar los parámetros con otras técnicas de optimización, pero ese tipo de enfoque no está libre del *No Free Lunch Theorem*, y, al no conocer el comportamiento exacto de los parámetros, la tarea de auto-adaptarlos puede resultar inefectiva. Es por ello que se debe tomar en cuenta al mismo proceso evolutivo como a la información que se obtiene para llevar a cabo la auto-adaptación.

4

Parámetros a Auto-adaptar

Como primera parte de esta investigación, fue necesario establecer claramente cuáles serían los parámetros a auto-adaptar y cuáles convenía más mantener fijos durante todo el proceso evolutivo. De ahí se desprende la necesidad de llevar a cabo un análisis de sensibilidad de parámetros, que nos mostrara de forma más clara su efecto en el desempeño del algoritmo adoptado.

En este capítulo se presenta el conjunto de parámetros que requiere el NSGA-II. Además, se agregaron nuevos parámetros para enriquecer con otras alternativas al algoritmo evolutivo. También se presentan los resultados arrojados por el análisis de sensibilidad de parámetros y las conclusiones derivadas de dicho análisis.

4.1 Conjunto de parámetros

El NSGA-II requiere de los siguientes parámetros [24]:

- Número de variables reales
- Número de variables binarias

- Número de funciones objetivo
- Funciones objetivo
- Número de restricciones
- Tamaño de población
- Número de generaciones
- Probabilidad de cruza
- Probabilidad de mutación de las variables reales
- Indice de distribución de la cruza de las variables reales
- Indice de distribución de la mutación de las variables reales
- Límites de las variables reales
- Si los límites de las variables reales son acotados
- Tipo de cruza para las variables binarias
- Número de bits para cada variable binaria
- Límites de las variables binarias
- Probabilidad de mutación binaria
- Semilla de aleatorios

De la lista anterior, existen parámetros que dependen de la naturaleza del problema, como son:

- Número de variables reales
- Límites de las variables reales
- Número de variables binarias
- Límites de las variables binarias
- Número de funciones objetivo

- Funciones objetivo

A los parámetros de la lista anterior no se les tomará en cuenta para ser auto-adaptados. Sin embargo, en el caso del número de variables reales, los límites de las variables reales, el número de las variables binaria y los límites de las variables binarias, al hacer uso de ambas codificaciones (real y binaria) no será necesario definir el número y límite de las variables por separado. La nueva lista de parámetros queda de la siguiente forma:

- Número de variables
- Límites de las variables
- Número de funciones objetivo
- Funciones objetivo

Para fines de esta tesis no se implementará ningún esquema para el manejo de restricciones por lo que el número de restricciones es igual a cero.

En cuanto a los límites de las variables reales, siempre se manejarán variables reales acotadas.

El número de bits para cada variable binaria se determina con los límites de cada variable y una precisión de ocho decimales.

La semilla de aleatorios será definida de la misma forma, es decir, como argumento de entrada otorgado por el usuario.

El indice de distribución de la cruza de las variables reales mantendrá el valor de uno y al indice de distribución de la mutación de la variables reales se le asignó el valor de cincuenta.

La probabilidad de mutación de las variables reales y la probabilidad de mutación binaria se conjuntarán en un solo argumento probabilidad de mutación que contendrá ambas codificaciones, quedando los parámetros restantes como:

- Tamaño de población
- Número de generaciones
- Probabilidad de cruza
- Probabilidad de mutación

- Tipo de cruza para las variables binarias

A la lista anterior le agregaremos los siguientes parámetros:

- Tipo de mutación para las variables reales
- Tipo de cruza para las variables reales
- Tipo de codificación

4.2 Análisis de sensibilidad de los parámetros

En estadística y en el diseño de experimentos se utiliza el análisis de varianza (ANOVA [56]) para comparar si los valores de un conjunto de datos numéricos son significativamente distintos a los valores de otro conjunto de datos. El procedimiento para comparar estos valores se basa en la varianza global presentada en los conjuntos de datos numéricos comparados.

Para obtener el conjunto de datos a comparar se tomaron en cuenta las siguientes funciones de prueba:

- ZDT3 por ser discontinua y bidimensional
- ZDT4 por ser no convexa y tener múltiples frentes
- DTLZ5 por no tener una distribución uniforme del verdadero frente de Pareto y por tener tres objetivos
- DTLZ6 por requerir un número elevado de evaluaciones para obtener un conjunto aceptable de soluciones que pertenezcan al verdadero frente de Pareto
- DTLZ7 por ser discontinua y tener tres objetivos

En el apéndice A se encuentra la definición de cada uno de los problemas antes mencionados. Además, se puede observar de forma gráfica el verdadero frente de Pareto de cada uno de estos problemas en el apéndice B.

En cuanto a los parámetros, se utilizaron los siguientes valores:

- Tamaño de población: 100, 200 y 500 individuos

- Número de generaciones: 100, 200 y 500
- Probabilidad de cruza: 0.5, 0.7 y 1.0
- Probabilidad de mutación: 0.001, 0.1 y 0.3
- Tipo de representación: binaria y real
- Tipo de cruza:
  - Binaria: dos puntos y uniforme
  - Real: SBX [44] y uniforme
- Tipo de mutación:
  - Binaria: única
  - Real: parameter-based [44] y de límite

El análisis se realizó con 5 funciones y 486 combinaciones diferentes según los parámetros antes listados. Se realizaron 20 ejecuciones diferentes por cada problema y cada combinación de parámetros.

Para evaluar los resultados se usaron las siguientes métricas: distancia generacional invertida [92] (DG Invertida) y el indicador ϵ-unario [106], comparando los resultados obtenidos con cada una de las combinaciones contra el verdadero frente de Pareto. La definición de las métricas adoptadas se encuentra en el apéndice C.

El ANOVA arrojará el valor del estadístico F y su nivel de significación (P-value). El nivel de significación nos va a permitir saber si aceptar o rechazar la hipótesis sin la necesidad de tener que comparar el valor de la F con su valor real de las tablas estadísticas de una F de Snedecor.

Los resultados determinarán si nuestra hipótesis (que diferentes valores de un parámetro no alteran el comportamiento del algoritmo) es verdadera o falsa. Un valor en el nivel de significación por debajo de 0.05 nos indica que los n tipos de pruebas son diferentes entre sí. Es decir, la combinación realmente afecta al algoritmo, o por lo menos alguna de ellas hace diferencia con respecto a las otras. Si el valor es por arriba de 0.05, entonces son iguales entre sí. Es decir, si se cambia el parámetro no afecta al algoritmo, y éste se sigue comportando de manera muy similar, sin cambios significativos.

En las siguientes tablas presentaremos el resultado del ANOVA con el problema tratado, la métrica usada (DG Invertida y ϵ-unario), el parámetro

analizado (el que queremos saber si afecta o no el comportamiento del algoritmo de forma significativa), el valor del estadístico F, el nivel de significación (P-value) y por último si afecta o no al algoritmo ("SI" indica que modificar el parámetro SI afecta al algoritmo, y "NO" indica que el valor puede ser fijado).

Problema	Métrica	F	P-value	¿Afecta?
ZDT3	DG Invertida	34.87478029	6.62799E-09	SI
ZDT4		2.492786256	0.115022392	**NO**
DTLZ5		4.749960781	0.029780338	SI
DTLZ6		868.0226074	4.892E-110	SI
DTLZ7		348.9248851	4.93133E-59	SI
ZDT3	ϵ-unario	26.17193957	4.51033E-07	SI
ZDT4		2.480493376	0.115919485	**NO**
DTLZ5		1.659212494	0.198325157	**NO**
DTLZ6		836.7987544	1.406E-107	SI
DTLZ7		447.5764703	7.92877E-71	SI

Tabla 4.1: Parámetro: codificación

En la tabla 4.1, observamos los resultados de emplear los dos tipos de codificación usados (real y binaria), y observamos como solo en tres casos el algoritmo no muestra ser sensible al cambio, sin embargo, en los siete casos restantes si muestra ser sensible.

En la tabla 4.2 se encuentran los resultados del parámetro número de generaciones, para este parámetro se utilizaron tres valores (100, 200 y 500 generaciones), en este caso para las funciones ZDT3, ZDT4 y DTLZ7 el algoritmo muestra no ser sensible al parámetro, sin embargo, para las funciones DTLZ5 y DTLZ6 si es sensible.

En la tabla 4.3 se observa como en el problema DTLZ5 para la métrica DG invertida el algoritmo muestra ser sensible al parámetro, sin embargo, las los nueve casos restantes no muestra ser sensible. El parámetro de la probabilidad de cruza es una candidato a ser fijado, ya que en la gran mayoría de los casos, el algoritmo no muestra un comportamiento diferente con valores diferentes del parámetro.

En la tabla 4.4 se encuentran los resultados del parámetro de probabilidad de mutación, donde sucede el caso totalmente opuesto al de la probabilidad

Problema	Métrica	F	P-value	¿Afecta?
ZDT3	DG Invertida	0.159715813	0.852430991	**NO**
ZDT4		0.664785076	0.514854077	**NO**
DTLZ5		3.269202096	0.038880125	SI
DTLZ6		7.932579135	0.000407658	SI
DTLZ7		1.761732218	0.17284891	**NO**
ZDT3	ϵ-unario	0.072223723	0.930332783	**NO**
ZDT4		0.665703386	0.514382796	**NO**
DTLZ5		3.389638862	0.034525019	SI
DTLZ6		8.308410126	0.00028339	SI
DTLZ7		1.656421492	0.191902715	**NO**

Tabla 4.2: Parámetro: número de generaciones

Problema	Métrica	F	P-value	¿Afecta?
ZDT3	DG Invertida	0.041533938	0.959320203	**NO**
ZDT4		0.181330645	0.834216263	**NO**
DTLZ5		3.226970565	0.040534533	SI
DTLZ6		0.132470018	0.875961009	**NO**
DTLZ7		0.179257526	0.835946193	**NO**
ZDT3	ϵ-unario	0.001363078	0.998637855	**NO**
ZDT4		0.181252697	0.834281242	**NO**
DTLZ5		2.85145982	0.058732787	**NO**
DTLZ6		0.148864195	0.861725648	**NO**
DTLZ7		0.240557745	0.786283382	**NO**

Tabla 4.3: Parámetro: probabilidad de cruza

Problema	Métrica	F	P-value	¿Afecta?
ZDT3	DG Invertida	7.457717294	0.000645888	SI
ZDT4		27.564213	4.61614E-12	SI
DTLZ5		47.59329407	1.36084E-19	SI
DTLZ6		3.513509012	0.030555953	SI
DTLZ7		45.810976	6.05937E-19	SI
ZDT3	ϵ-unario	3.000745597	0.050678385	**NO**
ZDT4		27.5813411	4.54572E-12	SI
DTLZ5		57.53999106	3.85362E-23	SI
DTLZ6		3.48807925	0.031331561	SI
DTLZ7		35.3022874	4.90356E-15	SI

Tabla 4.4: Parámetro: probabilidad de mutación

de cruza, ya que en nueve casos el algoritmo muestra ser sensible al parámetro de probabilidad de mutación.

En la tabla 4.5 observamos al parámetro de tamaño de población, donde sólo en tres ocasiones el algoritmo muestra no ser sensible al parámetro, por lo que no candidato a ser fijado.

En la tabla 4.6 observamos al tipo de cruza, que también muestra ser candidato para ser fijado. El algoritmo muestra ser sensible sólo en dos ocasiones, mientras que en las ocho restantes no es sensible al parámetro.

En la tabla 4.7 está el tipo de cruza real que sólo en tres ocasiones muestra no ser sensible, por lo que no es candidato a ser fijado.

En la tabla 4.8 encontramos los resultados del último parámetro, el tipo de mutación real, que sólo en un caso muestra que el algoritmo no es sensible al parámetro.

4.3 Resultados del análisis de sensibilidad

Después de haber realizado el ANOVA correspondiente, y como observamos en la tabla 4.3 (resultados de la probabilidad de cruza) y en la tabla 4.6 (resultados del tipo de cruza binaria) que no existen cambios significativos en el uso de diferentes valores para dichos parámetros en la mayoría de las funciones de prueba y métricas.

Problema	Métrica	F	P-value	¿Afecta?
ZDT3	DG Invertida	59.11759442	1.08146E-23	SI
ZDT4		6.544976319	0.001568151	SI
DTLZ5		2.091828852	0.124578242	**NO**
DTLZ6		4.103931166	0.017086764	SI
DTLZ7		1.682667334	0.186965275	**NO**
ZDT3	ϵ-unario	57.67345921	3.45994E-23	SI
ZDT4		6.543875286	0.001569833	SI
DTLZ5		3.79158528	0.023234223	SI
DTLZ6		5.049338871	0.00675632	SI
DTLZ7		2.018892643	0.133921616	**NO**

Tabla 4.5: Parámetro: tamaño de población

Problema	Métrica	F	P-value	¿Afecta?
ZDT3	DG Invertida	0.628778236	0.42897808	**NO**
ZDT4		2.897025077	0.090685349	**NO**
DTLZ5		0.93154176	0.335919153	**NO**
DTLZ6		6.127515746	0.01435142	SI
DTLZ7		0.91010041	0.341526833	**NO**
ZDT3	ϵ-unario	0.008283112	0.92759721	**NO**
ZDT4		2.899757018	0.090534695	**NO**
DTLZ5		1.945848751	0.164968326	**NO**
DTLZ6		6.394313494	0.012416893	SI
DTLZ7		1.302137746	0.255528163	**NO**

Tabla 4.6: Parámetro: tipo de cruza binaria

Problema	Métrica	F	P-value	¿Afecta?
ZDT3	DG Invertida	136.1209619	1.79919E-26	SI
ZDT4		17.30052552	4.09739E-05	SI
DTLZ5		70.06187541	1.77974E-15	SI
DTLZ6		1.64339313	0.200782785	**NO**
DTLZ7		8.546402654	0.003707294	SI
ZDT3	ϵ-unario	71.97534192	8.04069E-16	SI
ZDT4		17.29591876	4.10683E-05	SI
DTLZ5		55.36522757	9.18533E-13	SI
DTLZ6		2.541753787	0.111852618	**NO**
DTLZ7		0.79185669	0.374203635	**NO**

Tabla 4.7: Parámetro: tipo de cruza real

Problema	Métrica	F	P-value	¿Afecta?
ZDT3	DG Invertida	37.94118121	2.17288E-09	SI
ZDT4		20.0193809	1.06467E-05	SI
DTLZ5		159.6929248	5.29099E-30	SI
DTLZ6		87.98994064	1.21725E-18	SI
DTLZ7		0.233774546	0.62906769	**NO**
ZDT3	ϵ-unario	22.19544665	3.66832E-06	SI
ZDT4		20.03643122	1.05578E-05	SI
DTLZ5		163.3944616	1.5312E-30	SI
DTLZ6		102.9249321	3.61412E-21	SI
DTLZ7		9.532887032	0.002193704	SI

Tabla 4.8: Parámetro: tipo de mutación real

Problema	Métrica	$P_c = 0.5$	$P_c = 0.7$	$P_c = 1.0$
ZDT3	DG Invertida	**0.000880**	0.000881	0.000912
	ϵ-unario	1.013469	**1.013347**	1.013483
ZDT4	DG Invertida	5.708135	**5.708120**	5.708155
	ϵ-unario	74.886258	**74.886066**	74.886516
DTLZ5	DG Invertida	0.000626	**0.000608**	0.000831
	ϵ-unario	1.039501	**1.038434**	1.050836
DTLZ6	DG Invertida	0.025826	**0.025127**	0.026730
	ϵ-unario	2.152371	**2.117837**	2.192020
DTLZ7	DG Invertida	0.008151	**0.007885**	0.008816
	ϵ-unario	1.350153	**1.338782**	1.379305

Tabla 4.9: Valores promedio de cada métrica por problema del parámetro de probabilidad de cruza

Observamos en la tabla 4.3 como en nueve de las diez ocasiones el algoritmo muestra no ser sensible a cambios en el valor del parámetro probabilidad de cruza, lo que sugiere fijar el valor a implementar alguna técnica de auto-adaptación. Para elegir el mejor valor, observamos detalladamente los valores arrojados por cada métrica en cada uno de los problemas. En la tabla 4.9 observamos en cada renglón la comparación de los tres valores arrojados por la métrica y según el problema, donde cada valor es el promedio de la métrica para las diferentes ejecuciones. El valor sombreado es el que mejor entre los tres. De la tabla 4.9 observamos que el mejor valor para la probabilidad de cruza es de 0.7, ya que en nueve de las diez ocasiones muestra un desempeño superior a los valores de 0.5 y 1.0.

En la tabla 4.6 se observa como en ocho de las diez ocasiones el algoritmo muestra no ser sensible a cambios en el valor del parámetro tipo de cruza binaria, lo que sugiere fijar el valor a implementar alguna técnica de auto-adaptación. Para elegir que tipo de cruza binaria será la empleada, observamos detalladamente los valores arrojados por cada métrica en cada uno de los problemas. En la tabla 4.10 observamos en cada renglón la comparación de los dos valores arrojados por la métrica y según el problema, donde cada valor es el promedio de la métrica para las diferentes ejecuciones. El valor sombreado es el que mejor entre los tres. De la tabla 4.10 observamos que el mejor tipo de cruza binaria es la de dos puntos, ya que en nueve de

Problema	Métrica	Dos puntos	Uniforme
ZDT3	DG Invertida	0.000501	0.000501
	ϵ-unario	**1.005169**	1.005229
ZDT4	DG Invertida	**5.708054**	5.708114
	ϵ-unario	**74.885217**	74.885985
DTLZ5	DG Invertida	**0.000525**	0.000607
	ϵ-unario	**1.034995**	1.042265
DTLZ6	DG Invertida	**0.053067**	0.062239
	ϵ-unario	**3.325114**	3.732251
DTLZ7	DG Invertida	**0.020038**	0.022874
	ϵ-unario	**1.826628**	1.946154

Tabla 4.10: Valores promedio de cada métrica por problema del parámetro de tipo de cruza binaria

las diez ocasiones muestra un desempeño superior al tipo de cruza binaria uniforme.

5

Técnicas de Auto-adaptación

En esta capítulo se explicarán las diferentes técnicas de auto-adaptación implementadas. Comenzaremos por presentar el algoritmo de forma global, para posteriormente dar una explicación más específica y detallada de sus diferentes procesos.

Resulta complicado intentar separar cada uno de los parámetros de un algoritmo evolutivo multi-objetivo y explicar cómo se realizó e implementó su auto-adaptación correspondiente. Es por ello que se explica de forma global todo el proceso y, después, se desglosan de una forma más específica las operaciones y procedimientos realizados.

5.1 Algoritmo general

De una forma general, el comportamiento del algoritmo se puede observar en el algoritmo 4. Debemos recordar que la base de éste es el NSGA-II [24], por lo que es muy similar y su forma de operación conserva la naturaleza de dicho algoritmo. Simplemente se le agregaron operaciones y procedimientos que auto-adaptan los siguientes parámetros:

- Probabilidad de mutación

- Tipo de representación
- Tipo de cruza
- Tipo de mutación
- Tamaño de población
- Número máximo de generaciones

Algoritmo 4 NSGA-II con las técnicas de auto-adaptación implementadas

Entrada: Funciones objetivo, número de funciones objetivo, número de variables de decisión, límite de las variables de decisión
Salida: Conjunto de soluciones

```
1: Inicializar población de forma aleatoria;
2: Decodificar individuos;
3: Evaluar la población;
4: Jerarquización de la población
5: repeat
6:     Seleccionar padres (misma codificación);
7:     Cruzar padres;
8:     Mutar hijos;
9:     Decodificación de hijos;
10:    Evaluar la población de hijos;
11:    Unión de la población de padres con la de hijos;
12:    Elitismo
13:    Herencia - Fertilización
14:    if Generación Actual ≥ 100 then
15:        Análisis del Hipervolumen
16:        Agregar-Eliminar Individuos
17:    end if
18: until No haya mejoras según el hipervolumen
```

Como parámetros de entrada (es decir, la información que el usuario debe proporcionar) están las funciones objetivo, el número de estas funciones, el número de las variables de decisión, así como los límites de cada una de ellas.

El algoritmo inicia con una población de 100 individuos. La primera población (de 100 individuos) se genera de forma totalmente aleatoria; es decir,

todos los valores son establecidos de forma aleatoria. Se elige de forma aleatoria la codificación del individuo. Para el caso de individuos reales, se elige un valor real de forma aleatoria entre los límites de la variable. Además, se establece una probabilidad de mutación de forma aleatoria la cual se encuentre entre los límites $(0.001, 0.03)$. También se elegirá un tipo de cruza y un tipo de mutación de forma totalmente aleatoria. Si la codificación del individuo es binaria, entonces sólo se inicializa de forma aleatoria al cromosoma de las variables de decisión y al de la probabilidad de mutación.

El siguiente paso es decodificar las variables de decisión. Para el caso de la codificación real no se requiere la decodificación ya que los valores cromosómicos se usan directamente para evaluar las funciones objetivo. Para los individuos con codificación binaria se realiza la decodificación pertinente, a fin de poder evaluar las funciones objetivo.

El siguiente paso es el de evaluar a cada uno de los individuos de la población en cada una de las funciones objetivo. Posteriormente, se jeraquiza la población de forma idéntica a como lo realiza el NSGA-II.

En este momento comienza el proceso iterativo, donde la selección de los individuos es la primera operación a realizar. La selección adoptada es idéntica a la utilizada por el NSGA-II.

En cuanto a los operadores de cruza y mutación existen ciertos cambios, los cuales se explicarán más adelante en detalle, así como la forma en como se aplican cada uno de los operadores según las circunstancias.

Al terminar la cruza y mutación se procede a decodificar a las individuos hijos que cuentan con la codificación binaria, a fin de poder evaluarlos en las funciones objetivo. Después, se unen las dos poblaciones (padres e hijos) para formar una sola y seleccionar a los mejores individuos de dicha población. Este procedimiento se realiza sin cambio alguno, de forma idéntica a la adoptada en el NSGA-II original.

La operación de *Herencia - Fertilización* se explicará más adelante, ya que no es parte del NSGA-II original, y pretende agregar diversidad a la población, principalmente a los operadores y su aplicación.

El análisis de hipervolumen se inicia a partir de la generación 100, y es el criterio de paro del algoritmo. Monitorear la métrica del hipervolumen (Apéndice C) nos servirá para saber si vale la pena continuar iterando o si ya no existen cambios significativos. Además, nos permitirá saber en qué momento agregar nuevos individuos a la población a fin de aumentar su diversidad.

5.2 El individuo

Cada individuo contendrá la siguiente información:

- Variables de decisión (real o binaria)
- Probabilidad de mutación (real o binaria)
- Tipo de cruza
- Tipo de mutación
- Tipo de codificación
- Padres
- Fertilidad

	Individuo E	Individuo F
Codificación	Real	Binaria
Variables de decisión	$\overbrace{1.5}^{x_1}, \overbrace{-0.7}^{x_2}, \overbrace{2.3}^{x_3}$	$\overbrace{001}^{x_1}\ \overbrace{000101}^{x_2}\ \overbrace{1101}^{x_3}$
Probabilidad de Mutación	{0.01}	{0001011}
Tipo de cruza	Intermedia	-
Tipo de mutación	De límite	-
Padres	{Individuo A, Individuo B}	{Individuo C, Individuo D}
Fertilidad	2	0

Figura 5.1: Ejemplo de individuo con codificación real y codificación binaria.

Cada individuo se codificará de una forma, ya sea binaria o real, que a lo largo del proceso evolutivo puede cambiar. Más adelante se explicará el procedimiento para cambiar de una codificación a otra. En la figura 5.1 se

muestran dos individuos con diferente codificación, el individuo E tiene la codificación real, y el individuo F tiene la codificación binaria.

El individuo consta de un cromosoma para codificar las variables de decisión, las cuales se pueden codificar de forma binaria o real, según se desee. En el ejemplo, el individuo E con codificación real cuenta con tres variables de decisión, mientras que el individuo F cuenta con un cromosoma que internamente contiene a las tres variables de decisión, que deben ser decodificadas para llevar a cabo la evaluación de las funciones objetivo.

Para la codificación binaria la precisión es de ocho decimales, es decir, cada una de las variables de decisión se codificará con una precisión de ocho dígitos después del punto decimal. Para fines didácticos, en el ejemplo de la figura 5.1 no se presenta una precisión de ocho decimales, ya que la longitud del cromosoma resultante no permitiría apreciar de correctamente el resto de la información que en él presentamos.

Cada individuo contendrá un cromosoma adicional al de las variables de decisión, el cual corresponde a la probabilidad de mutación. Para el individuo E la probabilidad de mutación está codificada de forma real, mientra que para el individuo F, por tener codificación binaria, su probabilidad de mutación está codificada de forma binaria también.

El límite inferior de la probabilidad de mutación es de 0.001 y el límite superior de 0.3. Cada individuo cuenta con su probabilidad de mutación y no existe una probabilidad para toda la población. Para el caso de la codificación binaria de la probabilidad de mutación se asignaron quince bits para el tamaño del cromosoma. Para fines didácticos, en el ejemplo se presenta un cromosoma de sólo siete bits.

En total, el individuo tendrá dos cromosomas, los cuales serán afectados por los operadores genéticos (la cruza y la mutación). Los hijos generados tendrán la misma cantidad de variables de decisión que sus padres, incluyendo el cromosoma adicional que codifica la probabilidad de mutación.

En cuanto al tipo de cruza, sólo es relevante en el individuo E, ya que el individuo F tiene codificación binaria, y para esa codificación consideramos un solo tipo de cruza (de dos puntos). Cada individuo con codificación real, como el individuo E, tendrá codificado el tipo de cruza. Este valor sólo servirá para el caso en que la codificación sea de tipo real, ya que para la codificación binaria el tipo de cruza será siempre de dos puntos. Este valor indica el tipo de cruza con la que el individuo fue generado. Cuando un individuo es seleccionado como padre, el tipo de cruza servirá para definir, junto con el tipo de cruza del otro padre, con qué cruza se generarán los nuevos

individuos.

Para el tipo de cruza para representación real se implementaron cinco tipos diferentes: SBX [44], simple, uniforme, intermedia y dos puntos. Para mayores detalles sobre estas cruzas se puede consultar el Apéndice D.

Para el tipo de mutación, sólo es relevante para los individuos con codificación real, porque para el caso binario sólo se considera la mutación bit por bit tradicional. El tipo de mutación indicará cual será la mutación que afectará al individuo. Los tipos de mutación para representación real implementados son cuatro: parameter-based mutation, no uniforme, uniforme y de límite. Para mayor detalle sobre estos tipos de mutación se puede revisar el Apéndice D en la página 121.

En cuanto al indicador *Padres*, nos servirá para saber quiénes fueron los padres que crearon a este individuo, ya que existe el procedimiento que denominamos *Herencia-Fertilización*, que más adelante se explicará a detalle, y requeriremos saber quiénes fueron los padres que generaron a este nuevo individuo. Debemos recalcar que en cuanto se realiza la fusión de las poblaciones y todos los nuevos individuos se vuelven padres, el indicador ya no será necesario, ya que es sólo para uso de los hijos (nuevos individuos). Este parámetro se encuentra en ambos tipos de individuo, tanto en el E como en el F, sin importar el tipo de codificación.

El indicador de *Fertilidad*, por el contrario que el de *Padres*, es utilizado por los padres de la población, y nos indicará el número de generaciones que el individuo lleva sin aportar hijos que hayan sobrevivido a la siguiente generación. Este parámetro se encuentra en ambos tipos de individuo, tanto en el E como en el F, sin importar el tipo de codificación. Más adelante se explicará a detalle el procedimiento de modificación de parámetros.

Además de estos parámetros, se usan aquellos que el NSGA-II original define: la jerarquía (*rank*), la distancia de agrupamiento (*crowding distance*).

5.3 Operador de cruza

Como la selección nos arroja parejas de individuos con la misma codificación, podemos diferenciar claramente dos casos de cruza: cruza binaria y cruza de números reales.

Para los individuos con codificación binaria se utiliza la cruza de dos puntos. Sin embargo, debemos recordar que tenemos dos cromosomas, por lo que se realiza la cruza de dos puntos en el cromosoma de las variables

de decisión, y después en el cromosoma de la probabilidad de mutación. Los individuos hijos adoptarán codificación binaria.

En la figura 5.2 podemos observar cómo es que los individuos C y D tienen dos cromosomas, y la cruza se hace en cada uno de ellos. Para tal fin, se eligen dos puntos para cruzar las variables de decisión, y posteriormente se eligen dos nuevos puntos sobre el cromosoma de la probabilidad de mutación para realizar la cruza de la probabilidad de mutación. Los individuos resultantes son el I y J, que preservan la codificación binaria y además se muestran los padres de cada individuo.

Para los individuos con codificación real debemos elegir el tipo de cruza a realizar, ya que pueden no coincidir los padres en el tipo de cruza. Para elegir el tipo de cruza se realiza una función $flip(p)$[1] con $p = 0.9$. Si es verdadero entonces se elige al mejor padre (al de mejor jerarquía o *rank*); en caso contrario, se realizará $flip(0.5)$. Si obtenemos verdadero se elegirá el primer padre, y si es falso, se elegirá al otro. En caso que los dos padres tengan la misma jerarquía, el criterio de desempate será realizar $flip(0.5)$ para decidir a qué padre elegir.

Los hijos recibirán la codificación real y además recibirán la cruza con la que fueron creados (es decir, el tipo de cruza del padre elegido).

La cruza se realiza por separado para cada uno de los cromosomas (variables de decisión y probabilidad de mutación).

Como la probabilidad de mutación es sólo una variable más de decisión, para la cruza simple, de dos puntos y uniforme no es posible elegir un punto de cruza, por lo que se realiza un $flip(0.5)$ para saber si se intercambian o no los valores. En cuanto a la cruza intermedia asignamos el valor de $k = 0$ para realizar la cruza. La cruza SBX se aplica de forma natural.

En cuanto a qué tipo de mutación será la que los hijos tendrán, simplemente se realiza un $flip(0.5)$. Si se obtiene verdadero, el primer padre le otorga su mutación al primer hijo, y el segundo padre al segundo hijo. En caso de un valor falso, entonces el primer padre otorga al segundo hijo y el segundo padre al primer hijo.

En la figura 5.3 observamos la cruza del individuo A con el individuo B, para crear al individuo G y al individuo H.

El primer proceso que se ilustra en la figura 5.3, es decidir qué tipo de cruza se debe realizar. El individuo A tiene la cruza intermedia, mientras que el individuo B tiene la cruza uniforme. Para decidir qué tipo de cruza se

[1]Es una función que regresa un valor de verdadero con una probabilidad de p

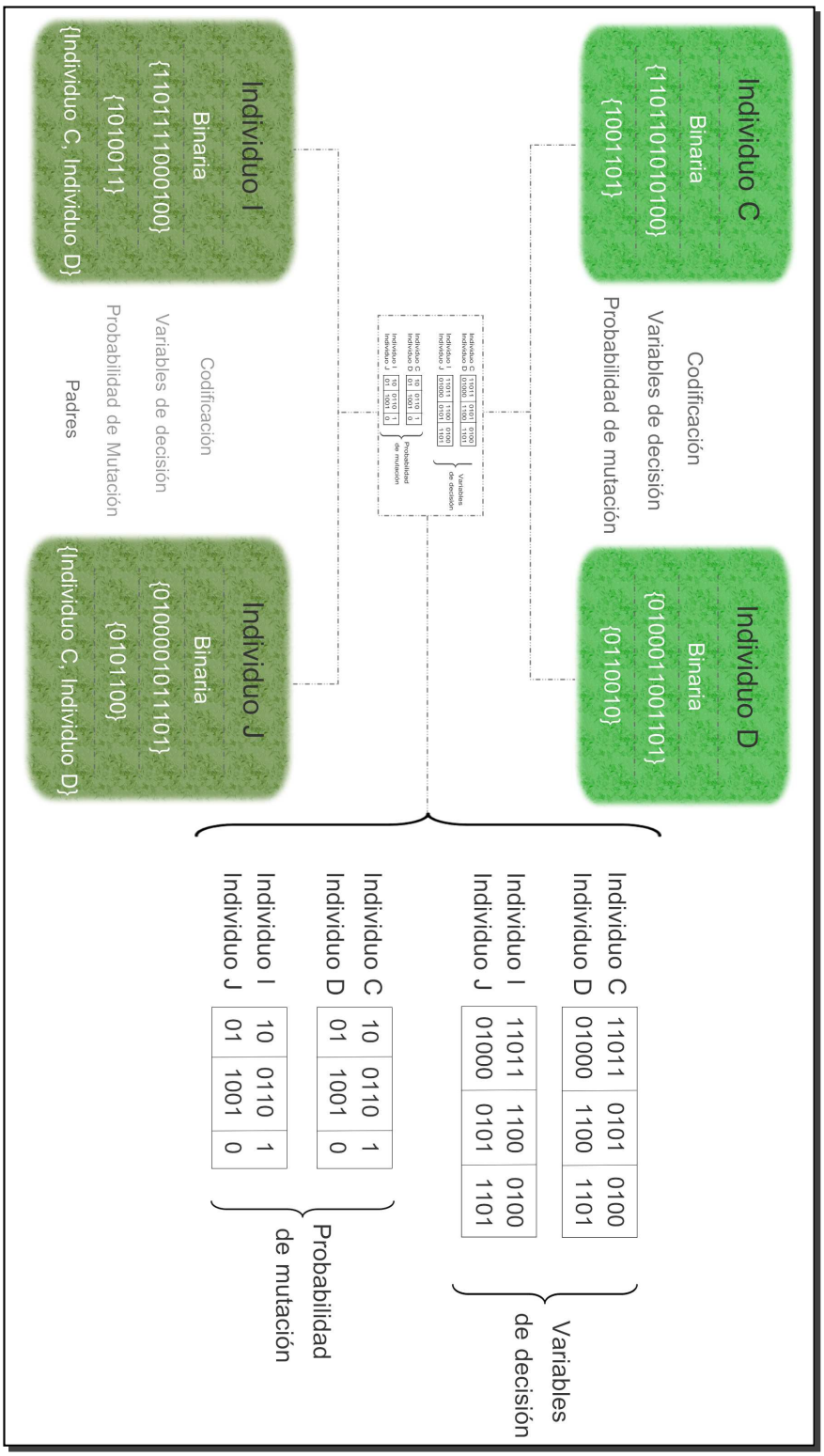


Figura 5.2: Ejemplo de cruza de individuos con codificación binaria.

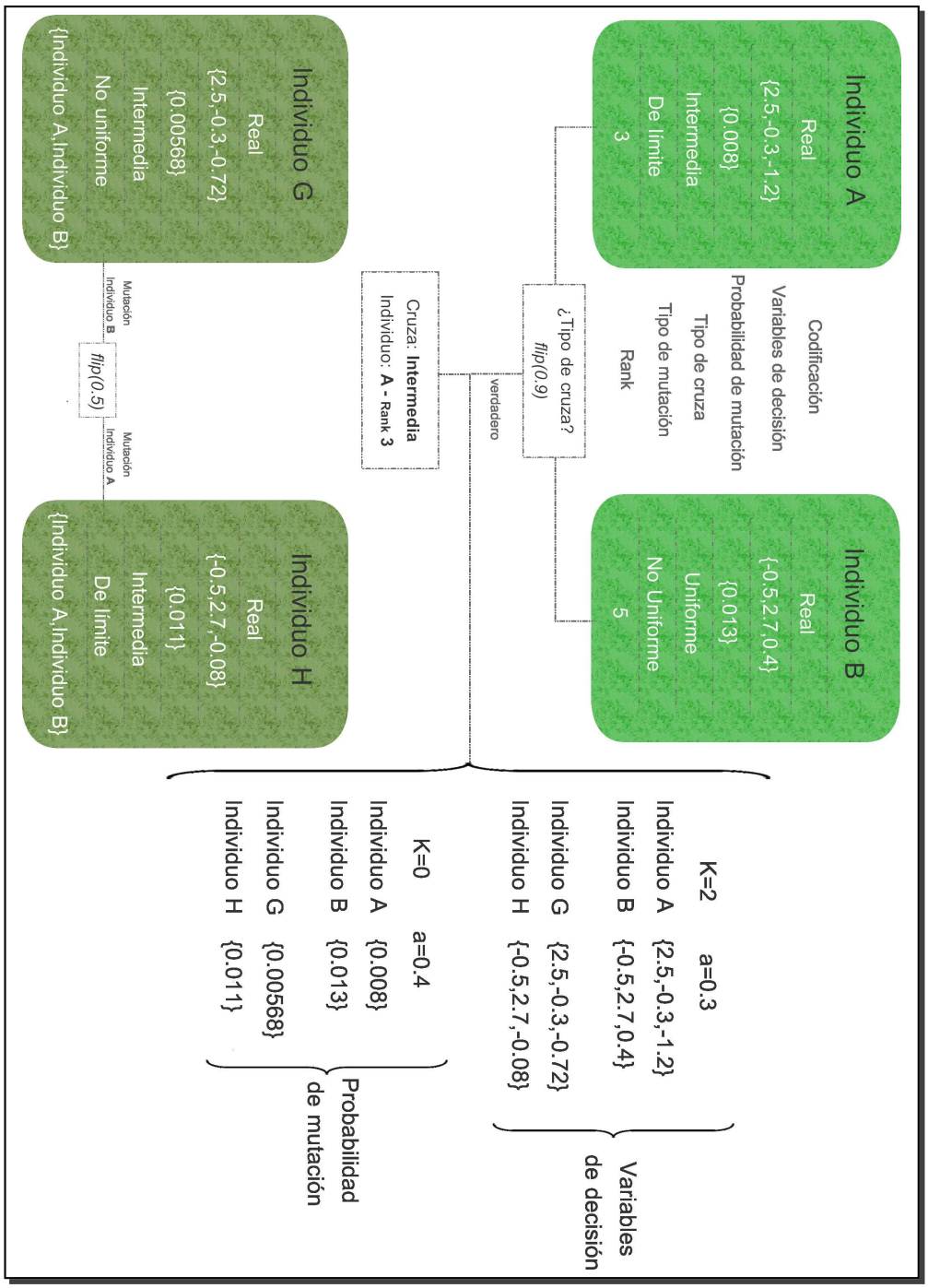


Figura 5.3: Ejemplo de cruza de individuos con codificación real.

realizará, se efectúa un $flip(0.9)$. En este caso, arrojó verdadero y se procede a realizar la cruza del individuo con el mejor *rank*, que en este caso es el individuo A, por lo que se realiza la cruza intermedia.

En el ejemplo de la figura 5.3 se obtienen los valores de $k = 2$ y $a = 0.3$ de forma aleatoria para la cruza de las variables de decisión. En cuanto a la probabilidad de mutación, que es otro cromosoma, se eligen los valores de $k = 0$ y $a = 0.4$ (el primero se definió así porque solamente es una variable, y el segundo se obtiene de manera aleatoria) para la cruza de la probabilidad de mutación.

Los dos nuevos individuos tendrán la codificación real y la cruza intermedia. Para el tipo de mutación se realiza un $flip(0.5)$. En este caso a la mutación del individuo B se le otorga al individuo G, mientras que la mutación del individuo A se le otorga al individuo H.

A los nuevos individuos se les establece los padres que los generaron, de la misma forma como se hizo en la cruza binaria.

Al tener una probabilidad de cruza fija de 0.7, tenemos que el 30 % de los casos no se realiza cruza, sino que simplemente se copian los padres, manteniendo su codificación, su tipo de cruza (en el caso que sean individuos con codificación real), variables de decisión y probabilidad de mutación.

5.4 Operador de mutación

Para este operador también tenemos dos opciones: el caso binario y el caso real.

Para los individuos con codificación binaria la probabilidad de mutación está entre $(0.001, 0.3)$. Este valor ya está contenido en el individuo, por lo que es necesario decodificarlo y saber el valor de la probabilidad de mutación que se le aplicará.

Teniendo ya el valor de la probabilidad de mutación, se muta el cromosoma de las variables de decisión y después se procede con el cromosoma de la probabilidad de mutación.

En la figura 5.4, observamos cómo se toma la probabilidad de mutación del individuo C, la cual decodifica, para saber cuál es el número real correspondiente. Después, se realiza la mutación en el cromosoma de las variables de decisión y se prosigue con la probabilidad de mutación. Posteriormente, se realiza la mutación sobre el parámetro de codificación. En el caso del individuo C no hubo mutación, pero en el ejemplo del individuo D de la misma

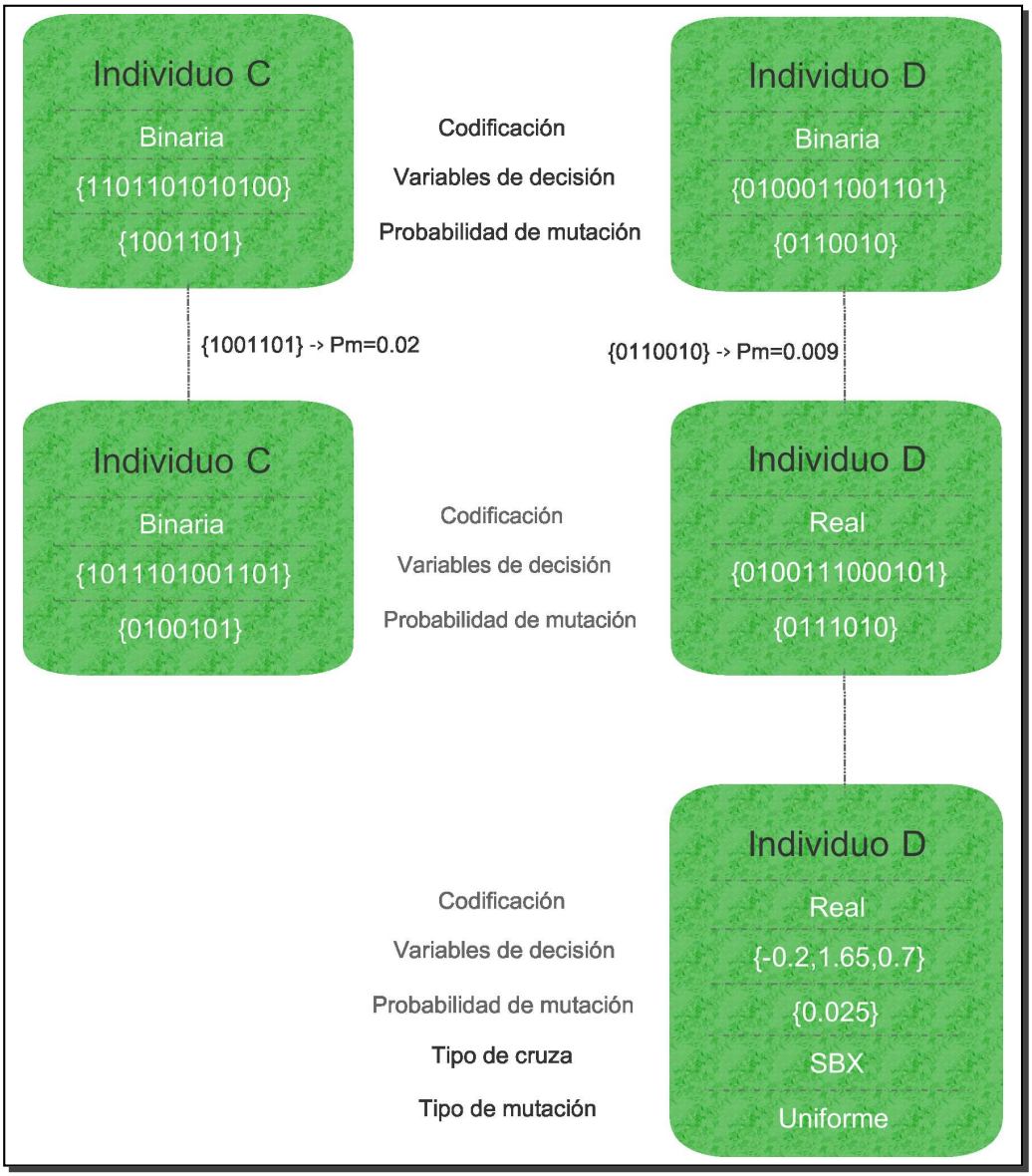


Figura 5.4: Ejemplo de mutación en dos individuos con codificación binaria.

figura la situación es diferente. Más adelante se explicará dicho ejemplo.

En cuanto al caso real que usa codificación real no podemos manejar los mismos porcentajes que los usados en el caso binario, ya que tenemos un número menor de variables reales en comparación con el número de genes existentes en la codificación binaria. Para ello se realiza un mapeo lineal de $(0.001, 0.3) \rightarrow (1/L, 0.5)$, donde L es el número de variables de decisión. Con esto se pretende que al menos una variable sea mutada y a lo más la mitad de éstas lo sea. En la siguiente ecuación podemos observar cómo es que se realiza dicho mapeo, donde p_m es la probabilidad de mutación del individuo:

$$p_m^{real} = \left(\left(\frac{0.5 - \frac{1}{L}}{0.3 - 0.001} \right) \times (p_m - 0.001) \right) + \frac{1}{L}$$

La probabilidad de mutación es tomada en cuenta como una variable más de decisión, por lo que no existe problema alguno para llevar a cabo los diferentes tipos de mutación real.

En cuanto a los demás parámetros (codificación, tipo de cruza y tipo de mutación), se realiza un mapeo similar al efectuado para la cruza de variables reales, es decir un mapeo lineal de $(0.001, 0.3) \rightarrow (1/8, 0.8)$.

$$p_m^{parámetros} = \left(\left(\frac{0.8 - \frac{1}{8}}{0.3 - 0.001} \right) \times (p_m - 0.001) \right) + \frac{1}{8}$$

Después de haber mutado las variables de decisión y la probabilidad de mutación (según el tipo de mutación indicada por el mismo individuo), se revisa si se muta la codificación (usando la probabilidad de mutación arrojada por el mapeo antes mencionado).

Si el individuo tiene una codificación binaria, se procede a cambiar las variables de decisión y la probabilidad a valores reales. En cuanto a los datos del tipo de cruza y el tipo de mutación, se les asigna un valor de forma aleatoria. En la figura 5.4, observamos el proceso cuando se muta la codificación de un individuo binario. Primero se realiza la decodificación de la probabilidad de mutación, se mutan las variables de decisión y las probabilidades de mutación de forma regular. Después se realiza la mutación de parámetros con el valor de mutación que nos arroja el mapeo. En este caso, el individuo D muta su codificación. Ahora, el individuo tiene una codificación real y es necesario agregar un tipo de cruza y un tipo de mutación de forma aleatoria. Además, las variables de decisión y la probabilidad de mutación cambian su codificación a números reales.

Sin embargo, si el individuo tiene una codificación real, se cambian sus variables de decisión y la probabilidad de mutación a la codificación binaria. Los datos de tipo de cruza y mutación se desechan.

Si el individuo con codificación real no muta su codificación, entonces se mutan los demás valores, es decir, el tipo de cruza y tipo de mutación. Para llevar a cabo la mutación de dichos valores, se aplica el $flip(p_m)$ correspondiente según la probabilidad de mutación derivada del mapeo. Si se debe aplicar la mutación al tipo de cruza o al tipo de mutación, simplemente se elige cualquiera de las opciones de forma aleatoria. Todo esto para el caso en que el individuo está codificado con números reales.

En la figura 5.5 se muestran dos ejemplos de mutación. Primero tenemos al individuo A que tiene una mutación de límite y una probabilidad de mutación de 0.008. El primer paso a realizar es el mapeo correspondiente de la mutación. Dicho proceso nos arroja una $p_m^{real} = 0.337$ y $p_m^{parámetros} = 0.140$. Después, se realiza la mutación sobre las variables de decisión (observamos que la tercera variable de decisión toma el valor de su límite inferior). Posteriormente, se realiza la mutación sobre la probabilidad de mutación, donde por medio de un $flip(0.5)$, que nos arrojó verdadero en este ejemplo, el valor de la probabilidad de mutación toma el valor del límite superior.

Por último se realiza la mutación de los parámetros. El primer parámetro revisado para saber si se llevará a cabo o no la mutación es la codificación. En este caso, no se mutó. Continuamos con el tipo de cruza, que no se muta. Por último se analiza el tipo de mutación, que sí se muta, por lo que se elige un tipo de mutación de forma aleatoria.

En la misma figura 5.5 tenemos al segundo ejemplo, el individuo B, que realiza el mapeo de la probabilidad de mutación y muta sus variables de decisión y probabilidad de mutación según lo dicta el mismo individuo, de la misma forma como la realizó el individuo A. Sin embargo, al analizar la codificación para saber si se va a mutar o no, ésta sí se muta, por lo que su valor cambia a binario. En el siguiente paso, al tener ahora el individuo B la codificación binaria, se deben codificar las variables de decisión y la probabilidad de mutación de forma binaria, y se desechan el tipo de cruza y el tipo de mutación.

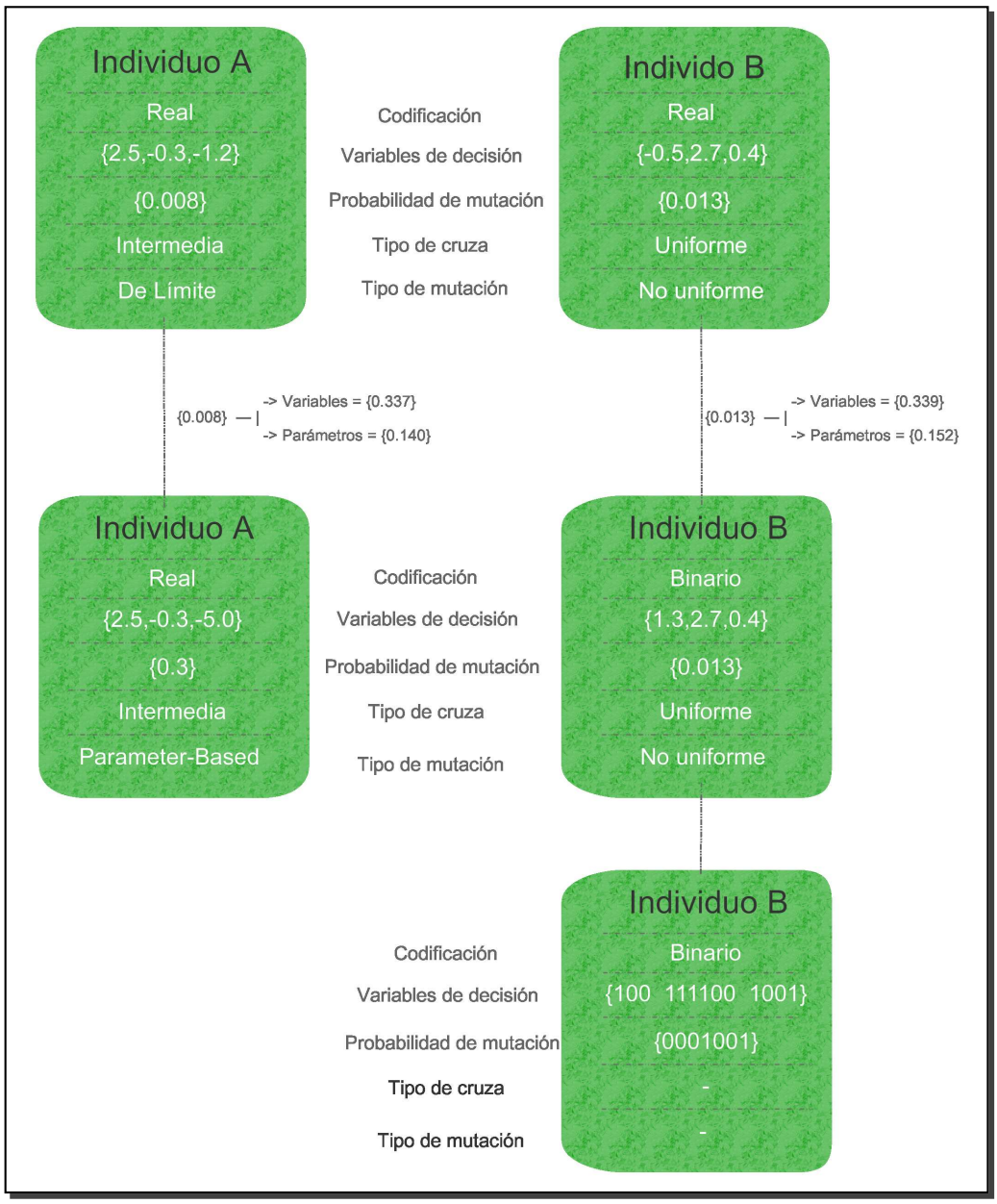


Figura 5.5: Ejemplo de mutación en dos individuos con codificación real.

5.5 Herencia - fertilización

En esta operación se revisa a la población de padres. Aunque la población de padres ya está fusionada con la de hijos, se debe tener bien definido cuáles son los hijos y cuáles son los padres.

Lo primero que se hace es actualizar los padres que han generado un hijo que se encuentra en la población resultante. Es decir, un hijo que haya sobrevivido. Si existe un padre que no haya generado al menos un hijo sobreviviente en las últimas cinco generaciones, se mutan sus parámetros.

La mutación de los parámetros se realizará según las características del individuo. Si el individuo tiene codificación real entonces su probabilidad de mutación se mutará según el tipo de mutación real que indique el mismo. Si la codificación es binaria, se aplica mutación binaria. Esto se observa gráficamente en las figuras 5.4 y 5.5.

En cuanto a los parámetros de codificación, tipo de cruza y tipo de mutación se mutan de la misma forma indicada en la sección 5.4. De la misma forma, si la codificación cambia, se debe de realizar el cambio en las variables de decisión y la probabilidad de mutación. Esto se muestra en la figura 5.5, donde se tienen dos ejemplos de individuos con codificación real y se indica la forma en que se lleva a cabo la mutación.

Nótese que las variables de decisión permanecen intactas, ya que recordemos que los padres son buenos individuos que siguen activos en el proceso evolutivo por la buena aptitud que presentan.

Este operador se introduce para dar diversidad a la población, ya que el que no se estén generando buenos hijos puede ser indicativo de que la cruza no está siendo efectiva, y mientras ésta no se modifique, probablemente no obtendremos resultados positivos.

5.6 Hipervolumen - criterio de paro

Para tener un criterio de paro, hacemos uso de la medida de desempeño denominada hipervolumen (ver Apéndice C en la página 117), que ayuda a medir la convergencia del algoritmo. Esta medida de desempeño no es una medida absoluta de convergencia, pero nos arroja una idea sobre el comportamiento del algoritmo y el camino que lleva recorrido cuando comparamos el hipervolumen obtenido en la iteración actual con el de generaciones pasadas.

En conjunto con el número de generaciones, también se encuentra el tamaño de población, ya que como Abbass indica [1], no es necesario contar con un gran tamaño de población, ya que esto se puede compensar con una búsqueda más larga en la que se use un buen mecanismo para mantener la diversidad. Debido a esto, para definir un criterio de paro para nuestro algoritmo, tomamos en cuenta estos dos aspectos.

El funcionamiento es muy simple, la población inicial será de 100 individuos, y se ejecutarán 100 generaciones en las que no se aplicará el hipervolumen. En la figura 5.6 podemos observar de manera gráfica el procedimiento, donde las esferas indican el tamaño de la población, así como el inicio y el fin del proceso evolutivo.

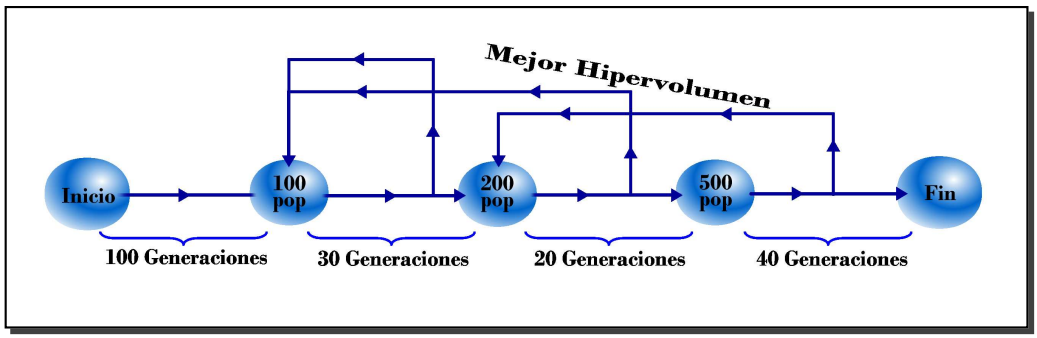


Figura 5.6: Modo de operación del criterio de paro.

A partir de la generación 100, se comenzará a analizar la población con el indicador del hipervolumen. Si después de 30 generaciones no existe ninguna mejora, entonces se agregarán 100 individuos generados de forma totalmente aleatoria (análogamente a como se generó la población inicial). Estos individuos darán mayor diversidad a la población existente, teniendo ahora 200 individuos. Si, por el contrario, antes de 30 generaciones aparece una generación con mejores individuos (que mejoren el indicador del hipervolumen), debemos reiniciar el contador. Es decir, tenemos que acumular 30 generaciones ininterrumpidas sin mejora del hipervolumen.

Debemos tomar en cuenta que para aplicar la métrica del hipervolumen, es necesario un punto de referencia (ver Apéndice C). Para fines de una comparación justa, este punto debe permanecer constante de una generación a otra. Lo que recomendamos es obtener este punto al aplicar por primera vez la métrica y hacer las comparaciones posteriores usando el mismo punto.

Ya con los 200 individuos y las 30 generaciones ininterrumpidas, continua-

mos iterando. Si transcurren 20 generaciones más sin mejorar el hipervolumen (con todo y la diversidad que agregamos), entonces agregamos 300 individuos más, de la misma forma en que agregamos los 100 individuos anteriores llegando a un total de 500 individuos.

Si antes de alcanzar las 20 generaciones (50 generaciones totales ininterrumpidas) existe una mejora en el hipervolumen, entonces le quitamos 100 individuos a la población (en este momento tenemos 200 individuos, y debemos quedar sólo con 100). Para ello se eligen a los 100 mejores individuos según su jerarquía asignada (o *rank* como lo señala el NSGA-II [24]). Si existen empates en la jerarquía, se desempata por medio de la distancia de agrupamiento (o *crowding distance*). El objetivo es quedarse con las mejores soluciones y con una buena distribución de la muestra. Además, se debe reiniciar el conteo de generaciones, es decir, debemos iniciar nuevamente la búsqueda de 30 generaciones ininterrumpidas antes de poder agregar 100 individuos más.

Contando con 500 individuos, seguimos iterando. Si transcurren 40 generaciones más sin mejorar el hipervolumen, podemos decir que no vale la pena seguir iterando, y damos por terminadas las iteraciones.

Si, por el contrario, encontramos un mejor hipervolumen antes de las 40 generaciones (90 generaciones totales ininterrumpidas sin mejorar el hipervolumen), entonces le quitamos 300 individuos a la población y mantenemos a los 200 mejores individuos de la misma forma como lo hicimos cuando quitamos 100 individuos. Además, el contador de iteraciones se debe colocar en 30 generaciones, es decir, ahora se buscará acumular 60 generaciones (20 generaciones con 200 individuos y 40 más con 500 individuos). Si antes de llegar a las 20 generaciones se encuentra un mejor hipervolumen, entonces se eliminan individuos y se cambia el contador de generaciones como en los casos antes mencionados.

El agregar o quitar individuos se realiza siempre de la misma forma, ya sea agregando nuevos individuos de forma totalmente aleatoria, o quitando algunos para dejar sólo a los mejores individuos según su jerarquía y distancia de agrupamiento.

Los valores de 100, 200 y 500 individuos son los más comúnmente utilizados por los diferentes investigadores. Valores más pequeños no serían efectivos en la fase de exploración y nos harían perder soluciones relevantes, ya que no contamos con un método de almacenamiento de soluciones para poblaciones pequeñas (como el μAG^2 [89]). En cuanto a valores mayores, nos arrojaría mucha sobrecarga en el procesamiento. Además, no es recomendable usar

poblaciones muy grandes, sino procesos más largos.

Con este mecanismo pretendemos dar diversidad a las soluciones y analizar la convergencia de las soluciones obtenidas, además de ir analizando los cambios existentes en la medida de desempeño para decidir el momento en que ya no es fructífero seguir iterando.

6

Diseño Experimental y Análisis de Resultados

Las pruebas se realizaron sobre las 5 funciones de la serie ZDT que no tienen restricciones y sobre las 7 funciones de la serie DTLZ (ver apéndice A y apéndice B). Se eligieron estas funciones por las diferentes características que presentan y por ser el tipo de problemas a los que buscamos darles solución.

Para realizar una comparación en el desempeño de forma cuantitativa utilizamos las métricas siguientes (ver apéndice C): distancia generacional inversa [92], indicador ϵ-unario [106], dispersión [24] y la cobertura de 2 conjuntos [102]. Las primeras tres hacen una comparación entre el conjunto de soluciones encontrada por un algoritmo con el verdadero frente de Pareto, y la última realiza una comparación directa entre las soluciones encontradas por dos algoritmos.

La comparación de resultados se hizo directamente con respecto al NSGA-II original [24]. Los parámetros usados para llevar a cabo las diferentes pruebas fueron:

- Tamaño de población = 500
- Tipo de variables (codificación) = *reales*

- Probabilidad de cruza = 0.7
- Probabilidad de mutación de las variables reales = 0.1
- Indice de distribución de la cruza de las variables reales = 1
- Indice de distribución de la mutación de las variables reales = 50
- Los límites de las variables reales = *acotados*
- Tipo de cruza para las variables reales = SBX
- Tipo de mutación para las variables reales = $Parameter - Based - Mutation$

Para cada problema en particular se definieron:

- Las funciones objetivo
- El número de las funciones objetivo
- El número de variables de decisión
- Los límites de cada una de las variables decisión

En cuanto al número de generaciones se realizaron dos tipos de prueba:

- **Número de Evaluaciones de las Funciones Objetivo (Evaluaciones)**: Usando el número promedio de evaluaciones de las funciones objetivo que realizó nuestra propuesta, por problema, se realiza un aproximado del número de generaciones que requiere el NSGA-II, para que utilizando 500 individuos realice el mismo número total de evaluaciones. El valor obtenido para cada problema se encuentra en las tablas 6.1 y 6.2.
- **Número de Generaciones (Generaciones)**: El número promedio de generaciones que realizó nuestra propuesta por problema, es el número de generaciones que el NSGA-II iterará. El valor del número de generaciones por problema se encuentra en las tablas 6.1 y 6.2.

En las tablas 6.1 y 6.2 observamos el promedio de la generación en que el algoritmo con técnicas de auto-adaptación se detiene, así como el promedio de las evaluaciones que éste realiza. Los dos renglones finales nos indican el número de generaciones que usará, por problema, el NSGA-II original. El primer renglón muestra el promedio de las generaciones en que el algoritmo con auto-adaptación se detiene, y el segundo es el número de generaciones requerido para que, usando una población de 500 individuos, el NSGA-II realice un número total de evaluaciones semejante a las que realizó el algoritmo con auto-adaptación.

Auto-Adaptación					
Problema	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
Promedio Generaciones	629.5	642.35	959.55	1675.15	2015.95
Promedio Evaluaciones	118,630	120,110	176,290	305,350	428,850
Generaciones	**630**	**643**	**960**	**1676**	**2016**
Evaluaciones	**238**	**241**	**353**	**611**	**858**

Tabla 6.1: Generaciones y número de evaluaciones en el algoritmo con auto-adaptación para los problemas ZDT

Auto-Adaptación							
Problema	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
Promedio Generaciones	378.9	323.15	496.85	288.25	247.05	249.3	305.7
Promedio Evaluaciones	68,535	57,210	84,570	52,570	44,540	49,245	58,390
Generaciones	**379**	**324**	**497**	**289**	**248**	**250**	**306**
Evaluaciones	**138**	**115**	**170**	**106**	**90**	**99**	**117**

Tabla 6.2: Generaciones y número de evaluaciones en el algoritmo con auto-adaptación para los problemas DTLZ

Se realizaron 20 ejecuciones diferentes con cada algoritmo utilizando diferentes semillas generadoras de números aleatorios.

En las siguientes secciones se presentan los resultados obtenidos. *NSGA-II evaluaciones* es el algoritmo original con el número de generaciones basado en el número promedio de evaluaciones que realiza nuestra propuesta. *NSGA-II*

generaciones es el algoritmo original con el número de generaciones basado en el promedio de las generaciones en que nuestra propuesta se detiene. *Auto-Adaptación* indica nuestra propuesta.

Por cada problema, se presenta en una tabla la comparación de resultados entre el NSGA-II evaluaciones y la propuesta con auto-adaptación, y en otra tabla la comparación de resultados entre el NSGA-II generaciones y la propuesta con auto-adaptación. Ambas tablas contienen los datos de las siguientes métricas: distancia generacional invertida, indicador ϵ-unario y dispersión. En cada tabla se presenta el promedio, la desviación estándar, el valor mínimo y el máximo de cada métrica. El valor sombreado es el que indica el menor valor (el mejor) comparando ambos algoritmos.

Para la métrica de cobertura de 2 conjuntos, se presenta una tabla, donde se muestran los resultados de la métrica entre el NSGA-II evaluaciones y nuestra propuesta con auto-adaptación. En la misma tabla, pero en la segunda parte se muestran los resultados de la métrica entre el NSGA-II generaciones y nuestra propuesta con auto-adaptación. Al igual que en las otras tablas, se muestra el promedio, la desviación estándar, el valor mínimo y el máximo de la métrica. Sin embargo, el número sombreado en este caso es el de mayor valor (aquí un valor mayor indica un mejor resultado).

También se presentan cuatro gráficas por problema. La de arriba a la izquierda muestra a la ejecución promedio de la distancia generacional invertida de nuestra propuesta (auto-adaptación). La de arriba a la derecha muestra la ejecución promedio de la distancia generacional invertida del NSGA-II evaluaciones. La de abajo a la izquierda muestra la ejecución promedio de la distancia generacional invertida del NSGA-II generaciones. La de abajo a la derecha muestra al verdadero frente de Pareto.

6.1 ZDT1

En la figura 6.1 podemos observar de forma gráfica que los tres algoritmos se encuentran sobre el verdadero frente. En cuanto a las métricas, observamos que los tres algoritmos se encuentran prácticamente empatados (tablas 6.3 y 6.4). Sólo con respecto a la dispersión, es que nuestra propuesta logra sobresalir en este caso, obteniendo mejores resultados.

En la tabla 6.5 observamos la comparación de resultados de los algoritmos de forma más directa, usando la cobertura de 2 conjuntos. Aquí comparamos nuestro algoritmo con auto-adaptación con respecto al NSGA-II original en

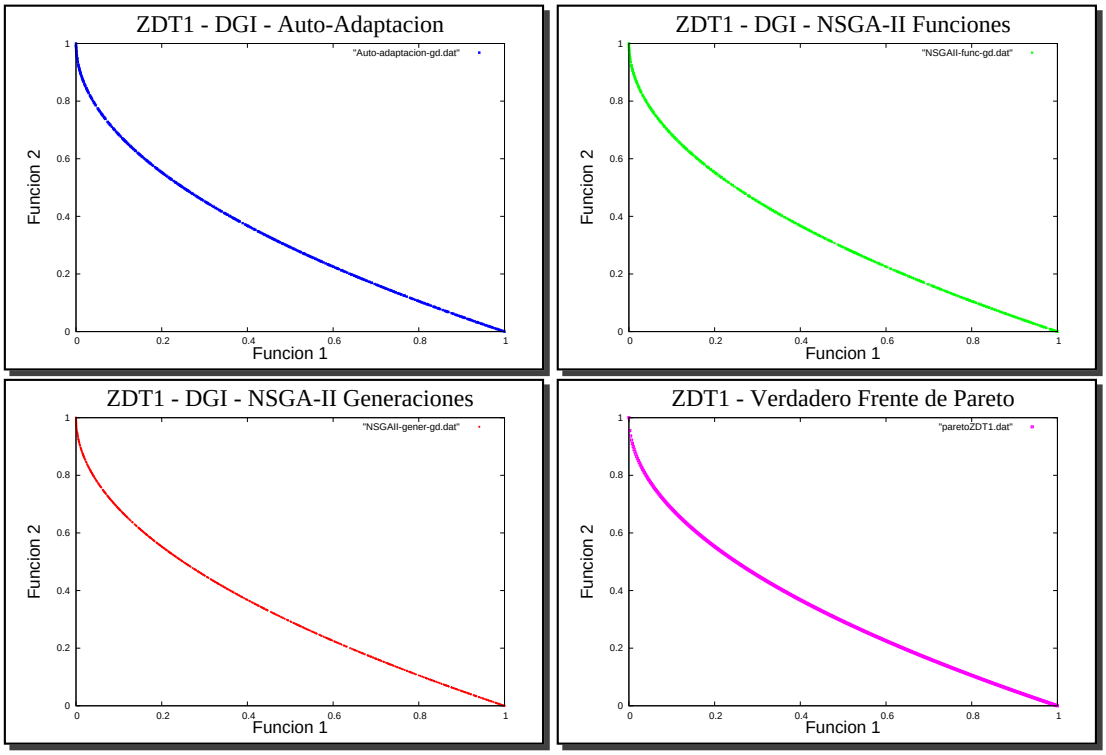


Figura 6.1: Comparación de las ejecuciones promedio por métrica y el verdadero frente de Pareto

Problema	ZDT1				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II evaluaciones	0.000058	0.000002	0.000055	0.000061
	Auto-Adaptación	**0.000056**	**0.000001**	**0.000054**	**0.000058**
Indicador ϵ-unario	NSGA-II evaluaciones	1.002235	**0.000322**	**1.001712**	**1.002855**
	Auto-Adaptación	**1.002214**	0.000373	1.001724	1.003285
Dispersión	NSGA-II evaluaciones	0.65008	0.040356	0.575088	0.715359
	Auto-Adaptación	**0.542132**	**0.029852**	**0.495311**	**0.610552**

Tabla 6.3: Métricas del problema ZDT1

Problema		ZDT1			
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II generaciones	0.000058	0.000002	0.000055	0.000061
	Auto-Adaptación	**0.000056**	**0.000001**	**0.000054**	**0.000058**
Indicador ϵ-unario	NSGA-II generaciones	**1.002152**	**0.000225**	1.001747	**1.002599**
	Auto-Adaptación	1.002214	0.000373	**1.001724**	1.003285
Dispersión	NSGA-II generaciones	0.663761	0.034663	0.599662	0.720436
	Auto-Adaptación	**0.542132**	**0.029852**	**0.495311**	**0.610552**

Tabla 6.4: Métricas del problema ZDT1

ZDT1 Cobertura de 2 conjuntos				
% que Domina A a B	Media	σ	Peor	Mejor
NSGA-II evaluaciones a Auto-adaptación	0.0049	0.001179	**0.004**	0.008
Auto-adaptación a NSGA-II evaluaciones	**0.0068**	**0.002638**	0.002	**0.012**
NSGA-II generaciones a Auto-adaptación	0.0051	0.001338	**0.004**	0.008
Auto-adaptación a NSGA-II generaciones	**0.0063**	**0.002629**	0.002	**0.012**

Tabla 6.5: Resultados de la métrica cobertura de dos conjuntos del problema ZDT1

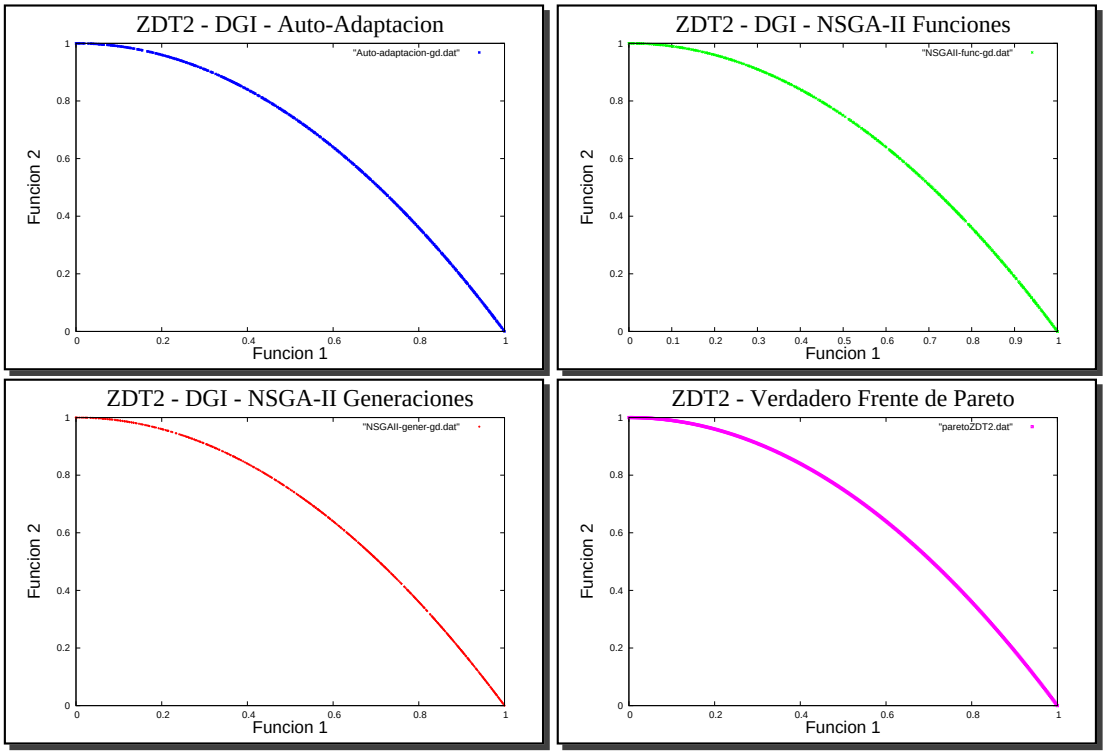


Figura 6.2: Comparación de las ejecuciones promedio por métrica y el verdadero frente de Pareto

sus dos formas (evaluaciones y generaciones). Podemos observar cómo en ambos casos, nuestra propuesta resulta ligeramente superior al NSGA-II original (en sus dos variantes).

6.2 ZDT2

En la figura 6.2 podemos observar de forma gráfica que los tres algoritmos se encuentran sobre el verdadero frente. En cuanto a las métricas, observamos que el algoritmo con técnicas de auto-adaptación supera marginalmente a las dos formas del algoritmo original (ver tablas 6.6 y 6.7). Al igual que en el problema ZDT1, en la métrica de dispersión, nuestra propuesta obtiene resultados ligeramente mejores.

En la tabla 6.8 observamos los resultados de la métrica cobertura de 2

Problema	ZDT2				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II evaluaciones	0.000059	**0.000002**	0.000055	**0.000063**
	Auto-Adaptación	**0.000058**	0.000003	**0.000054**	0.000065
Indicador ϵ-unario	NSGA-II evaluaciones	1.001938	0.000440	1.001341	1.002926
	Auto-Adaptación	**1.001792**	**0.000346**	**1.001251**	**1.002637**
Dispersión	NSGA-II evaluaciones	0.666593	0.037943	0.562049	0.732862
	Auto-Adaptación	**0.539112**	**0.025966**	**0.491643**	**0.585362**

Tabla 6.6: Métricas del problema ZDT2

Problema	ZDT2				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II generaciones	0.000059	**0.000002**	0.000056	**0.000063**
	Auto-Adaptación	**0.000058**	0.000003	**0.000054**	0.000065
Indicador ϵ-unario	NSGA-II generaciones	1.001967	**0.000317**	1.001527	1.002782
	Auto-Adaptación	**1.001792**	0.000346	**1.001251**	**1.002637**
Dispersión	NSGA-II generaciones	0.662773	0.035102	0.59052	0.719997
	Auto-Adaptación	**0.539112**	**0.025966**	**0.491643**	**0.585362**

Tabla 6.7: Métricas del problema ZDT2

ZDT2 Cobertura de 2 conjuntos				
% que Domina A a B	Media	σ	Peor	Mejor
NSGA-II evaluaciones a Auto-adaptación	0.0047	0.001453	0.002	0.008
Auto-adaptación a NSGA-II evaluaciones	**0.0090**	**0.002324**	**0.006**	**0.014**
NSGA-II generaciones a Auto-adaptación	0.0046	0.001114	0.002	0.006
Auto-adaptación a NSGA-II generaciones	**0.0072**	**0.001720**	**0.004**	**0.010**

Tabla 6.8: Resultados de la métrica cobertura de dos conjuntos del problema ZDT2

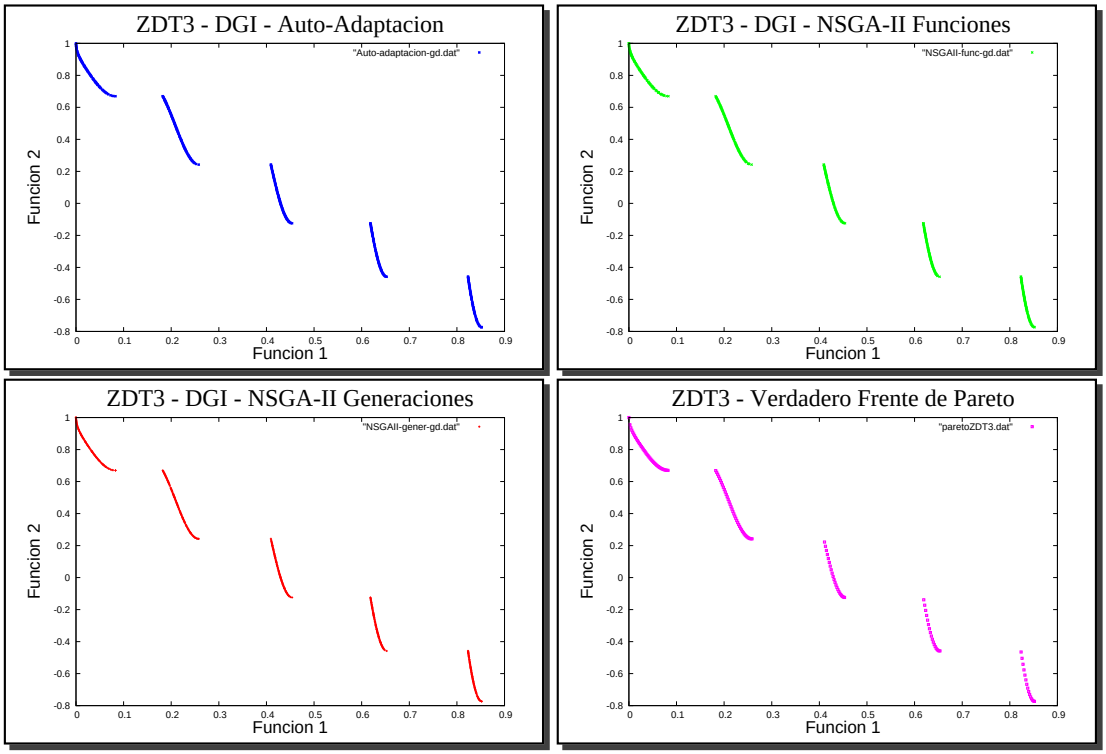


Figura 6.3: Comparación de las ejecuciones promedio por métrica y el verdadero frente de Pareto

conjuntos y, al igual que en la función ZDT1, podemos observar que nuestra propuesta obtuvo resultados ligeramente mejores. De hecho, en este caso, la diferencia de resultados es más grande que en el ejemplo anterior, lo que indica que nuestra propuesta tuvo un mejor desempeño en este problema.

6.3 ZDT3

En la figura 6.3 observamos que los tres algoritmos tienen un buen desempeño, alcanzando el verdadero frente de Pareto. En cuanto a las métricas, las dos versiones del algoritmo original mejoran marginalmente a nuestra propuesta en el indicador ϵ-unario. Sin embargo, en las dos métricas restantes, nuestra propuesta se mantiene ligeramente superior (ver tablas 6.9 y 6.10).

En la tabla 6.11 se muestran los resultados de la métrica de cobertura de

Problema	ZDT3				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II evaluaciones	0.000132	0.000007	**0.000116**	**0.000143**
	Auto-Adaptación	**0.000127**	0.000007	0.000117	0.000147
Indicador ϵ-unario	NSGA-II evaluaciones	**1.001863**	**0.000393**	**1.001127**	**1.002632**
	Auto-Adaptación	1.002024	0.000609	1.001314	1.003831
Dispersión	NSGA-II evaluaciones	0.746819	0.036011	0.666930	0.837671
	Auto-Adaptación	**0.612249**	**0.021919**	**0.569107**	**0.645176**

Tabla 6.9: Métricas del problema ZDT3

Problema	ZDT3				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II generaciones	0.000132	**0.000006**	0.000122	**0.000146**
	Auto-Adaptación	**0.000127**	0.000007	**0.000117**	0.000147
Indicador ϵ-unario	NSGA-II generaciones	**1.001830**	**0.000226**	1.001439	**1.002171**
	Auto-Adaptación	1.002024	0.000609	**1.001314**	1.003831
Dispersión	NSGA-II generaciones	0.732614	0.030007	0.648737	0.775920
	Auto-Adaptación	**0.612249**	**0.021919**	**0.569107**	**0.645176**

Tabla 6.10: Métricas del problema ZDT3

ZDT3 Cobertura de 2 conjuntos				
% que Domina A a B	Media	σ	Peor	Mejor
NSGA-II evaluaciones a Auto-adaptación	0.0093	0.002629	0.002	0.016
Auto-adaptación a NSGA-II evaluaciones	**0.0114**	**0.004341**	**0.006**	**0.020**
NSGA-II generaciones a Auto-adaptación	0.0117	0.003363	0.006	0.018
Auto-adaptación a NSGA-II generaciones	**0.0120**	**0.004147**	0.006	**0.022**

Tabla 6.11: Resultados de la métrica cobertura de dos conjuntos del problema ZDT3

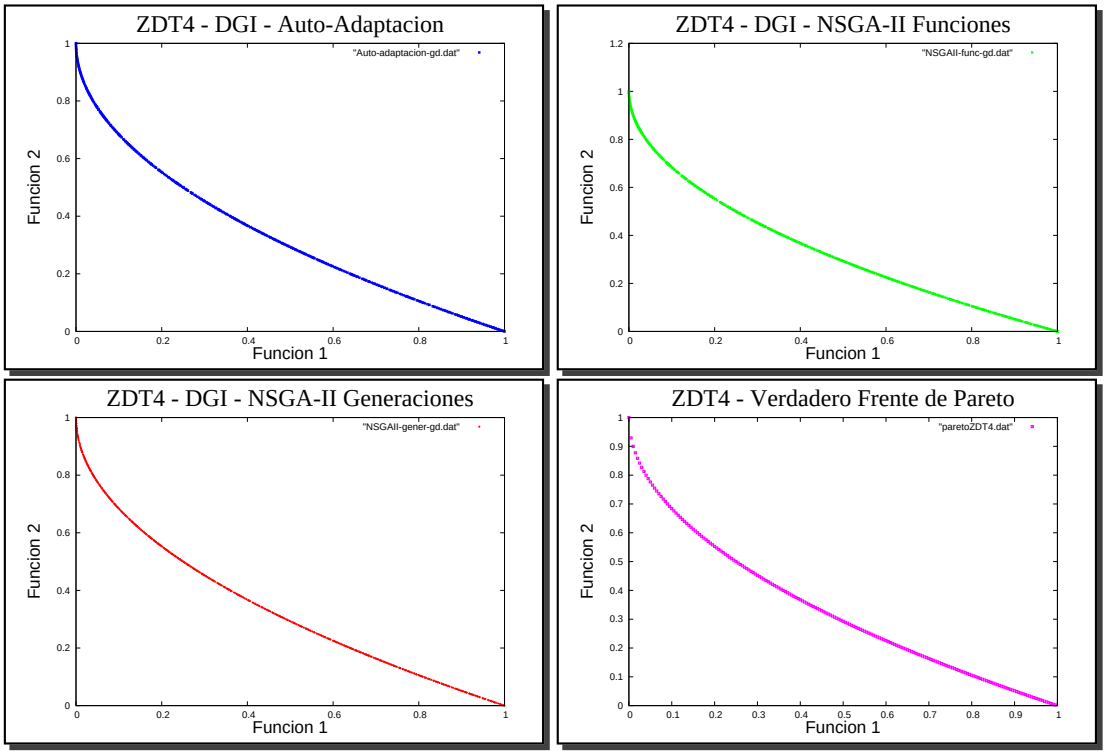


Figura 6.4: Comparación de las ejecuciones promedio por métrica y el verdadero frente de Pareto

dos conjuntos. Podemos observar que, en ambos casos, se muestra ligeramente superior nuestra propuesta con auto-adaptación a las 2 versiones del NSGA-II original. Aunque la diferencia no es significativa, el algoritmo con auto-adaptación se muestra superior.

6.4 ZDT4

De nueva cuenta, los tres algoritmos tienen un buen desempeño, alcanzando el verdadero frente de Pareto como se muestra en la figura 6.4. Sin embargo, al analizar las medidas de desempeño, podemos observar cómo las dos versiones del algoritmo original superan marginalmente al nuestro en la distancia generacional invertida y en el indicador ϵ-unario. En cuanto a la dispersión, nuestra propuesta de auto-adaptación se encuentra ligeramente

por debajo del desempeño del algoritmo original (ver tablas 6.12 y 6.13).

Problema	ZDT4				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II evaluaciones	**0.000083**	0.000004	**0.000077**	**0.000093**
	Auto-Adaptación	0.000093	0.000004	0.000085	0.000099
Indicador ϵ-unario	NSGA-II evaluaciones	**1.001709**	**0.000316**	**1.001311**	**1.002525**
	Auto-Adaptación	1.001916	0.000420	1.001464	1.002996
Dispersión	NSGA-II evaluaciones	**0.556536**	**0.019120**	**0.513790**	**0.589066**
	Auto-Adaptación	0.801464	0.044060	0.710141	0.882463

Tabla 6.12: Métricas del problema ZDT4

Problema	ZDT4				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II generaciones	**0.000084**	0.000004	**0.000079**	**0.000091**
	Auto-Adaptación	0.000093	0.000004	0.000085	0.000099
Indicador ϵ-unario	NSGA-II generaciones	**1.001674**	**0.000324**	**1.001260**	**1.002480**
	Auto-Adaptación	1.001916	0.000420	1.001464	1.002996
Dispersión	NSGA-II generaciones	**0.578010**	**0.041313**	**0.480830**	**0.657318**
	Auto-Adaptación	0.801464	0.044060	0.710141	0.882463

Tabla 6.13: Métricas del problema ZDT4

En la tabla 6.14 se muestran los resultados de la métrica de cobertura de 2 conjuntos. En este caso, la comparación entre el NSGA-II evaluaciones y la propuesta de auto-adaptación muestra una clara superioridad de nuestra propuesta. Sin embargo, al usar más generaciones (y, por ende, tener un número de evaluaciones) al NSGA-II, la diferencia entre el NSGA-II y nuestra propuesta se acorta, aunque la propuesta con auto-adaptación se mantiene superior.

6.5 ZDT6

Para esta función, el desempeño de nuestra propuesta se advierte claramente superior, como se observa en la figura 6.5. Ahí puede observarse cómo el algoritmo original en sus dos formas converge a un falso frente de Pareto.

ZDT4 Cobertura de 2 conjuntos				
% que Domina A a B	Media	σ	Peor	Mejor
NSGA-II evaluaciones a Auto-adaptación	0.0046	0.003040	0	0.012
Auto-adaptación a NSGA-II evaluaciones	**0.0297**	**0.008373**	**0.016**	**0.048**
NSGA-II generaciones a Auto-adaptación	0.0193	**0.018839**	0.004	**0.064**
Auto-adaptación a NSGA-II generaciones	**0.0248**	0.007884	**0.016**	0.048

Tabla 6.14: Resultados de la métrica cobertura de dos conjuntos del problema ZDT4

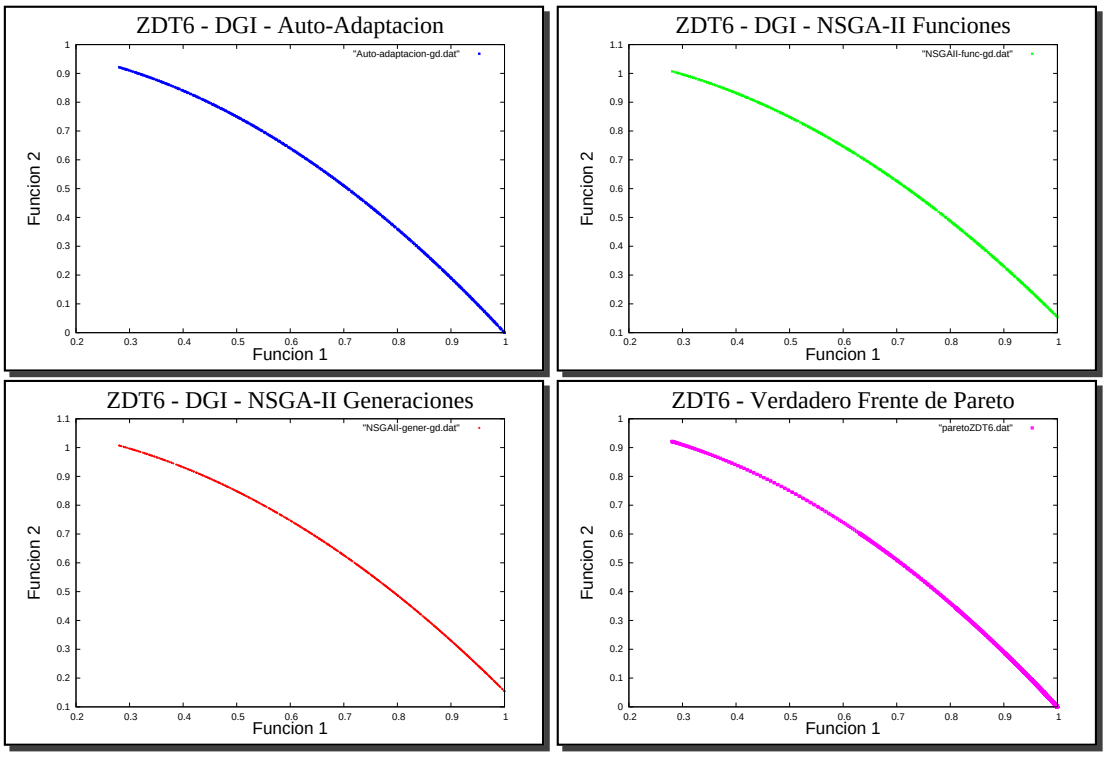


Figura 6.5: Comparación de las ejecuciones promedio por métrica y el verdadero frente de Pareto

Aunque el NSGA-II original muestra mejor dispersión en las dos variantes utilizadas, nuestro algoritmo produce un conjunto de soluciones, que cuentan con mejor convergencia (ver tablas 6.15 y 6.16). Evidentemente, la mejor convergencia resulta más importante que la dispersión cuando algún algoritmo no logra llegar al verdadero frente de Pareto.

Problema	ZDT6				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II evaluaciones	0.002230	0.000191	0.001758	0.002485
	Auto-Adaptación	**0.000018**	**0.000002**	**0.000015**	**0.000025**
Indicador ϵ-unario	NSGA-II evaluaciones	1.140147	0.011069	1.112545	1.154877
	Auto-Adaptación	**1.001409**	**0.000171**	**1.001081**	**1.001759**
Dispersión	NSGA-II evaluaciones	**0.788426**	**0.024690**	**0.730368**	**0.837412**
	Auto-Adaptación	0.856744	0.046337	0.781996	0.928869

Tabla 6.15: Métricas del problema ZDT6

Problema	ZDT6				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II generaciones	0.002230	0.000191	0.001758	0.002485
	Auto-Adaptación	**0.000018**	**0.000002**	**0.000015**	**0.000025**
Indicador ϵ-unario	NSGA-II generaciones	1.140147	0.011069	1.112545	1.154877
	Auto-Adaptación	**1.001409**	**0.000171**	**1.001081**	**1.001759**
Dispersión	NSGA-II generaciones	**0.796272**	**0.024848**	**0.734806**	**0.831883**
	Auto-Adaptación	0.856744	0.046337	0.781996	0.928869

Tabla 6.16: Métricas del problema ZDT6

En la tabla 6.17 observamos los resultados de la métrica de cobertura de 2 conjuntos, y, como se mencionó antes, los resultados de nuestra propuesta son mejores que los producidos por el NSGA-II original en sus dos formas (generaciones y evaluaciones). Esto debido a que el NSGA-II converge a un falso frente de Pareto, lo cual no le ocurre a nuestro algoritmo propuesto.

ZDT6 Cobertura de 2 conjuntos				
% que Domina A a B	Media	σ	Peor	Mejor
NSGA-II evaluaciones a Auto-adaptación	0	0	0	0
Auto-adaptación a NSGA-II evaluaciones	**1**	0	**1**	**1**
NSGA-II generaciones a Auto-adaptación	0	0	0	0
Auto-adaptación a NSGA-II generaciones	**1**	0	**1**	**1**

Tabla 6.17: Resultados de la métrica cobertura de dos conjuntos del problema ZDT6

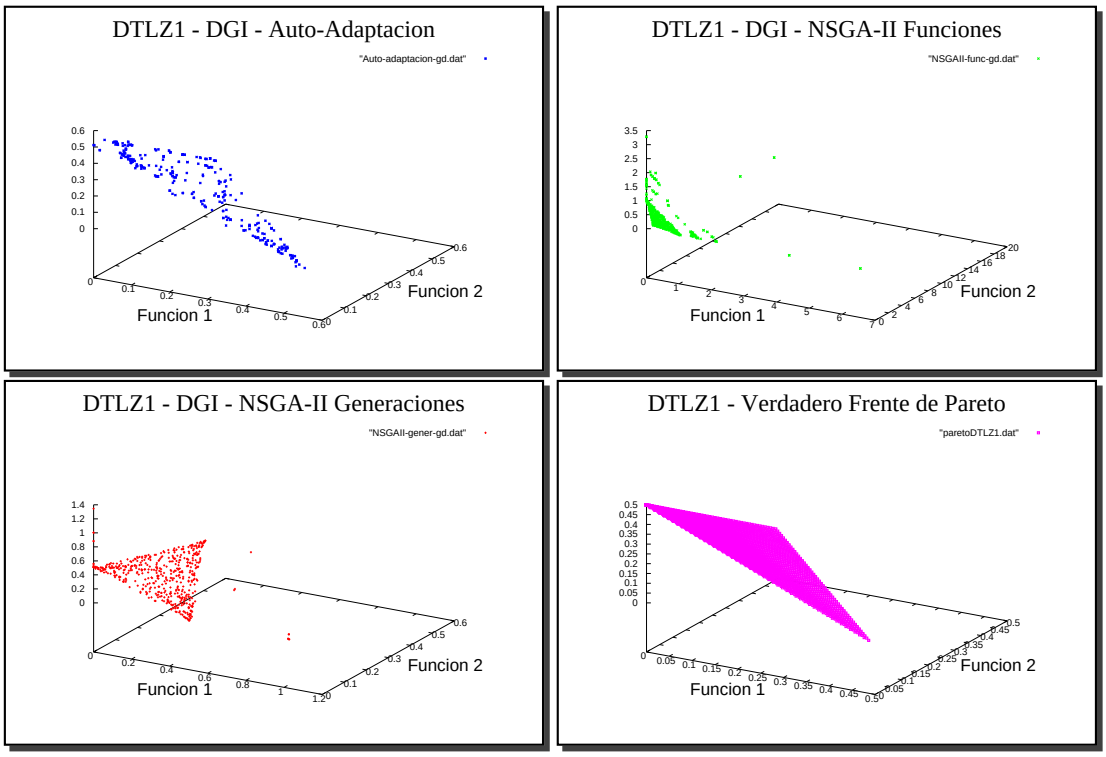


Figura 6.6: Comparación de las ejecuciones promedio por métrica y el verdadero frente de Pareto

6.6 DTLZ1

Para esta función, el desempeño de nuestra propuesta se nota claramente superior, como se observa en la figura 6.6, en la que resulta evidente que el algoritmo original, en sus dos versiones, converge a un falso frente de Pareto. Nuevamente, nuestro algoritmo si logra converger al verdadero frente de Pareto, aunque con una distribución más pobre de soluciones (ver tablas 6.18 y 6.19).

Problema	DTLZ1				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II evaluaciones	0.005666	0.006462	0.000278	0.028005
	Auto-Adaptación	**0.002161**	**0.003941**	**0.000243**	**0.012603**
Indicador ϵ-unario	NSGA-II evaluaciones	1.190444	0.181980	1.019547	1.788120
	Auto-Adaptación	**1.079437**	**0.123285**	**1.017503**	**1.420721**
Dispersión	NSGA-II evaluaciones	0.974938	0.190992	0.627571	1.417455
	Auto-Adaptación	**0.546923**	**0.160001**	**0.438101**	**1.158641**

Tabla 6.18: Métricas del problema DTLZ1

Problema	DTLZ1				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II generaciones	**0.001938**	**0.003133**	**0.000242**	**0.011599**
	Auto-Adaptación	0.002161	0.003941	0.000243	0.012603
Indicador ϵ-unario	NSGA-II generaciones	**1.066070**	**0.082606**	1.018933	**1.320658**
	Auto-Adaptación	1.079437	0.123285	**1.017503**	1.420721
Dispersión	NSGA-II generaciones	0.576810	**0.087172**	0.446846	**0.787951**
	Auto-Adaptación	**0.546923**	0.160001	**0.438101**	1.158641

Tabla 6.19: Métricas del problema DTLZ1

En la tabla 6.20 se encuentran los resultados de la métrica de cobertura de dos conjuntos. Para el caso en que comparamos al NSGA-II evaluaciones con nuestra propuesta, observamos claramente un comportamiento superior de nuestra propuesta. Sin embargo, al darle al NSGA-II generaciones casi el triple de generaciones de las utilizadas en el NSGA-II evaluaciones, los resultados cambian. Aunque la diferencia no es abrumadora, el NSGA-II

DTLZ1 Cobertura de 2 conjuntos				
% que Domina A a B	Media	σ	Peor	Mejor
NSGA-II evaluaciones a Auto-adaptación	0.139875	0.308772	0	1
Auto-adaptación a NSGA-II evaluaciones	**0.750600**	**0.354045**	**0.022**	1
NSGA-II generaciones a Auto-adaptación	**0.295764**	0.434846	0	1
Auto-adaptación a NSGA-II generaciones	0.227900	**0.370091**	0	1

Tabla 6.20: Resultados de la métrica cobertura de dos conjuntos del problema DTLZ1

generaciones es superior a nuestra propuesta, si bien, esto es a costa de un mayor costo computacional.

6.7 DTLZ2

La figura 6.7 muestra gráficamente que los tres algoritmos logran alcanzar el verdadero frente de Pareto.

En cuanto a las medidas de desempeño (tablas 6.21 y 6.22), la diferencia es mínima, salvo en la dispersión donde el NSGA-II original logra tener un comportamiento mejor pero sólo marginalmente.

Problema	DTLZ2				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II evaluaciones	**0.000172**	**0.000004**	0.000166	0.000182
	Auto-Adaptación	0.000174	0.000005	**0.000164**	**0.000181**
Indicador ϵ-unario	NSGA-II evaluaciones	1.049461	0.006330	1.039488	1.065528
	Auto-Adaptación	**1.046300**	**0.005485**	**1.035423**	**1.054368**
Dispersión	NSGA-II evaluaciones	**0.429743**	**0.022495**	**0.398782**	**0.474392**
	Auto-Adaptación	0.490002	0.034583	0.433252	0.584898

Tabla 6.21: Métricas del problema DTLZ2

En la tabla 6.23 están los resultados de la métrica de cobertura de dos conjuntos. Para el caso en que comparamos al NSGA-II evaluaciones con nuestra propuesta, observamos una ligera superioridad del NSGA-II evaluaciones con respecto nuestra propuesta, si bien ésta es marginal. Como en la

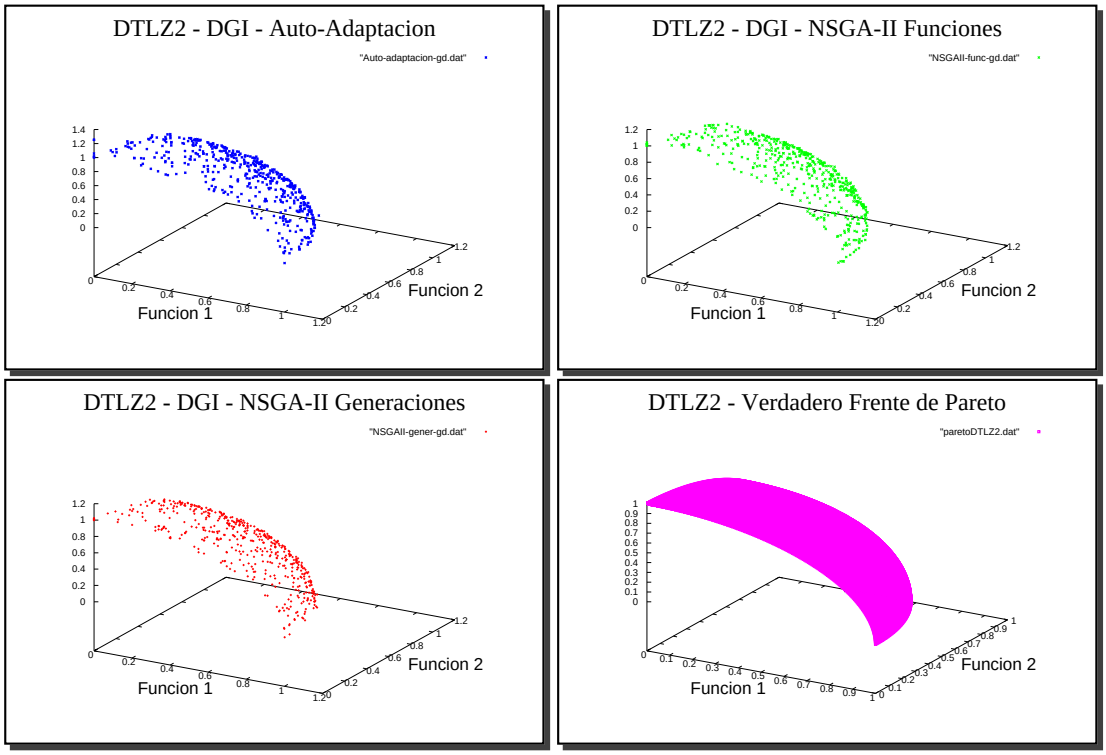


Figura 6.7: Comparación de las ejecuciones promedio por métrica y el verdadero frente de Pareto

Problema	DTLZ2				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II generaciones	**0.000168**	**0.000004**	**0.000161**	**0.000173**
	Auto-Adaptación	0.000174	0.000005	0.000164	0.000181
Indicador ϵ-unario	NSGA-II generaciones	1.047081	0.006184	1.039555	1.064226
	Auto-Adaptación	**1.046300**	**0.005485**	**1.035423**	**1.054368**
Dispersión	NSGA-II generaciones	**0.429741**	**0.017543**	**0.401063**	**0.461253**
	Auto-Adaptación	0.490002	0.034583	0.433252	0.584898

Tabla 6.22: Métricas del problema DTLZ2

DTLZ2 Cobertura de 2 conjuntos				
% que Domina A a B	Media	σ	Peor	Mejor
NSGA-II evaluaciones a Auto-adaptación	**0.0908**	**0.075507**	**0.014**	**0.258**
Auto-adaptación a NSGA-II evaluaciones	0.0629	0.038979	0.008	0.118
NSGA-II generaciones a Auto-adaptación	**0.1088**	**0.077285**	**0.016**	**0.252**
Auto-adaptación a NSGA-II generaciones	0.0434	0.023825	0.010	0.09

Tabla 6.23: Resultados de la métrica cobertura de dos conjuntos del problema DTLZ2

función DTLZ1, al darle al NSGA-II generaciones una mayor cantidad de generaciones de las utilizadas en el NSGA-II evaluaciones, la diferencia aumenta, aunque ésta no es abrumadora. El NSGA-II generaciones es superior a nuestra propuesta, a costa de un mayor número de evaluaciones.

6.8 DTLZ3

En la figura 6.8, observamos que nuestra propuesta converge a un falso frente de Pareto. Sin embargo, el NSGA-II evaluaciones se encuentra aún más lejano del verdadero frente de Pareto, y el NSGA-II generaciones logra acercarse, pero sin hacerlo tanto como nuestra propuesta con.

En cuanto a las métricas (ver tablas 6.24 y 6.25), nuestra propuesta es claramente superior, tanto en términos de convergencia como de dispersión aunque este último aspecto, la diferencia es marginal.

Problema	DTLZ3				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II evaluaciones	0.250904	0.098975	0.092845	0.465332
	Auto-Adaptación	**0.012097**	**0.013000**	**0.000513**	**0.048591**
Indicador ϵ-unario	NSGA-II evaluaciones	10.176407	3.861924	4.233005	17.550244
	Auto-Adaptación	**1.396945**	**0.380866**	**1.031143**	**2.38701**
Dispersión	NSGA-II evaluaciones	1.188861	**0.085993**	0.852953	**1.29091**
	Auto-Adaptación	**0.931206**	0.336853	**0.423347**	1.41652

Tabla 6.24: Métricas del problema DTLZ3

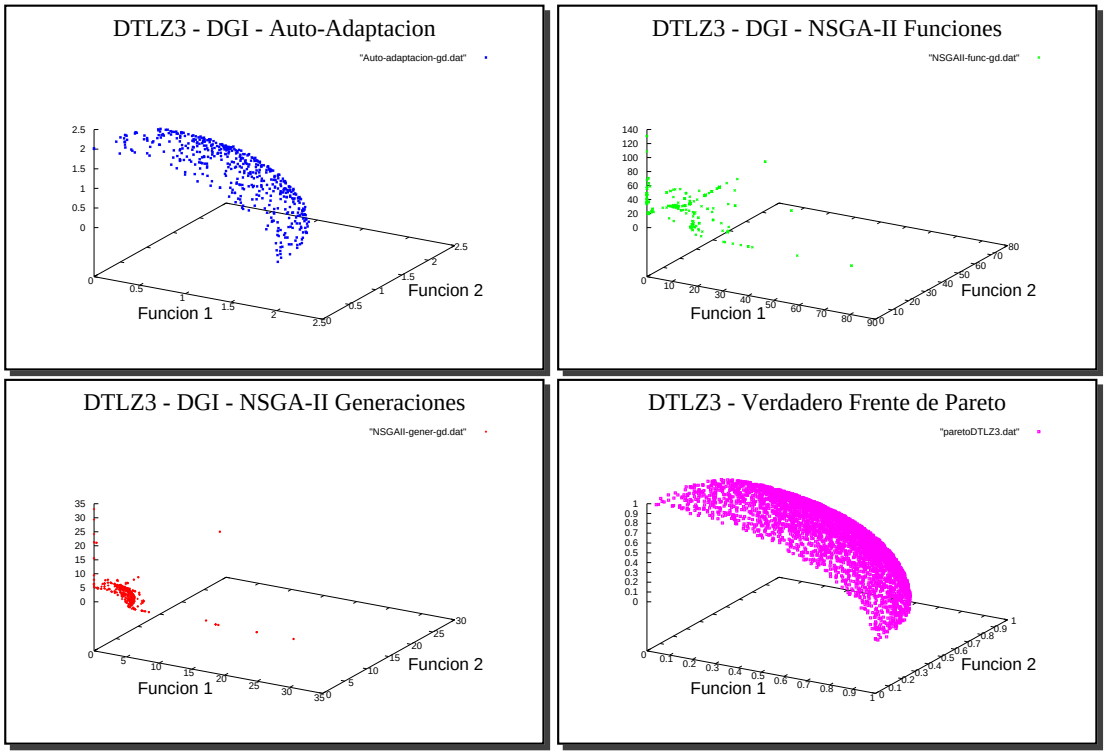


Figura 6.8: Comparación de las ejecuciones promedio por métrica y el verdadero frente de Pareto

Problema	DTLZ3				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II generaciones	0.055570	0.032845	0.003258	0.120746
	Auto-Adaptación	**0.012097**	**0.013000**	**0.000513**	**0.048591**
Indicador ϵ-unario	NSGA-II generaciones	2.667639	0.900274	1.181378	4.478372
	Auto-Adaptación	**1.396945**	**0.380866**	**1.031143**	**2.387010**
Dispersión	NSGA-II generaciones	0.955311	0.117838	0.748755	**1.146335**
	Auto-Adaptación	**0.931206**	**0.336853**	**0.423347**	1.416520

Tabla 6.25: Métricas del problema DTLZ3

DTLZ3 Cobertura de 2 conjuntos				
% que Domina A a B	Media	σ	Peor	Mejor
NSGA-II evaluaciones a Auto-adaptación	0.000847	**0.00123**	0	0.004
Auto-adaptación a NSGA-II evaluaciones	**1**	0	**1**	**1**
NSGA-II generaciones a Auto-adaptación	0.083727	0.233275	0	0.850163
Auto-adaptación a NSGA-II generaciones	**0.885700**	**0.264055**	**0.142**	**1**

Tabla 6.26: Resultados de la métrica cobertura de dos conjuntos del problema DTLZ3

En la tabla 6.26 mostramos los resultados de la métrica de cobertura de 2 conjuntos, apreciándose claramente que nuestra propuesta es superior al NSGA-II original, en sus dos versiones.

6.9 DTLZ4

En la figura 6.9, se muestran los resultados de los tres algoritmos los cuales parecen estar en el verdadero frente de Pareto, excepto por algunos puntos que se desvían ligeramente de él.

En cuanto a las métricas (ver tablas 6.27 y 6.28), sus valores son muy similares, teniéndose diferencias mínimas para las tres métricas.

Problema	DTLZ4				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II evaluaciones	**0.000547**	**0.000009**	0.000523	**0.000558**
	Auto-Adaptación	0.000558	0.000030	**0.000514**	0.000641
Indicador ϵ-unario	NSGA-II evaluaciones	1.042668	0.004336	1.035325	**1.049226**
	Auto-Adaptación	**1.039888**	**0.004084**	**1.033388**	1.049298
Dispersión	NSGA-II evaluaciones	**0.412915**	**0.019106**	**0.385593**	**0.463825**
	Auto-Adaptación	0.453508	0.036211	0.397520	0.528320

Tabla 6.27: Métricas del problema DTLZ4

En la tabla 6.29 están los resultados de la métrica de cobertura de 2 conjuntos, donde nuestra propuesta es superior para el NSGA-II evaluaciones. Sin embargo, al aumentar el número de generaciones, observamos que la

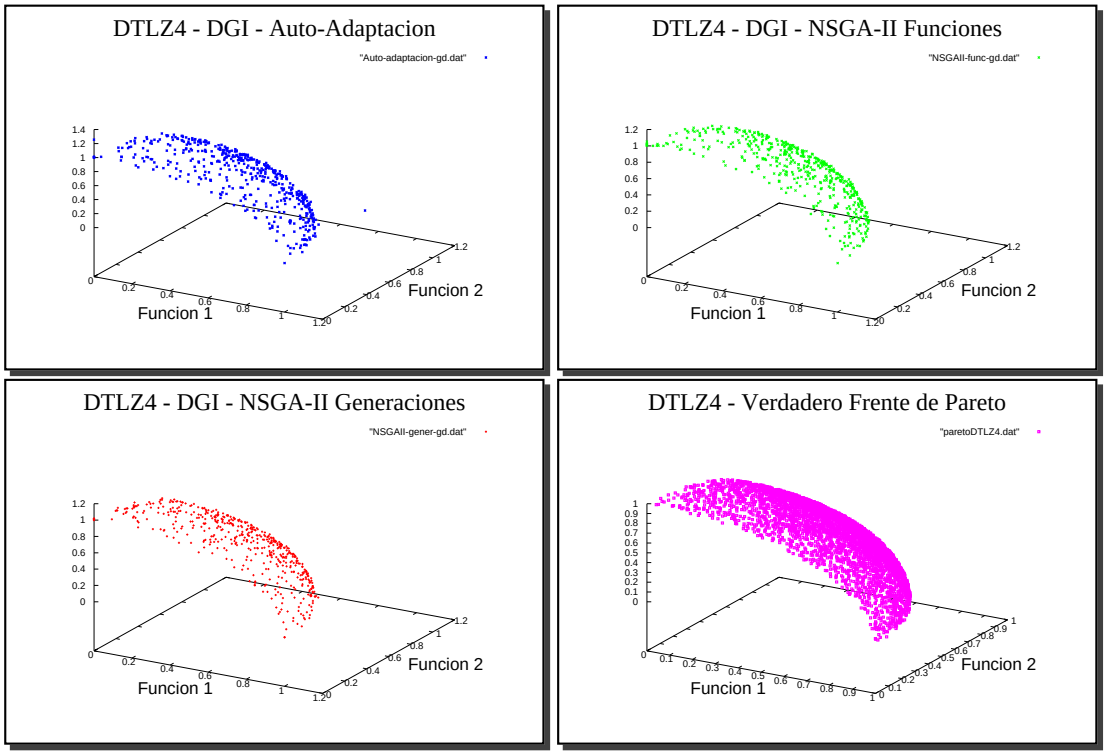


Figura 6.9: Comparación de las ejecuciones promedio por métrica y el verdadero frente de Pareto

Problema	DTLZ4				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II generaciones	**0.000542**	0.000008	0.000528	**0.000557**
	Auto-Adaptación	0.000558	**0.000030**	**0.000514**	0.000641
Indicador ϵ-unario	NSGA-II generaciones	1.040609	0.004801	**1.033100**	1.050510
	Auto-Adaptación	**1.039888**	**0.004084**	1.033388	**1.049298**
Dispersión	NSGA-II generaciones	**0.424239**	**0.018287**	**0.389756**	**0.457516**
	Auto-Adaptación	0.453508	0.036211	0.397520	0.528320

Tabla 6.28: Métricas del problema DTLZ4

DTLZ4 Cobertura de 2 conjuntos				
% que Domina A a B	Media	σ	Peor	Mejor
NSGA-II evaluaciones a Auto-adaptación	0.0510	**0.038613**	0.006	**0.142**
Auto-adaptación a NSGA-II evaluaciones	**0.0702**	0.027261	**0.014**	0.108
NSGA-II generaciones a Auto-adaptación	**0.0842**	**0.048682**	**0.028**	**0.202**
Auto-adaptación a NSGA-II generaciones	0.0546	0.019820	0.022	0.096

Tabla 6.29: Resultados de la métrica cobertura de dos conjuntos del problema DTLZ4

situación cambia, y el NSGA-II generaciones se muestra superior a nuestra propuesta.

6.10 DTLZ5

En la figura 6.10, se muestran los resultados de los tres algoritmos, los cuales parecen estar en el verdadero frente de Pareto. Sin embargo, el NSGA-II generaciones y nuestra propuesta acumulan varias soluciones del en la parte izquierda del frente de Pareto.

En cuanto a las métricas (ver tablas 6.30 y 6.31), el NSGA-II evaluaciones muestra una mejor dispersión. Sin embargo, en las demás métricas, la diferencia es marginal, al igual que en las tres métricas arrojadas por el NSGA-II generaciones, donde la diferencia también es mínima.

Problema	DTLZ5				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II evaluaciones	**0.000015**	**0.000002**	**0.000012**	**0.000022**
	Auto-Adaptación	0.000037	0.000013	0.000025	0.000089
Indicador ϵ-unario	NSGA-II evaluaciones	**1.001418**	**0.000338**	**1.000971**	**1.002128**
	Auto-Adaptación	1.002976	0.001318	1.001944	1.007790
Dispersión	NSGA-II evaluaciones	**0.523047**	0.092002	**0.407403**	**0.729096**
	Auto-Adaptación	0.746702	**0.018931**	0.714991	0.785060

Tabla 6.30: Métricas del problema DTLZ5

En la tabla 6.32 se muestran los resultados de la métrica de cobertura de 2 conjuntos, donde se ve que el NSGA-II evaluaciones es superior a nuestra

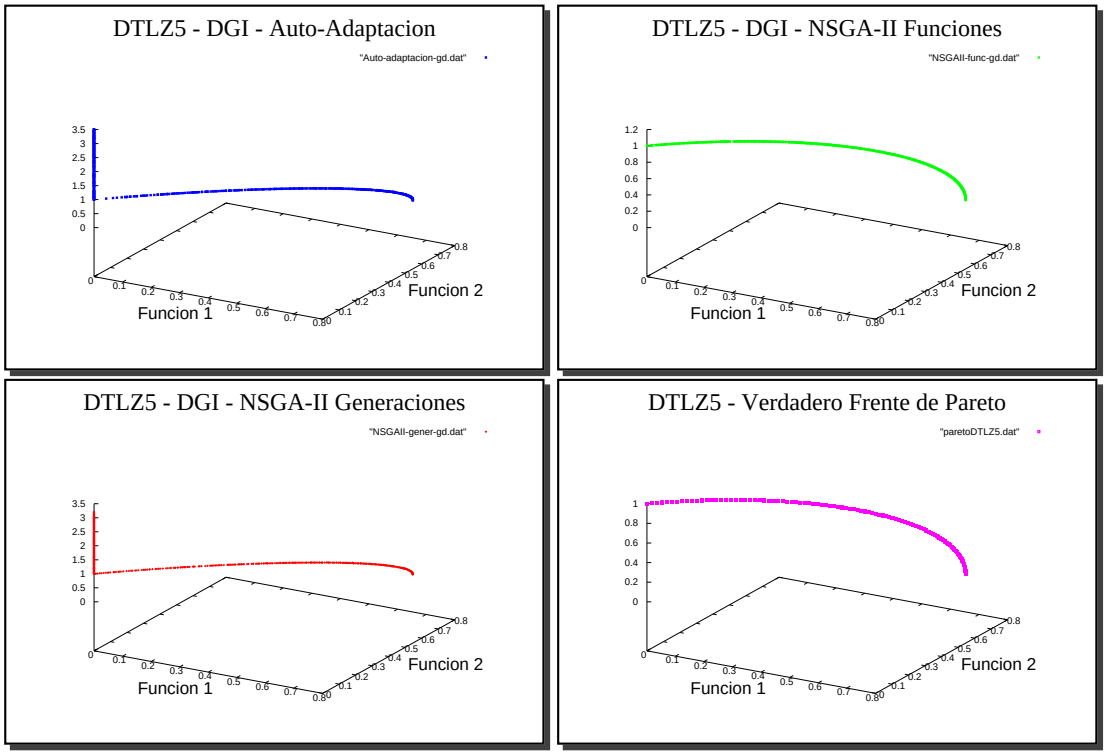


Figura 6.10: Comparación de las ejecuciones promedio por métrica y el verdadero frente de Pareto

Problema	DTLZ5				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II generaciones	**0.000023**	**0.000002**	**0.000018**	**0.000028**
	Auto-Adaptación	0.000037	0.000013	0.000025	0.000089
Indicador ϵ-unario	NSGA-II generaciones	**1.001947**	**0.000312**	**1.001289**	**1.002576**
	Auto-Adaptación	1.002976	0.001318	1.001944	1.007790
Dispersión	NSGA-II generaciones	**0.717033**	**0.018537**	**0.670793**	**0.751835**
	Auto-Adaptación	0.746702	0.018931	0.714991	0.785060

Tabla 6.31: Métricas del problema DTLZ5

DTLZ5 Cobertura de 2 conjuntos				
% que Domina A a B	Media	σ	Peor	Mejor
NSGA-II evaluaciones a Auto-adaptación	**0.6411**	**0.165884**	**0.184**	**0.914**
Auto-adaptación a NSGA-II evaluaciones	0.0738	0.096098	0	0.414
NSGA-II generaciones a Auto-adaptación	**0.6303**	0.046058	**0.566**	**0.718**
Auto-adaptación a NSGA-II generaciones	0.4508	**0.059951**	0.274	0.524

Tabla 6.32: Resultados de la métrica cobertura de dos conjuntos del problema DTLZ5

propuesta, ya que no cuenta con demasiadas soluciones concentradas en el lado izquierdo, sino que logra una buena dispersión de éstas. Con el NSGA-II generaciones se obtiene un resultado similar a nuestra propuesta y por ello la diferencia se vuelve menor.

6.11 DTLZ6

En la figura 6.11, podemos observar que nuestra propuesta alcanza el verdadero frente de Pareto, pero, al igual que en el problema ZDT5, varias soluciones se encuentran en el lado izquierdo del frente. En cuanto al NSGA-II evaluaciones, éste no logra obtener el verdadero frente. El NSGA-II generaciones logra acercarse al verdadero frente de Pareto, pero muchas soluciones quedan fuera de éste.

En cuanto a las métricas (ver tablas 6.33 y 6.34), nuestra propuesta muestra ser superior en todos los casos, excepto por la dispersión, en cuyo caso, el NSGA-II generaciones, es ligeramente superior a nuestra propuesta con auto-adaptación.

En la tabla 6.35 se muestran los resultados de la métrica de cobertura de 2 conjuntos, apreciándose que nuestra propuesta tiene un comportamiento superior al del NSGA-II en sus dos formas. Sin embargo, la diferencia no es abrumadora, aunque sí significativa.

6.12 DTLZ7

En la figura 6.12, observamos que la propuesta con auto-adaptación y el NSGA-II evaluaciones parecen encontrarse muy cerca del verdadero frente

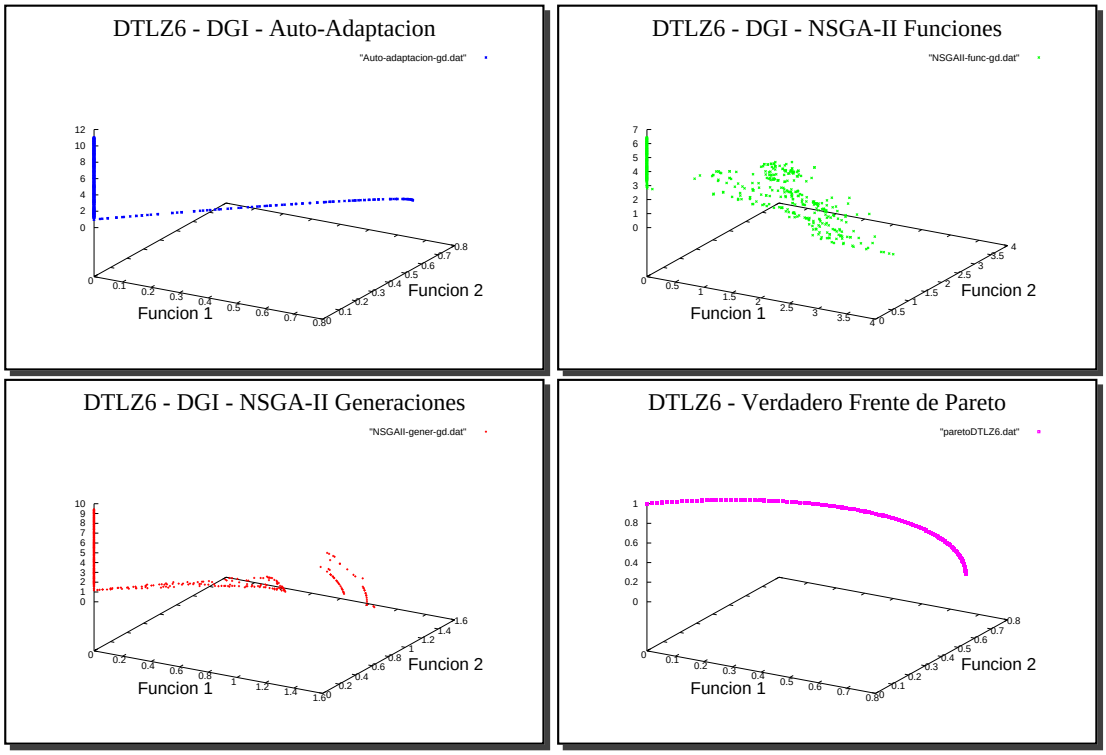


Figura 6.11: Comparación de las ejecuciones promedio por métrica y el verdadero frente de Pareto

Problema	DTLZ6				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II evaluaciones	0.015200	0.002084	0.011071	0.020656
	Auto-Adaptación	**0.000094**	**0.000034**	**0.000067**	**0.000215**
Indicador ϵ-unario	NSGA-II evaluaciones	1.732627	0.082842	1.577305	1.887822
	Auto-Adaptación	**1.008537**	**0.004132**	**1.003963**	**1.021298**
Dispersión	NSGA-II evaluaciones	0.972776	0.067786	0.880334	1.113631
	Auto-Adaptación	**0.804278**	**0.013795**	**0.786544**	**0.828621**

Tabla 6.33: Métricas del problema DTLZ6

Problema		DTLZ6			
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II generaciones	0.002186	0.000930	**0.000070**	0.003638
	Auto-Adaptación	**0.000094**	**0.000034**	0.000067	**0.000215**
Indicador ϵ-unario	NSGA-II generaciones	1.103957	0.040262	1.004424	1.163503
	Auto-Adaptación	**1.008537**	**0.004132**	**1.003963**	**1.021298**
Dispersión	NSGA-II generaciones	**0.782302**	0.025047	**0.747496**	0.846386
	Auto-Adaptación	0.804278	**0.013795**	0.786544	**0.828621**

Tabla 6.34: Métricas del problema DTLZ6

DTLZ6 Cobertura de 2 conjuntos				
% que Domina A a B	Media	σ	Peor	Mejor
NSGA-II evaluaciones a Auto-adaptación	0.6431	**0.024605**	0.6	0.694
Auto-adaptación a NSGA-II evaluaciones	**1**	0	**1**	**1**
NSGA-II generaciones a Auto-adaptación	0.7659	0.01956	0.718	0.806
Auto-adaptación a NSGA-II generaciones	**0.9893**	**0.04664**	**0.786**	**1**

Tabla 6.35: Resultados de la métrica cobertura de dos conjuntos del problema DTLZ6

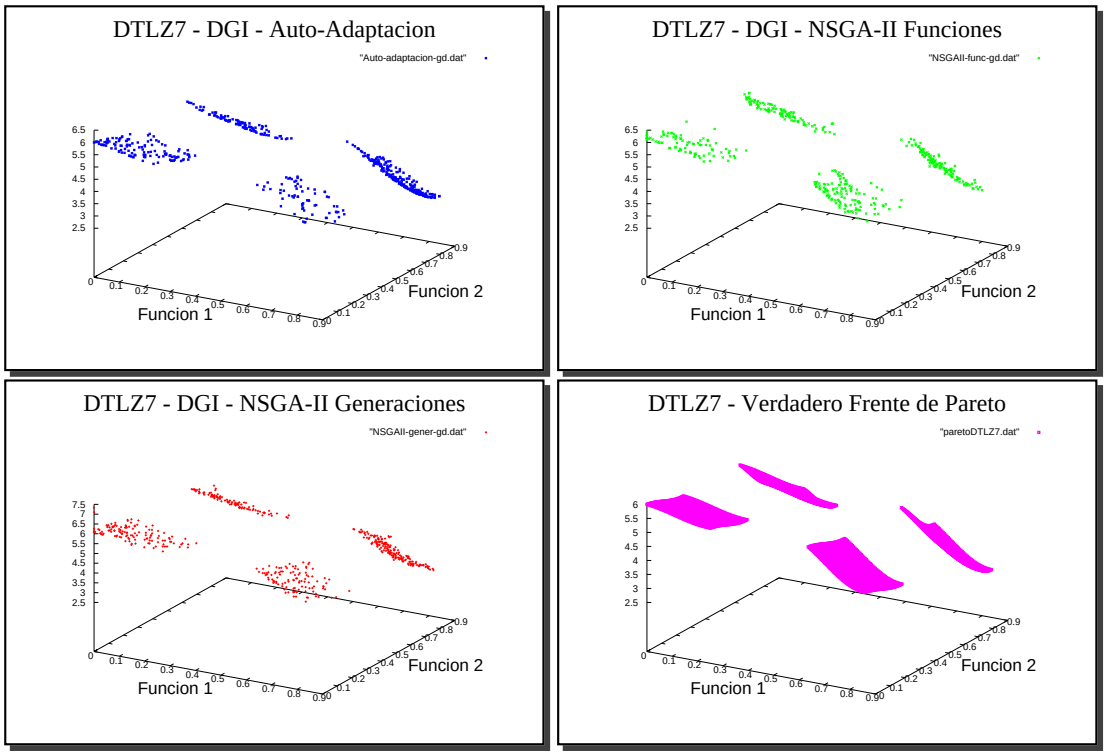


Figura 6.12: Comparación de las ejecuciones promedio por métrica y el verdadero frente de Pareto

de Pareto, mientras que el NSGA-II generaciones tiene algunas soluciones un poco apartadas del verdadero frente de Pareto.

En cuanto a las métricas (ver tablas 6.33 y 6.34), nuestra propuesta muestra ser superior en la distancia generacional inversa y la el indicador ϵ-unario. Sin embargo, en cuanto a dispersión, el NSGA-II en sus dos formas, logra obtener mejores resultados, si bien las diferencias son marginales en todos los casos.

Problema	DTLZ7				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II evaluaciones	0.000817	0.000125	0.000603	0.001076
	Auto-Adaptación	**0.000389**	**0.000016**	**0.000361**	**0.000425**
Indicador ϵ-unario	NSGA-II evaluaciones	1.081822	0.021139	1.050040	1.138003
	Auto-Adaptación	**1.026261**	**0.003132**	**1.020909**	**1.033658**
Dispersión	NSGA-II evaluaciones	**0.478671**	0.038158	**0.404630**	**0.551611**
	Auto-Adaptación	0.531645	**0.021466**	0.494259	0.583806

Tabla 6.36: Métricas del problema DTLZ7

Problema	DTLZ7				
Métrica	Algoritmo	Media	σ	Mejor	Peor
DG Invertida	NSGA-II generaciones	0.000579	0.000062	0.000469	0.000771
	Auto-Adaptación	**0.000389**	**0.000016**	**0.000361**	**0.000425**
Indicador ϵ-unario	NSGA-II generaciones	1.057326	0.011657	1.039717	1.073805
	Auto-Adaptación	**1.026261**	**0.003132**	**1.020909**	**1.033658**
Dispersión	NSGA-II generaciones	**0.440922**	0.030344	**0.385480**	**0.507187**
	Auto-Adaptación	0.531645	**0.021466**	0.494259	0.583806

Tabla 6.37: Métricas del problema DTLZ7

En la tabla 6.38 se muestran los resultados de la métrica de cobertura de 2 conjuntos, apreciándose que nuestra propuesta tiene un comportamiento claramente superior al del NSGA-II en sus dos formas.

DTLZ7 Cobertura de 2 conjuntos				
% que Domina A a B	Media	σ	Peor	Mejor
NSGA-II evaluaciones a Auto-adaptación	0.000100	0.000436	0	0.002000
Auto-adaptación a NSGA-II evaluaciones	**0.825055**	**0.046788**	**0.732**	**0.938596**
NSGA-II generaciones a Auto-adaptación	0.000300	0.000954	0	0.004
Auto-adaptación a NSGA-II generaciones	**0.736087**	**0.054787**	**0.614**	**0.834**

Tabla 6.38: Resultados de la métrica cobertura de dos conjuntos del problema DTLZ7

6.13 Resultados generales

Nuestra propuesta logra ser competitiva y en varios casos superior al NSGA-II.

Para el caso donde se realiza el mismo número de evaluación de evaluaciones, se puede decir que la comparación es más justa. Nuestra propuesta muestra ser superior en la gran mayoría de los casos.

En cuanto al NSGA-II generaciones, el algoritmo NSGA-II cuenta con un mayor número de evaluaciones que nuestra propuesta, lo que le da cierta ventaja. Sin embargo, en este caso, los resultados siguen siendo favorables para nuestra propuesta, que se mantiene competitiva y, en varios casos, superior.

Estos resultados hacen evidente la viabilidad de nuestra propuesta con auto-adaptación, ya que ésta obtiene no sólo resultados equiparables a los de un algoritmo cuyos parámetros se ajustaron manualmente, sino que además, en algunos casos, incluso los superan.

7

Conclusiones

El análisis de la sensibilidad de un algoritmo evolutivo multi-objetivo a sus parámetros nos otorgó datos muy relevantes para trabajos futuros sobre auto-adaptación, ya que nos indicó que los valores del porcentaje de cruza y el tipo de cruza para codificaciones binarias se pueden establecer en un valor fijo para problemas que tengan las características de la serie de problemas ZDT y DTLZ (Apéndice A y B).

Nuestra propuesta, que implementa un conjunto de técnicas de auto-adaptación, no requiere la especificación de ningún parámetro extra. Además, elimina la definición de los parámetros básicos de un algoritmo evolutivo multi-objetivo. De esta forma, no se le requiere al usuario la definición de los diferentes parámetros que típicamente deben ajustarse manualmente en un algoritmo evolutivo multi-objetivo. Se hace notar que no se eliminaron los parámetros, sino se emplean diferentes técnicas de auto-adaptación, que regulan los valores que deben tomar cada uno de ellos.

Las diferentes técnicas fueron implementadas sobre el algoritmo del estado del arte NSGA-II [24], el cual muestra un desempeño muy competitivo con respecto a algoritmos evolutivos multi-objetivo similares. Para validar nuestra propuesta se efectuó una comparación de resultados entre el NSGA-II en su forma original y el NSGA-II con nuestras técnicas propuestas de

auto-adaptación.

Nuestra propuesta mostró ser bastante competitiva y en varios casos logró incluso superar al NSGA-II original en su desempeño, además de no requerir un ajuste empírico de parámetros. Esto vuelve al algoritmo propuesto una mejor opción para el usuario que busca usar fácilmente un algoritmo evolutivo multi-objetivo sin necesidad de volverse un experto en su diseño y utilización.

Las técnicas presentadas no requieren ser auxiliadas por algoritmos adicionales de cómputo suave (como redes neuronales ó mapas auto-organizados), sino que requieren únicamente información obtenida del mismo proceso evolutivo. Buscamos que los individuos, la población y el paso de las generaciones nos indiquen el camino a seguir, sin olvidar que la diversidad debe mantenerse para estar en continuo enriquecimiento de los individuos, así como la mejora de los mismos.

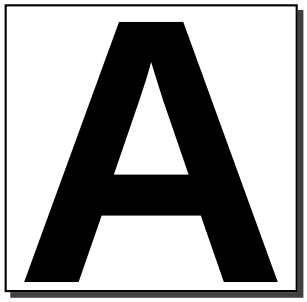
Nuestra propuesta evita la tarea de realizar múltiples ejecuciones y pruebas empíricas de los parámetros que logren el mejor desempeño posible de un algoritmo evolutivo multi-objetivo al usarse en un problema arbitrario.

7.1 Trabajo a futuro

Nuestra propuesta no implementa ningún técnica para el manejo de restricciones, por lo que a futuro sería muy recomendable implementar algún tipo de técnica que continúe en la misma línea trazada en esta tesis. Es decir, la técnica adoptada no debe agregar ningún parámetro, o bien éstos deben ser auto-adaptados. Esto nos permitirá preservar el objetivo principal de esta tesis: contar con un algoritmo evolutivo multi-objetivo libre de parámetros que deban ser definidos empíricamente por el usuario.

En nuestra propuesta se implementan cinco tipos de cruzas y cuatro tipos de mutaciones (Apéndice D). Sin embargo, es posible agregar más tipos de cruza o de mutación, por lo que se propone como trabajo a futuro realizar un estudio más exhaustivo del comportamiento de las cruzas y mutaciones, a fin de contar con mecanismos más generales de auto-adaptación.

También implementamos dos tipos de codificaciones, por lo que también se presenta como un trabajo a futuro el implementar otros tipos de codificaciones (por ejemplo, entera) y extender los mecanismos de auto-adaptación correspondientes.



Funciones de Prueba

Función	Objetivos	Límites	Características
ZDT1	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}, g) = 1 - \sqrt{f_1/g(\vec{x})}$ $g(\vec{x}) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i$	n = 30 $0 \leq x_i \leq 1$, $i = 1, 2, \ldots, 30$	convexa, conectado
ZDT2	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}, g) = 1 - (f_1/g(\vec{x}))^2$ $g(\vec{x}) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i$	n = 30 $0 \leq x_i \leq 1$, $i = 1, 2, \ldots, 30$	no convexa, conectado
ZDT3	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}, g) = 1 - \sqrt{f_1/g(\vec{x})} - (f_1/g(\vec{x})) \sin(10\pi f_1)$ $g(\vec{x}) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i$	n = 30 $0 \leq x_i \leq 1$, $i = 1, 2, \ldots, 30$	discontinuo
ZDT4	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}, g) = 1 - \sqrt{f_1/g(\vec{x})}$ $g(\vec{x}) = 1 + 10(n-1) + \sum_{i=2}^{n} (x_i^2 - 10\cos(4\pi x_i))$	n = 10 $0 \leq x_1 \leq 1$, $-5 \leq x_i \leq 5$, $i = 2, 3 \ldots, 10$	no convexo, multi frontal (21^9 frentes locales)
ZDT6	$f_1(\vec{x}) = 1 - \exp(-4x_1) \sin^6(6\pi x_1)$ $f_2(\vec{x}, g) = 1 - (f_1/g(\vec{x}))^2$ $g(\vec{x}) = 1 + 9[(\sum_{i=2}^{10} x_i)/9]$	n = 10 $0 \leq x_i \leq 1$, $i = 2, 3 \ldots, 10$	no convexa

Tabla A.1: Definición y descripción de las 5 funciones de prueba sin restricciones del grupo ZDT utilizadas en esta tesis.

Función	Objetivos	Límites	Características
DTLZ1	$f_1(\vec{x}) = \frac{1}{2}x_1x_2(1+g(\vec{x}))$ $f_2(\vec{x}) = \frac{1}{2}x_1(1-x_2)(1+g(\vec{x}))$ $f_3(\vec{x}) = \frac{1}{2}(1-x_1)(1+g(\vec{x}))$ $g(\vec{x}) = 100[5+\sum_{i=3}^{n}(x_i-0.5)^2-\cos(20\pi(x_i-0.5))]$	n = 7 $0 \leq x_i \leq 1$ $i = 1,\ldots,7$	frente de Pareto lineal, multi frontal (11^5-1 frentes locales)
DTLZ2	$f_1(\vec{x}) = \cos(\frac{\pi}{2}x_1)\cos(\frac{\pi}{2}x_2)(1+g(\vec{x}))$ $f_2(\vec{x}) = \cos(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2)(1+g(\vec{x}))$ $f_3(\vec{x}) = \sin(\frac{\pi}{2}x_1)(1+g(\vec{x}))$ $g(\vec{x}) = \sum_{i=3}^{n}(x_i-0.5)^2$	n = 12 $0 \leq x_i \leq 1$ $i = 1,\ldots,12$	convexo, conectado
DTLZ3	$f_1(\vec{x}) = \cos(\frac{\pi}{2}x_1)\cos(\frac{\pi}{2}x_2)(1+g(\vec{x}))$ $f_2(\vec{x}) = \cos(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2)(1+g(\vec{x}))$ $f_3(\vec{x}) = \sin(\frac{\pi}{2}x_1)(1+g(\vec{x}))$ $g(\vec{x}) = 100[10+\sum_{i=3}^{n}(x_i-0.5)^2-\cos(20\pi(x_i-0.5))]$	n = 12 $0 \leq x_i \leq 1$ $i = 1,\ldots,12$	no convexo, multi frontal ($3^{10}-1$ frentes locales
DTLZ4	$f_1(\vec{x}) = \cos(\frac{\pi}{2}x_1^{\alpha})\cos(\frac{\pi}{2}x_2^{\alpha})(1+g(\vec{x}))$ $f_2(\vec{x}) = \cos(\frac{\pi}{2}x_1^{\alpha})\sin(\frac{\pi}{2}x_2^{\alpha})(1+g(\vec{x}))$ $f_3(\vec{x}) = \sin(\frac{\pi}{2}x_1^{\alpha})(1+g(\vec{x}))$ $g(\vec{x}) = \sum_{i=3}^{12}(x_i-0.5)^2$	n = 12; $\alpha = 100$ $0 \leq x_i \leq 1$ $i = 1,\ldots,12$	convexo, conectado
DTLZ5	$f_1(\vec{x}) = \cos(\frac{\pi}{2}\theta_1)\cos(\frac{\pi}{2}\theta_2)(1+g(\vec{x}))$ $f_2(\vec{x}) = \cos(\frac{\pi}{2}\theta_1)\sin(\frac{\pi}{2}\theta_2)(1+g(\vec{x}))$ $f_3(\vec{x}) = \sin(\frac{\pi}{2}\theta_1)(1+g(\vec{x}))$ $\theta_1 = x_1 \cdot \left(\frac{\pi}{2}\right)$, $\theta_2 = \frac{\pi}{4\cdot(1+g(\vec{x}))} \cdot (1+2x_2 \cdot g(\vec{x}))$, $g(\vec{x}) = \sum_{i=3}^{n}(x_i-0.5)^2$	n = 12 $0 \leq x_i \leq 1$ $i = 1,\ldots,12$	frente lineal curvo
DTLZ6	$f_1(\vec{x}) = \cos(\frac{\pi}{2}\theta_1)\cos(\frac{\pi}{2}\theta_2)(1+g(\vec{x}))$ $f_2(\vec{x}) = \cos(\frac{\pi}{2}\theta_1)\sin(\frac{\pi}{2}\theta_2)(1+g(\vec{x}))$ $f_3(\vec{x}) = \sin(\frac{\pi}{2}\theta_1)(1+g(\vec{x}))$ $\theta_1 = x_1 \cdot \left(\frac{\pi}{2}\right)$, $\theta_2 = \frac{\pi}{4\cdot(1+g(\vec{x}))} \cdot (1+2x_2 \cdot g(\vec{x}))$, $g(\vec{x}) = \sum_{i=3}^{n}(x_i)^{0.1}$	n = 12 $0 \leq x_i \leq 1$ $i = 1,\ldots,12$	más dificil que DTLZ5
DTLZ7	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = x_2$ $f_3(\vec{x}) = (1+g(\vec{x}))\cdot h(f_1,f_2,g(\vec{x}))$ $g(\vec{x}) = 1+\frac{9}{22}\cdot\sum_{i=3}^{n}(x_i)$ $h(f_1,f_2,g(\vec{x})) = 3-\sum_{i=1}^{2}\left(\frac{f_i}{1+g}(1+sin(3\pi f_i))\right)$	n = 22 $0 \leq x_i \leq 1$ $i = 1,\ldots,22$	discontinuo

Tabla A.2: Definición y descripción de las 7 funciones de prueba sin restricciones del grupo DTLZ utilizadas en esta tesis.

B

Gráficas del Frente de Pareto Verdadero

En las siguientes figuras se muestra de forma gráfica el frente de Pareto real de cada una de las funciones del Apéndice A.

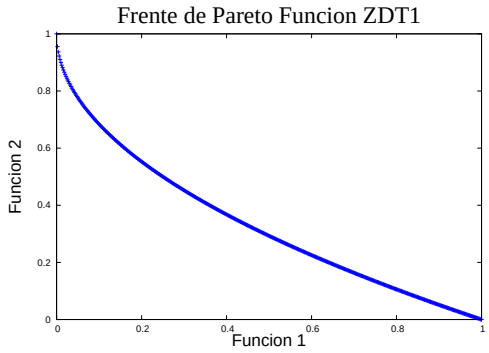


Figura B.1: Frente de Pareto ZDT1

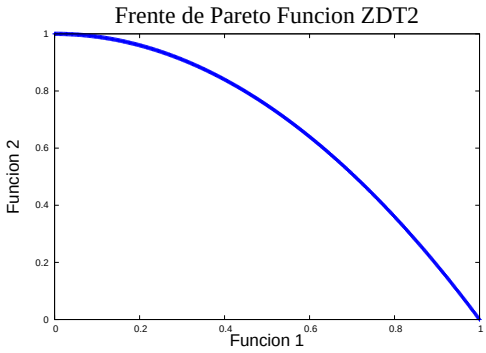


Figura B.2: Frente de Pareto ZDT2

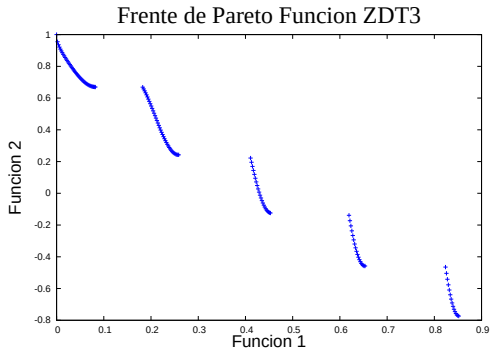


Figura B.3: Frente de Pareto ZDT3

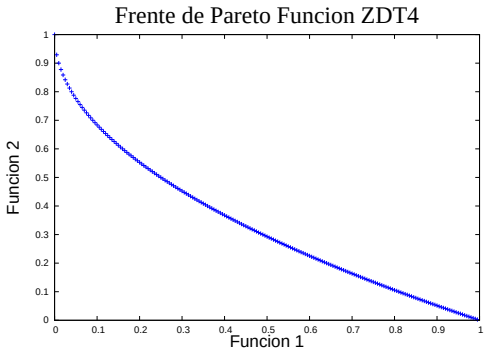


Figura B.4: Frente de Pareto ZDT4

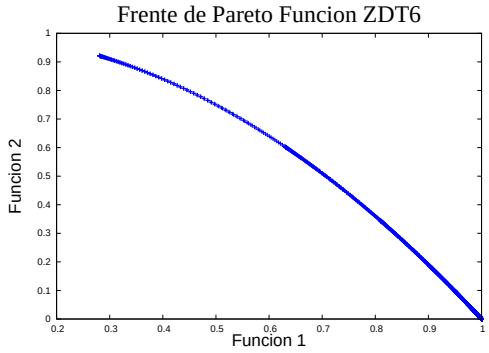


Figura B.5: Frente de Pareto ZDT6

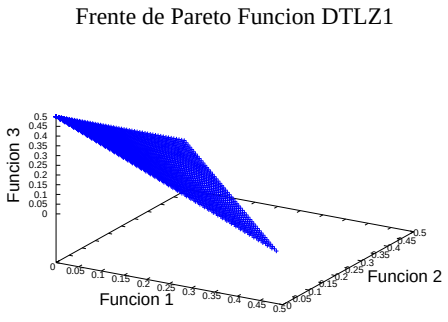


Figura B.6: Frente de Pareto DTLZ1

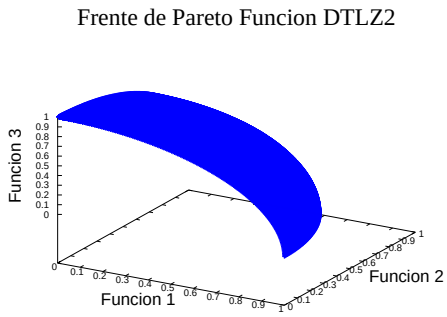


Figura B.7: Frente de Pareto DTLZ2

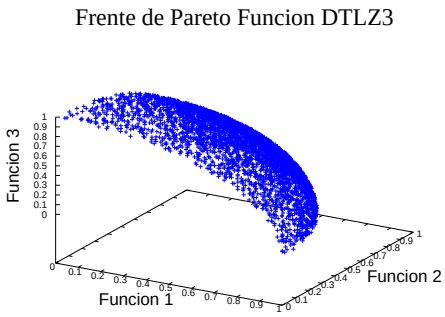


Figura B.8: Frente de Pareto DTLZ3

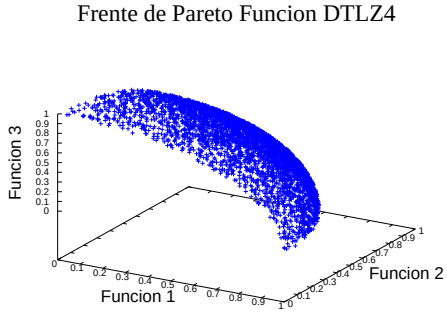


Figura B.9: Frente de Pareto DTLZ4

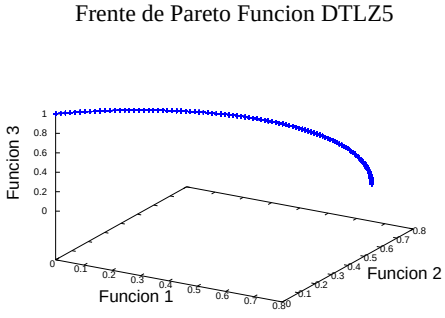


Figura B.10: Frente de Pareto DTLZ5

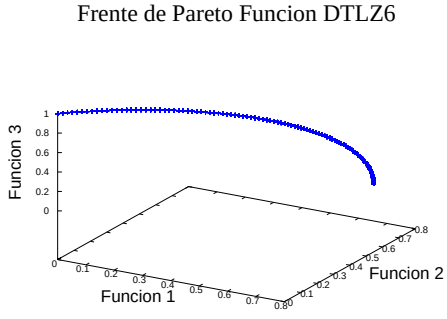


Figura B.11: Frente de Pareto DTLZ6

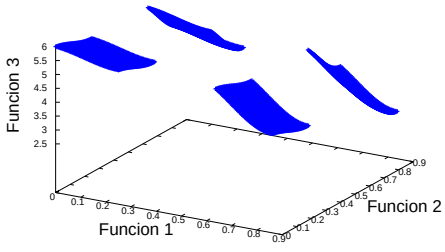
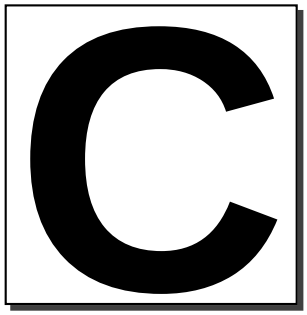


Figura B.12: Frente de Pareto DTLZ7



Medidas de Desempeño

A continuación se definen y describen las medidas de desempeño utilizadas en esta tesis para realizar el análisis del desempeño de los algoritmos estudiados.

Distancia Generacional Invertida(DG Invertida)

La distancia generacional [92] representa el promedio de la distancia de los vectores del frente obtenido (F_{actual}) al verdadero frente de Pareto ($F_{verdadero}$). Está definida de la siguiente forma:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$$

donde n es el número de vectores de F_{actual} y d_i es la distancia euclidiana de cada vector de F_{actual} con el vector más cercano de $F_{verdadero}$.

Una $GD = 0$ indica que F_{actual} pertenece a $F_{verdadero}$. Cualquier otro valor distinto indica el grado en que se aleja F_{actual} de $F_{verdadero}$.

En cuanto a la distancia generacional invertida [92], ésta representa el promedio de la distancia de los vectores del conjunto $F_{verdadero}$ al conjunto F_{actual}. Se define de la siguiente manera:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n}$$

donde n es el número de vectores en $F_{verdadero}$ y d_i es la distancia euclidiana de cada vector $F_{verdadero}$ con respecto al vector más cercano de F_{actual}.

Lo que la distancia generacional invertida pretende es medir el progreso global hacia el verdadero frente de Pareto y la extensión del conjunto F_{actual} con respecto al $F_{verdadero}$.

Un frente obtenido que esté lejano del verdadero frente o que sólo abarque una región pequeña de todo el frente se verá penalizado por la esta medida de desempeño. $IGD = 0$ indica que F_{actual} pertenece a $F_{verdadero}$, y cualquier otro valor indica qué tan lejos está F_{actual} de $F_{verdadero}$.

Indicador ϵ-unario

El indicador ϵ [106] cuenta con una versión multiplicativa y una aditiva, además de poder ser unario (comparando un conjunto de vectores contra el verdadero frente de Pareto) o binaria (comparando dos conjuntos de vectores).

Para los fines de esta tesis se utilizó el indicador epsilon multiplicativo unario [106], al que llamaremos indicador ϵ-unario. El indicador ϵ-unario de F_{actual} nos arroja un valor ϵ por el cual se debe multiplicar a cada vector de $F_{verdadero}$ para que este nuevo conjunto de vectores sea dominado por F_{actual}.

$$\epsilon_{unaria}(F_{actual}) = inf_{\epsilon \in \mathbb{R}}\{\forall z^2 \in F_{verdadero} | \exists z^1 \in F_{actual} : z_i^2 \leq \epsilon \cdot z_i^1 \ \forall i\}$$

El mejor valor que esta medida de desempeño nos puede arrojar es 1, indicándonos que F_{actual} pertenece a $F_{verdadero}$. Cualquier otro valor nos indicará la lejanía de F_{actual} con respecto a $F_{verdadero}$.

Dispersión

La dispersión (*spread*), porpuesta por Deb [24], también conocida como medida de desempeño Δ, mide la distribución de las soluciones a lo largo del verdadero frente.

Esta medida de desempeño se define de la siguiente forma:

$$\Delta = \frac{\sum_{i=1}^{k} d_i^e + \sum_{i=1}^{|F_{verdadero}|} \left| d_i - \bar{d} \right|}{\sum_{i=1}^{k} d_i^e + |F_{verdadero}| \, \bar{d}}$$

donde d_i^e es la distancia de las soluciones extremas de la i-ésima función objetivo en F_{actual} y $F_{verdadero}$. $|F_{verdadero}|$ es la cantidad de soluciones contenidas en $F_{verdadero}$. d_i es la distancia entre cada punto en F_{actual} al punto más cercano en $F_{verdadero}$. $\bar{d}$ es el promedio de todas las distancias d_i.

Podemos observar que $0 \leq \Delta \leq 1$ y entre más pequeño sea el valor de esta medida de desempeño, mejor es la distribución de las soluciones. Una distribución perfecta arrojará $\Delta = 0$, donde todas las soluciones tienen un valor d_i igual y las soluciones abarcan todo el verdadero frente de Pareto.

Cobertura de 2 conjuntos

Conocida como *Two Set Coverage*, esta medida de desempeño fue propuesta por Zitzler [102], y realiza una comparación entre dos conjuntos. Se define de la siguiente forma:

$$C(A, B) = \frac{|\{b \in B; \exists a \in A : b \preceq a\}|}{|B|}$$

Esta medida de desempeño nos arrojará un valor de $C = 1$ si todos los puntos en A dominan o igualan a todos los puntos en B. En caso opuesto, el valor de la métrica será $C = 0$. La cobertura de dos conjuntos representa el porcentaje total de las soluciones en B que son dominadas por las soluciones en A.

Hipervolumen

El hipervolumen, propuesto por Zitzler [101], también es conocido como *Size of the dominated space* (tamaño del espacio dominado).

El hipervolumen calcula el volumen, en el espacio de las funciones objetivo, que cubren los miembros del frente obtenido F_{actual}. Esto representa el tamaño de la región que domina F_{actual} a partir de un punto de referencia. La métrica se define de la siguiente forma:

$$H = \bigcup_{i=1}^{n} R\left(u_i, u^*\right)$$

donde la función R es la región acotada por u_i y el vector de referencia u^*. F_{actual} cuenta con n vectores, $u^* \notin F_{actual}$ y todos los vectores de F_{actual} deben dominar a u^*. La función R se define, tomando en cuenta k objetivos, de la siguiente forma:

$$R\left(u_i, u^*\right) = \left\{v | v < u^*, u_i < v, v \in \mathbb{R}^k\right\}$$

El vector de referencia debe ser cuidadosamente elegido, ya que, para problemas de minimización será muy diferente que para problemas de maximización.

D

Tipos de Cruza y Mutación para Codificación Real

A continuación se definen y describen los tipos de cruza y mutación utilizadas en las técnicas de auto-adaptación propuestas en esta tesis.

D.1 Cruzas reales

Definimos a dos padres con n variables de decisión como $P_1 = \langle v_1, v_2, \ldots, v_n \rangle$ y $P_2 = \langle w_1, w_2, \ldots, w_n \rangle$

SBX

Simulated Binary Crossover (SBX) [44] se define de la siguiente forma, tomando en cuenta que los dos hijos resultantes se definen como $H^1 = \langle h_1^1, h_2^1, \ldots, h_n^1 \rangle$ y $H^2 = \langle h_1^2, h_2^2, \ldots, h_n^2 \rangle$:

La función $random()$ genera un número aleatorio $u \in (0, 1)$.

$|v_i - w_i|$ representa el valor absoluto de la diferencia ente v_i y w_i.

Algoritmo 5 Algoritmo SBX

for $i = 1$ hasta n **do**
  $u_1 = random()$
  if $u_1 \leq 0.5$ **then**
    if $|v_i - w_i| > 10^{-6}$ **then**
      $y_1 = MIN\{v_i, w_i\}$
      $y_2 = MAX\{v_i, w_i\}$
      if $(y_1 - LB_i) > (UB_i - y_2)$ **then**
        $\beta = 1 + \left(\frac{2\times(UB_i - y_2)}{(y_2 - y_1)}\right)$
      else
        $\beta = 1 + \left(\frac{2\times(y_1 - LB_i)}{(y_2 - y_1)}\right)$
      end if
      $\alpha = 2 - \left(\frac{1}{\beta}\right)^{n_c+1}$
      $u_2 = random()$
      if $u_2 \leq \frac{1}{\alpha}$ **then**
        $\beta' = (\alpha \times u_2)^{\frac{1}{n_c+1}}$
      else
        $\beta' = \left(\frac{1}{2-(\alpha\times u_2)}\right)^{\frac{1}{n_c+1}}$
      end if
      $h_i^1 = 0.5 \times ((y_1 + y_2) - (\beta' \times (y_2 - y_1)))$
      $h_i^2 = 0.5 \times ((y_1 + y_2) + (\beta' \times (y_2 - y_1)))$
    else
      $h_i^1 = 0.5 \times ((v_i + w_i) - (1.0 \times (w_i - v_i)))$
      $h_i^2 = 0.5 \times ((v_i + w_i) + (1.0 \times (w_i - v_i)))$
    end if
    if $h_i^1 < LB_i$ **then**
      $h_i^1 = LB_i$
    end if
    if $h_i^1 > UB_i$ **then**
      $h_i^1 = UB_i$
    end if
    if $h_i^2 < LB_i$ **then**
      $h_i^2 = LB_i$
    end if
    if $h_i^2 > UB_i$ **then**
      $h_i^2 = UB_i$
    end if
  else
    $h_i^1 = v_i$
    $h_i^2 = w_i$
  end if
end for

$MIN\{v_i, w_i\}$ nos devuelve el menor de los dos valores, mientras que $MAX\{v_i, w_i\}$ devuelve al mayor.

LB_i y UB_i representan el límite inferior y el límite superior, respectivamente, de la i-ésima variable de decisión.

Para el índice de distribución de la cruza de las variables reales (n_c) los autores sugieren n_c =1 ó 2. Para fines de nuestra implementación el valor es de 1.

Simple

Muy similar a la cruza de un punto en codificación binaria, pero aplicada a la real.

Tomamos un valor k que nos indicará en dónde realizar la cruza, por lo cual $k \in (1, n)$. Los hijos quedarían de la siguiente forma:

$$H_1 = \langle v_1, \ldots, v_k, w_{k+1}, \ldots, w_n \rangle$$
$$H_2 = \langle w_1, \ldots, w_k, v_{k+1}, \ldots, v_n \rangle$$

Uniforme

Similar a la cruza uniforme para la codificación binaria. Usa la función $flip(p)$, que regresa un valor de verdadero ($true$) con una probabilidad p. Definimos el valor de la variable i del hijo 1 como H_i^1 y a la del hijo 2 como H_i^2. Los hijos se definen de la siguiente manera:

$$\begin{cases} H_i^1 = v_i, H_i^2 = w_i & \text{si } flip(p) = true \\ H_i^2 = v_i, H_i^1 = w_i & \text{de lo contrario} \end{cases}$$

El procedimiento anterior se debe realizar $\forall i \in (1, n)$. Para nuestro caso, el valor de p es de 0.7 (la probabilidad de cruza).

Intermedia

Para esta cruza definimos un valor $k \in (1, n)$ y un valor de $a \in (0, 1)$, los hijos serán:

$$H_1 = \langle v_1, \ldots, v_k, w_{k+1} * a + v_{k+1} * (1 - a), \ldots, w_n * a + v_n * (1 - a) \rangle$$

$$H_2 = \langle w_1, \ldots, w_k, v_{k+1} * a + w_{k+1} * (1-a), \ldots, v_n * a + w_n * (1-a) \rangle$$

Dos Puntos

Esta cruza es similar a la cruza de dos puntos para codificación binaria y es también similar a la cruza simple antes mencionada, con la diferencia de que aquí se usan dos puntos.

Definimos dos valores $k_1, k_2 \in (1, n)$ donde $k_1 < k_2$. Los hijos generados son los siguientes:

$$H_1 = \langle v_1, \ldots, v_{k_1}, w_{k_1+1}, \ldots, w_{k_2}, v_{k_2+1}, \ldots, v_n \rangle$$
$$H_2 = \langle w_1, \ldots, w_{k_1}, v_{k_1+1}, \ldots, v_{k_2}, w_{k_2+1}, \ldots, w_n \rangle$$

D.2 Mutaciones

Definimos a un individuo con n variables de decisión como $P = \langle V_i, \ldots, V_n \rangle$.

También definimos la función $flip(p)$ que devuelve el valor de verdadero ($true$) con una probabilidad p.

Parameter-Based Mutation

Propuesta en [44]. Su procedimiento es el siguiente:

1. Se genera un número aleatorio $u \in (0, 1)$
2. Se obtiene

$$\bar{\delta} = \begin{cases} (2u)^{\frac{1}{\eta_m + 1}} & \text{si } u \leq 0.5 \\ UB & \text{de lo contrario} \end{cases}$$

   donde η_m es el índice de distribución de la mutación de la variables reales, que, para fines de nuestra implementación, es: $\eta_m = 50$.

3. El nuevo valor quedará de la siguiente forma:

$$V'_k = V_k + \bar{\delta}\Delta_{max}$$

donde Δ_{max} es el grado máximo de perturbación permitida, y se define como $\Delta_{max} = UB - LB$, donde LB es el límite inferior de la variable y UB el límite superior.

No uniforme

Propuesta en [61], define al nuevo valor de la siguiente manera:

$$V'_k = \begin{cases} V_k + \Delta(t, UB - V_k) & \text{si } flip(0.5) = true \\ V_k - \Delta(t, V_k - LB) & \text{de lo contrario} \end{cases}$$

donde LB es el límite inferior de la variable y UB el límite superior. Y:

$$\Delta(t, y) = y * \left(1 - r^{\left(1 - \frac{t}{T}\right)^b}\right)$$

donde r es un número aleatorio entre 0 y 1, T es el número máximo de generaciones y b es un parámetro que define el grado de no uniformidad de la mutación (el autor sugiere $b = 5$, y éste es el valor usado en nuestra propuesta).

En cuanto al parámetro T no lo podemos definir como tal, ya que nuestra propuesta no define un número de generaciones máximo. Por ello, en las primeras 300 generaciones el valor es de 300. Al alcanzar cualquier número de generación múltiplo de 300, el valor de T se incrementa en 300. Es decir, al llegar a la generación 900 debemos tener un valor de T de 900 y modificarlo agregándole 300 para dejarlo en 1200.

Uniforme

Para este tipo de mutación, el nuevo valor se define de la siguiente manera:

$$V'_k = rnd(LB, UB)$$

donde la función rnd genera un número real aleatorio con distribución uniforme. LB es el límite inferior de la variable, y UB el límite superior.

De límite

Tomando a LB como el límite inferior de la variable y a UB como el límite superior, tenemos:

$$V'_k = \begin{cases} LB & \text{si } flip(0.5) = true \\ UB & \text{de lo contrario} \end{cases}$$

Bibliografía

[1] Abbass, H. A. The Self-Adaptive Pareto Differential Evolution Algorithm. In *Congress on Evolutionary Computation (CEC'2002)* (Piscataway, New Jersey, May 2002), vol. 1, IEEE Press, pp. 831–836.

[2] Abbasss, H. A. A pareto differential evolution approach to vector optimization problems. In *The IEEE Congress on Evolutionary Computation* (Seul, Korea, 2001), pp. 971–978.

[3] Arabas, J., Michalewicz, Z., and Mulawka, J. J. Gavaps - a genetic algorithm with varying population size. In *International Conference on Evolutionary Computation* (1994), pp. 73–78.

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

[5] Bäck, T., and David B. Fogel, y. Z. M., Eds. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, Nueva York, EE. UU., 1997.

[6] Bienert, P. Aufbau einer optimierungsautomatik f¨ur drei parameter. Master's thesis, Universidad Técnica de Berlin, 1967.

[7] Booker, L. B. *Intelligent Behavior as an Adaptation to the Task Environment*. PhD thesis, Logic of Computers Group, University of Michigan, Ann Arbor, Michigan, EE. UU., 1982.

[8] Bremerman, H. J. Optimization through evolution and recombination in self-organizing systems. In *Spartan Books* (Washington D.C., EE. UU., 1962), pp. 93–106.

[9] Brindle, A. *Genetic Algorithms for Function Optimization*. PhD thesis, Department of Computer Science, University of Alberta, Edmonton, Alberta, Canada, 1981.

[10] Büche, D., Guidati, G., Stoll, P., and Koumoutsakos, P. Self-organazing maps for pareto optimization of airfoils. In *Parallel Problem Solvinf from Nature/PPSN VII* (Granada, Spain, September 2002), J. J. M. G. et al., Ed., no. 2439 in Lecture Notes in Computer Science, Springer-Verlag, pp. 122–131.

[11] Coello Coello, C. A. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal 1*, 3 (August 1999), 269–308.

[12] Coello Coello, C. A., and Mariano Romero, C. E. Evolutionary Algorithms and Multiple Objective Optimization. In *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, M. Ehrgott and X. Gandibleux, Eds. Kluwer Academic Publishers, Boston, 2002, pp. 277–331.

[13] Coello Coello, C. A., and Toscano Pulido, G. A Micro-Genetic Algorithm for Multiobjective Optimization. In *First International Conference on Evolutionary Multi-Criterion Optimization*, E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, Eds. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001, pp. 126–140.

[14] Coello Coello, C. A., and Toscano Pulido, G. Multiobjective optimization using a micro-genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO´2001)* (San Francisco California, 2001), L. S. et al., Ed., Morgan Kaufmann Publishers, pp. 274–282.

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

[16] Cooper, W. W., Charnes, A., and Ferguson., R. O. Optimal estimation of executive compensation by linear programming. *Management Science 1*, 2 (1955), 138–151.

[17] Corne, D. W., Jerram, N. R., Knowles, J. D., and Oates, M. J. PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)* (San Francisco, California, 2001), L. Spector, E. D. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, Eds., Morgan Kaufmann Publishers, pp. 283–290.

[18] Darwin, C. R. *The Origin of Species by Means of Natural Selection, or The Preservation of Favoured Races in the Struggle for Life*. Penguin Books, Nueva York, EE. UU, 1959.

[19] de Castro, L. N., and Timmis, J. *Artificial Immune Systems: A new Computational Intelligence Approach*. Springer Verlag, 2002.

[20] Deb, K. Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design. In *Evolutionary Algorithms in Engineering and Computer Science*, K. Miettinen, M. M. Mäkelä, P. Neittaanmäki, and J. Periaux, Eds. John Wiley & Sons, Ltd, Chichester, UK, 1999, ch. 8, pp. 135–161.

[21] Deb, K. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.

[22] Deb, K. Evolutionary Multi-Objective Optimization Without Additional Parameters. In *Parameter Setting in Evolutionary Algorithms*, F. G. Lobo, C. F. Lima, and Z. Michalewicz, Eds. Springer-Verlag, Berlin, 2007, pp. 241–257.

[23] Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature VI Conference* (Paris, France, 2000), M. Schoenauer,

K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds., Springer. Lecture Notes in Computer Science No. 1917, pp. 849–858.

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

[25] DeJong, K. A. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, EE. UU., 1975.

[26] Dorigo, M., Maniezzo, V., and Colorni, A. The ant system: Optimization by a colony of cooperating agents. In *IEEE Transactions on Systems, Man, and Cybernetics-Part B 26* (1996), pp. 29–41.

[27] Edgeworth, F. Y. *Mathematical Physics*. P. Keagan, 1881.

[28] Eiben, A. E., Michalewicz, Z., Schoenauer, M., and Smith, J. E. Parameter control in evolutionary alkgorithms. In *Parameter Setting in Evolutionary Algorithms*, F. G. Lobo, C. F. Lima, and Z. Michalewicz, Eds. Springer-Verlag, Berlin, 2007, pp. 19–46.

[29] Erickson, M., Mayer, A., and Horn, J. The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems. In *First International Conference on Evolutionary Multi-Criterion Optimization*, E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, Eds. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001, pp. 681–695.

[30] Fishburn., P. C. Lexicographic orders, utilities and decision rueles: A survey. *Management Science 20*, 11 (1974), 1442–1471.

[31] Fogel, D. B., Ed. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*. The Institute of Electrical and Electronic Engineers, New York, EE. UU., 1995.

[32] Fogel, D. B., Ed. *Evolutionary Computation. The Fossil Record. Selected Readings on the History of Evolutionary Algorithms*. The Institute of Electrical and Electronic Engineers, New York, EE. UU., 1998.

[33] Fogel, L. J. *On the Organization of Intellect.* PhD thesis, University of California, Los Angeles, California, EE. UU., 1964.

[34] Fogel., L. J. *Artificial Intelligence through Simulated Evolution. Forty Years of Evolutionary Programming.* John Wiley & Sons, Inc., Nueva York, EE. UU., 1999.

[35] Fonseca, C. M., and Fleming, P. J. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms* (San Mateo, California, 1993), S. Forrest, Ed., University of Illinois at Urbana Champaign, Morgan Kauffman Publishers, pp. 416–423.

[36] Fraser, A. S. Simulation of genetic systems by automatic digital computers. In *Australian J. of Biol. Sci.* (1957), vol. 10, pp. 484–491.

[37] Gass, S., and Saaty, T. The computational algorithm for the parametric objective function. In *Naval Ressearch Logistics Quarterly* (1955), vol. 2, pp. 39–45.

[38] Goldberg, D. E., and Richardson, J. Genetic algorithm with sharing for multimodal function optimization. In *Genetic Algorithms and Theri applications: Proceedings of the Second International Conference on Genetic Algorithms* (1987), I. J. J. Grenfestette, Ed., Lawrence Erlbaum, pp. 41–49.

[39] Grefenstette, J. J. Genesis : A system for using genetic search precerures. In *In Proceedings of the 1984 Conference on intelligent Systems and Machines* (1984), pp. 161–165.

[40] Holland, J. H. *Adaptation in Natural and Artificial Systems.* Ann Arbor, University Press, 1975.

[41] Horn, J., and Nafpliotis, N. Multiobjective Optimization using the Niched Pareto Genetic Algorithm. Tech. Rep. IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[42] Horn, J., and Nafpliotis., N. Multiobjective optimization using the niched pareto genetic algorithm. Tech. Rep. IlliGAl Report 93005,

University of Illinois at Urbana-Champaign, Urbana, Illinois, EE. UU., 1993.

[43] Horn, J., Nafpliotis, N., and Goldberg, D. E. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence* (Piscataway, New Jersey, June 1994), vol. 1, IEEE Press, pp. 82–87.

[44] Kalyanmoy Deb, and Ram Bhushan Agrawal. Simulated Binary Crossover for Con- tinuous Search Space. *Complex Systems 9* (1995), 115–148.

[45] Keeney, R. N., and Raiffa., H. *Decision with Multiple Objectives: Preferences and Value Tradeoffs.* John Wiley and Sons, 1976.

[46] Kennedy, J., and Eberhart, R. C. *Swarm Intelligence.* Morgan Kaufmann Publishers, California, EE. UU., 2001.

[47] Knowles, J. D., and Corne, D. W. The pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation* (Piscataway, NJ: IEEE Press, 1999), pp. 98–105.

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

[49] Knowles, J. D., Corne, D. W., and Oates, M. J. The Pareto-Envelope based Selection Algorithm for Multiobjective Optimization. In *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI)* (Berlin, September 2000), Springer, pp. 839–848.

[50] Kohonen, T. *Self-organazing maps*, 3rd. ed. Springer series in information sciences, 2001.

[51] Koza, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, EE. UU., 1992.

[52] Kuhn, H. W., and Tucker, A. Nonlinear programming. In *In Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability* (Berkeley, 1951), U. of California Press, Ed.

[53] Kumar, R., and Rockett, P. Improved sampling of the pareto-front in multiobjective genetic optimizations by steady-state evolution: a pareto converging genetic algorithm. *Evol. Comput. 10*, 3 (2002), 283–314.

[54] Laguna, M., and Martí, R. *Scatter Search: Methodology and Implementations in C*. Kluwer Academic Publishers, 2003.

[55] Landon, L. S., Haimes, Y. Y., and Wismer., D. A. On a bicriterion formulation of the problems of itegrated system identification and system optimization. In *IEEE Transactions on Systems* (1971), no. 1 in Man and Cybernetics, pp. 296–297.

[56] Lindman, H. R. Analysis of variance in complex experimental designs. *SIAM Review 18*, 1 (1976), 134–137.

[57] López Jaimes, A. Diseño de un algoritmo genético paralelo. Master's thesis, CINVESTAV-IPN, México, D.F., 2005. Director de tesis: Dr. Carlos A. Coello Coello.

[58] Maxfield, A. C. M., and Fogel, L. Artificial intelligence through a simulation of evolution. In *Biophysics and Cybernetics Systems: Proceedings of the Second Cybernetics Sciences* (Washington D.C., EE. UU., 1965), Spartan Books.

[59] Menchaca Méndez, A. Algoritmo híbrido para resolver problemas de optimización con restricciones. Master's thesis, CINVESTAV-IPN, México, D. F., Noviembre 2008. Director de tesis: Dr. Carlos A. Coello Coello.

[60] Mendel, G. J. Experiments in plant hybridisation. *Journal of Royal Horticultural Society 26*, 1–32 (1901).

[61] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*, third ed. Springer-Verlag, Nueva York, EE. UU., 1996.

[62] Michalewicz, Z., and Fogel, D. B. *How to Solve It: Modern Heuristics.* Springer, Berlin, Alemania, 2000.

[63] Miettinen, K. M. *Nonlinear Multiobjective Optimization.* Kluwer academic, Norwell, Massachusetts, EE. UU., 1999.

[64] Mäkelä, M. M. Issues of implementing a fortran subroutine packae nsolib for nonsmooth optimization. Tech. rep., University of Jyväskylä, 1993.

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

[66] Myers, R. H., and Montgomery, D. C. *Response Surface Methodology - Process and Product Optimization Using Designed Experiments.* John Wiley and Sons, 2002.

[67] Oei, C. K., Goldberg, D. E., and Chang, S.-J. Tournament selection, niching, and the preservation of diversity. Tech. Rep. 91011, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, Illinois, EE. UU., Diciembre 1991.

[68] Osyczka, A. *Multicreterion Optimization in engineering with FORTRAN programs.* Ellis Horwood Limited, 1984.

[69] Rao Singiresu, S. *Engineering Optimization: Theory and Practice*, third ed. John Wiley and Sons, 1996.

[70] Rechenberg, I. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution.* Fromman-Holzboog Verlag, Stuttgart, Alemania, 1973.

[71] Reynolds, R. G., Michalewicz, Z., and Cavaretta, M. Using cultural algorithms for constraint handling in genocop. In *In Proceedings of the Fourth Annual Conference on Evolutionary Programming* (Cambridge, Massachusetts, EE. UU., 1995), J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, Eds., MIT Press, pp. 298–305.

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

[73] Rudolph, G. Convergence of non-elitist strategies. In *Proceedings of the First IEEE International Conference on Evolutionary Computation* (1994), IEEE Press, pp. 63–66.

[74] Schaffer, J. D. *Multiple objective optimization with vector evaluated genetic algorithms*. PhD thesis, Vanderbilt University, Nashville TN, EE. UU., 1984.

[75] Schaffer, J. D. Multiple objective optimization with vector evaluated genetic algorithms. In *In Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms* (1985), L. Erlbaum, Ed., pp. 93–100.

[76] Schaffer, J. D., Caruana, R. A., Eshelman, L. J., and Das, R. A study of control parameters affecting online performance of genetic algorithms for function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms* (San Mateo, California, 1989), J. D. Schaffer, Ed., Morgan Kaufmann Publishers, pp. 51–60.

[77] Schott, J. R. Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.

[78] Schwefel, H.-P. Kybernetische evolution als strategie der experimentall forschung in der strömungstechnik. Master's thesis, Universidad Tecnica de Berlin, 1965.

[79] Schwefel, H.-P. ((adaptive mechanismen in der biologischen evolution un ihr einfluß auf die evolutionsgeschwindigkeit.)) inf. téc. del working group of bionics and evolution techniques at the institute for measurement and control technology. Tech. Rep. 215/3, Technical University of Berlin, 1974.

[80] Schwefel, H.-P. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhäuser Verlag, Basel, Alemania, 1977.

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

[82] Schwefel, H.-P. *Evolution and Optimum Seeking*. Wiley, New York, EE. UU., 1995.

[83] Serrano Hernández, V. A. Métodos para reducir evaluaciones en algoritmos evolutivos multi-objetivo, basados en aproximación de funciones. Master's thesis, CINVESTAV-IPN, México, D. F., 2007.

[84] Srinivas, N., and Deb, K. Multiobjective optimization using nondominated sorting in genetic algorithms. Tech. rep., Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India, 1993.

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

[86] Storn, R., and Price, K. Differential evolution - a simple and efficient adaptative scheme for global optimization over continuous spaces. Tech. Rep. TR-95–12, International Computer Science, Berkeley, California, March 1995.

[87] Tan, K. C., Lee, T. H., and Khor, E. F. Evolutionary altorithms with dynamic population size and local exploration for multiobjective optimization. *IEEE Transactions on Evolutionary Computation 5*, 6 (December 2001), 565–588.

[88] Toscano Pulido, G. Optimización multiobjetivo usando un micro algoritmo genético. Master's thesis, Universidad Veracruzana - LANIA, 2001. Director de tesis: Dr. Carlos A. Coello Coello.

[89] Toscano Pulido, G., and Coello Coello, C. A. The micro genetic algorithm 2: Towards online adaptation in evolutionary multiobjective optimization. *Second International Conference, EMO* (2003), 252–266.

[90] Trautmann, H., Ligges, U., Mehnen, J., and Preuss, M. A Convergence Criterion for Multiobjective Evolutionary Algorithms Based on Systematic Statistical Testing. In *Parallel Problem Solving from Nature–PPSN X*, G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, Eds. Springer. Lecture Notes in Computer Science Vol. 5199, Dortmund, Germany, September 2008, pp. 825–836.

[91] Van Veldhuizen, D. A. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

[92] Van Veldhuizen, D. A., and Lamont, G. B. Multiobjective evolutionary algorithm research: A history and analysis. Tech. Rep. TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.

[93] Vilfredo, P. *Cours D'Economie Politique*. F. Rouge, 1896.

[94] Wetzel, A. Evaluation of the effectiveness of genetic algorithms in combinatorial optimization. Tech. rep., University of Pittsburgh, 1983.

[95] Whitley, D. The genitor algorithm and selection pressure: Why rank-based of reproductive trials is best. In *Third International Conference on Genetic Algorithms*, J. D. Schaffer, Ed. Morgan Kaufmann, San Mateo, California, 1989, pp. 116–121.

[96] Wolpert, D. H., and Macready, W. G. No free lunch theorems for optimization. *IEEE Transactiones On Evolutionary Computation 1*, 1 (1997), 67–82.

[97] Yu, P. L. A Class of Solutions for Group Decision Problems. *Management Science 19*, 8 (1973), 936–946.

[98] Zadeh, L. Optimality and non-scalar-valued performance criteria. In *IEEE transactions on Automatic Control* (1963), vol. 8, pp. 59–60.

[99] Zeleny, M., and Cochrane, J. *Multiple Criteria Decision Making*. University of South California Press, Columbia, South California, EE. UU., 1973, ch. Compromise programming, pp. 262–301.

[100] Zielinski, K., and Laur, R. Adaptive Parameter Setting for a Multi-Objective Particle Swarm Optimization Algorithm. In *2007 IEEE Congress on Evolutionary Computation (CEC'2007)* (Singapore, September 2007), IEEE Press, pp. 3019–3026.

[101] Zitzler, E. *Evolutionary algorithms for multiobjective optimization: Methods and applications*. Doctoral dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999.

[102] Zitzler, E., Deb, K., and Thiele, L. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Tech. Rep. 70, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, December 1999.

[103] Zitzler, E., Laumanns, M., and Thiele, L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Tech. Rep. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.

[104] Zitzler, E., Teich, J., and Bhattacharyya, S. S. Evolutionary Algorithm Based Exploration of Software Schedules for Digital Signal Processors. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99)* (San Francisco, California, July 1999), W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., vol. 2, Morgan Kaufmann, pp. 1762–1769.

[105] Zitzler, E., and Thiele, L. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation 3*, 4 (November 1999), 257–271.

[106] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Grunert da Fonseca, V. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation 7*, 2 (April 2003), 117–132.