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

Una nueva propuesta para optimización multiobjetivo basada en búsqueda dispersa (Scatter Search)

Tesis que presenta

Noel Antonio Ramírez Santiago

Para obtener el grado de

Maestro en Ciencias

En la especialidad de

Ingeniería Eléctrica

Director de la Tesis: **Dr. Carlos A. Coello Coello**
Codirector de la Tesis: **Dr. Alfredo García Hernández-Díaz**

México, D. F.

Diciembre 2006

Dedicado a la memoria de mi padre. Gracias por enseñarme, cuidarme, amarme y preparame para afrontar la vida. Te llevo en mi corazón en cada paso que doy.

Agradecimientos

Agradezco a mi madre por su infinito apoyo en cada momento de mi vida, que con su ejemplo y educación ha sabido guiarme. Te amo y admiro Mamá.

A mi hermana que ha sido mi compañera y amiga en cada momento que hemos compartido.

A mi novia Toñita por su amor incondicional, su amistad y comprensión, por permitirme entrar en su vida. Te amo.

A mis amigos Mireya, Mario, Edna, Israel por darme su apoyo y amistad. Por hacer de estos años algo increíble, inolvidable y único.

A mis no menos queridos amigos Luis y Gregorio por facilitarme las cosas con sus conocimientos al emprender esta Tesis.

Al Dr. Carlos A. Coello Coello por compartir sus conocimientos y guiarme para concluir una meta muy importante en mi vida.

Al Dr. Alfredo García Hernández-Díaz por transmitirme sus conocimientos a lo largo de este tiempo de la tesis y sobre todo por brindarme su amistad y apoyo.

A mis sinodales, Dr. Francisco José Rodriguez Henríquez y Dr. Luis Gerardo de la Fraga, por llevar a cabo la revisión de mi tesis, y así obtener un mejor trabajo.

A Sofi que nos ofrece su cariño y nos hace sentir como en casa sin tener ninguna obligación de hacerlo.

Al CINVESTAV por permitirme pertenecer y coloborar en esta gran institución.

Al CONACYT por su apoyo económico ya que sin esta ayuda concluir esta meta habría sido muy difícil.

Agradezco la beca terminal de maestría obtenida a través del proyecto NSF-CONACyT titulado "Artificial Immune Systems for Multiobjective Optimization" (Ref. 42435-Y), cuyo responsable es el Dr. Carlos A. Coello Coello.

Resumen

En el mundo real existe una gran cantidad de problemas de optimización para los cuales los métodos de programación matemática no pueden garantizar que la solución obtenida sea óptima y, en muchos de ellos, ni siquiera pueden aplicarse. Es estos problemas las metaheurísticas se vuelven una alternativa viable. De entre las diversas metaheurísticas disponibles en la actualidad, los algoritmos evolutivos se cuentan entre los más populares debido a su simplicidad conceptual y su eficacia.

En el mundo real, los problemas de optimización multiobjetivo se presentan con mucha frecuencia. En ellos hay que optimizar dos o más funciones objetivo que se encuentran normalmente en conflicto entre sí. La solución de problemas multiobjetivo usando metaheurísticas (sobre todo, algoritmos evolutivos) es un tema que ha adquirido gran popularidad en los últimos años.

En este documento presentamos dos algoritmos evolutivos multiobjetivo híbridos, basados en *optimización mediante cúmulos de partículas* (PSO) y *búsqueda dispersa* : el primero se caracteriza por tener un mecanismo de selección basado en optimización global, el segundo algoritmo se caracteriza por contar con un mecanismo de selección basado en exploraciones de diversas regiones dentro de la población secundaria. Ambos algoritmos se hibridizan con la búsqueda dispersa, a fin de aprovechar los mecanismos de diversificación de esta metaheurística.

La motivación de esta hibridación es aprovechar las características que nos brinda cada uno de los algoritmos a combinarse. PSO ofrece una rápida convergencia al adoptar un nuevo criterio de selección de líderes propuesto en esta tesis. Por su parte la búsqueda dispersa nos brinda una mayor diversificación de soluciones para generar las partes faltantes del frente de Pareto.

Las dos propuestas realizadas son evaluadas utilizando funciones de prueba estándar. Los resultados obtenidos se comparan con respecto al NSGA-II y el SPEA2, que son algoritmos representativos del estado del arte en el área. Nuestra propuesta muestra tener una mejor convergencia y dispersión de soluciones en la mayoría de las funciones realizando tan solo 4,000 evaluaciones de la función de aptitud.

Abstract

In the real world, there is a large number of optimization problems for which mathematical programming techniques cannot guarantee that the solution obtained is optimum and, in many of them, such methods are not even applicable. It is precisely in these problems in which the use of metaheuristics is a viable choice. From the many metaheuristics currently available, evolutionary algorithms are among the most popular due to their conceptual simplicity and their efficacy.

In the real world, multiobjective optimization problems appear very often. In them, we aim to optimize two or more objective functions which are normally in conflict with each other. The solution of multiobjective optimization problems using metaheuristics (particularly, evolutionary algorithms) is a topic that has become very popular in the last few years.

In this document, we present two hybrid multiobjective evolutionary algorithms, based on *particle swarm optimization* (PSO) and *scatter search*: the first has a selection mechanism based on global optimization, the second algorithm consists of a selection mechanism based on explorations of diverse regions within the secondary population. Both algorithms are hybridized with scatter search, so the diversification mechanisms of this metaheuristic are exploited.

The motivation for this hybridization is to exploit the features that these two metaheuristics can provide when combining them. PSO offers a fast convergence when adopting a new leader selection criterion proposed in this thesis. On the other hand, scatter search provides us with a greater diversity of solutions, which are used to fill the gaps in the generated Pareto fronts.

The two proposed approaches are evaluated using standard test functions and metrics. Our results are compared with respect to the NSGA-II and SPEA2, which are algorithms representative of the state-of-the-art in the area. Our proposed approach exhibits a better convergence and spread of solutions in most of the test functions adopted, while performing only 4,000 fitness function evaluations.

Índice general

Índice de Figuras xv

Índice de Tablas xix

Índice de Algoritmos xxi

1. Introducción 1
1.1. Organización de la Tesis 2

2. Computación Evolutiva 5
2.1. Neodarwinismo 7
2.2. Principales paradigmas de la Computación Evolutiva 9
2.2.1. Programación evolutiva 11
2.2.2. Estrategias evolutivas 13
2.2.3. Algoritmos genéticos 14

3. Optimización Multiobjetivo 17
3.1. Optimización 17
3.2. Optimización global 18
3.3. Conceptos de optimización multiobjetivo 20
3.4. Clasificación de los algoritmos multiobjetivo 22
3.4.1. Métodos basados en funciones agregativas 24

3.4.2. Métodos basados en la población 24
3.4.3. Métodos basados en conceptos de Pareto 25

4. Optimización Mediante Cúmulos de Partículas 29
4.1. Modelo cultural de Axelrod 30
4.2. Proceso de aprendizaje 32
4.3. Sistemas de cúmulos 33
4.3.1. Principios de los cúmulos de partículas 33
4.3.2. Algoritmo de la optimización mediante cúmulos de partículas 35
4.3.3. Trabajos relacionados 38

5. Búsqueda Dispersa 41
5.1. Orígenes de la búsqueda dispersa 41
5.2. Esquema de la búsqueda dispersa 43
5.3. Algoritmo de la búsqueda dispersa 44
5.4. Trabajos relacionados 46

6. Descripción del Esquema Propuesto 49
6.1. Algoritmos del PSO 50
6.1.1. Mecanismo de selección basado en optimización global 50
6.1.1.1. Población secundaria 51
6.1.1.2. Selección del líder 54
6.1.1.3. Recombinación BLX-α 58
6.1.1.4. Mutación 60
6.1.1.5. Descripción del algoritmo. 61
6.1.2. Mecanismo de selección basado en exploraciones de diversas regiones dentro de la población secundaria . . . 62
6.1.2.1. Selección del líder 62
6.1.2.2. Descripción del algoritmo 65
6.2. Algoritmo de búsqueda dispersa 66
6.2.1. Conjunto de soluciones iniciales 66
6.2.2. Método para obtener soluciones dispersas 68
6.2.3. Método para actualizar el conjunto de referencia 70
6.2.4. Método para generar subconjuntos 72
6.2.5. Método de combinación 72
6.2.6. Metodo de mejora 72
6.2.7. Descripción del algoritmo 73

7. Análisis de Resultados **75**
7.1. Métricas . . . 76
7.1.1. Distancia generacional invertida (DGI) . . . 76
7.1.2. Distribución (D) . . . 76
7.1.3. Cobertura (C) . . . 77
7.2. Comparación de resultados . . . 77
7.2.1. Función de Kursawe . . . 80
7.2.2. Función ZDT1 . . . 82
7.2.3. Función ZDT2 . . . 84
7.2.4. Función ZDT3 . . . 86
7.2.5. Función ZDT4 . . . 88
7.2.6. Función ZDT6 . . . 90
7.2.7. Función DTLZ1 . . . 92
7.2.8. Función DTLZ2 . . . 94
7.2.9. Función DTLZ3 . . . 96
7.2.10. Función DTLZ4 . . . 98
7.3. Conclusiones sobre los resultados obtenidos . . . 100

8. Conclusiones y Trabajo Futuro **101**
8.1. Conclusiones . . . 101
8.2. Trabajo Futuro . . . 102

A. Funciones de prueba **105**
A.1. Función de Kursawe . . . 105
A.2. Función ZDT1 . . . 106
A.3. Función ZDT2 . . . 106
A.4. Función ZDT3 . . . 107
A.5. Función ZDT4 . . . 107
A.6. Función ZDT6 . . . 108
A.7. Función DTLZ1 . . . 108
A.8. Función DTLZ2 . . . 109
A.9. Función DTLZ3 . . . 109
A.10.Función DTLZ4 . . . 110

B. Convergencia del MOPSOSS en las Funciones de Prueba Adoptadas **111**

Bibliografía **117**

Índice de figuras

2.1. Ejemplo de una cruza de dos puntos, usando cadenas binarias. 10
2.2. Ejemplo del operador de mutación en una cadena binaria. . . 10
2.3. Máquina de estados finita. 11

3.1. Ejemplo de la dominancia de Pareto para dos funciones objetivo. 22

4.1. Topología de anillo. 38
4.2. Topología con vecinos completamente conectados. 39

5.1. Esquema de la búsqueda dispersa 43

6.1. Ilustración del concepto de dominancia y dominancia-ϵ. 52
6.2. Ilustración de dos soluciones ϵ-dominadas iguales. 54
6.3. Selección del líder utilizando una función objetivo a la vez. . . 56
6.4. Selección del líder utilizando una función agregativa no lineal. 56
6.5. Intervalos de acción del operador de recombinación BLX-α. . . 58
6.6. Selección del líder creando un vector ideal. 64
6.7. Esquema propuesto de la búsqueda dispersa. 68
6.8. Conjunto de soluciones iniciales. El conjunto P es la unión de la población secundaria y terciaria. 69
6.9. Soluciones dispersas. 70
6.10. Selección de soluciones para crear el conjunto de referencia . . 71

7.1. Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función de Kursawe 81
7.2. Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función ZDT1 . . . 83
7.3. Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función ZDT2 . . . 85
7.4. Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función ZDT3 . . . 87
7.5. Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función ZDT4 . . . 89
7.6. Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función ZDT6 . . . 91
7.7. Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función DTLZ1 . . 93
7.8. Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función DTLZ2 . . 95
7.9. Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función DTLZ3 . . 97
7.10. Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función DTLZ4 . . 99

B.1. Gráfica de convergencia del algoritmo MOPSOSS v2 en la función de Kursawe (realizando 7,000 evaluaciones de la función objetivo). 112
B.2. Gráfica de convergencia del algoritmo MOPSOSS v2 en la función ZDT1 (realizando 7,000 evaluaciones de la función objetivo). 112
B.3. Gráfica de convergencia del algoritmo MOPSOSS v2 en la función ZDT2 (realizando 7,000 evaluaciones de la función objetivo). 113
B.4. Gráfica de convergencia del algoritmo MOPSOSS v2 en la función ZDT3 (realizando 7,000 evaluaciones de la función objetivo). 113
B.5. Gráfica de convergencia del algoritmo MOPSOSS v2 en la función ZDT4 (realizando 30,000 evaluaciones de la función objetivo). 114
B.6. Gráfica de convergencia del algoritmo MOPSOSS v2 en la función ZDT6 (realizando 7,000 evaluaciones de la función objetivo). 114
B.7. Gráfica de convergencia del algoritmo MOPSOSS v2 en la función DTLZ1 (realizando 150,000 evaluaciones de la función objetivo). 115

B.8. Gráfica de convergencia del algoritmo MOPSOSS v2 en la función DTLZ2 (realizando 7,000 evaluaciones de la función objetivo). 115
B.9. Gráfica de convergencia del algoritmo MOPSOSS v2 en la función DTLZ3 (realizando 150,000 evaluaciones de la función objetivo). 116
B.10. Gráfica de convergencia del algoritmo MOPSOSS v2 en la función DTLZ4 (realizando 150,000 evaluaciones de la función objetivo). 116

Índice de tablas

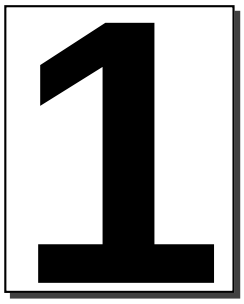
4.1. Población inicial de 100 individuos en un arreglo bi-dimensional para efectuar la simulación utilizando el paradigma de Axelrod. 31
4.2. Resultado de la simulación utilizando una función estocástica de similitud (paradigma de Axelrod). 32

7.1. Parámetros utilizados en los algoritmos para realizar las pruebas. 79
7.2. Resultados de las métricas DGI y D para la función de Kursawe 80
7.3. Resultados de la métrica de cobertura para la función de Kursawe 81
7.4. Resultados de las métricas DGI y D para la función ZDT1 . . 82
7.5. Resultados de la métrica de cobertura para la función ZDT1 . 83
7.6. Resultados de las métricas DGI y D para la función ZDT2 . . 84
7.7. Resultados de la métrica de cobertura para la función ZDT2 . 85
7.8. Resultados de las métricas DGI y D para la función ZDT3 . . 86
7.9. Resultados de la métrica de cobertura para la función ZDT3 . 86
7.10. Resultados de las métricas DGI y D para la función ZDT4 . . 88
7.11. Resultados de la métrica de cobertura para la función ZDT4 . 88
7.12. Resultados de las métricas DGI y D para la función ZDT6 . . 90
7.13. Resultados de la métrica de cobertura para la función ZDT6 . 91
7.14. Resultados de las métricas DGI y D para la función DTLZ1 . 92
7.15. Resultados de la métrica de cobertura para la función DTLZ1 93

7.16. Resultados de las métricas DGI y D para la función DTLZ2 . 94
7.17. Resultados de la métrica de cobertura para la función DTLZ2 94
7.18. Resultados de las métricas DGI y D para la función DTLZ3 . 96
7.19. Resultados de la métrica de cobertura para la función DTLZ3 96
7.20. Resultados de las métricas DGI y D para la función DTLZ4 . 98
7.21. Resultados de la métrica de cobertura para la función DTLZ4 98

Lista de Algoritmos

1. Algoritmo de la programación evolutiva 12
2. Algoritmo de la Estrategia Evolutiva $(1+1)-EE$ 14
3. Algoritmo Genético 16
4. Algoritmo para comparar dos soluciones utilizando el criterio de dominancia de Pareto. 23
5. Algoritmo del PSO 37
6. Algoritmo de la Búsqueda Dispersa 45
7. Algoritmo para agregar soluciones a la población secundaria. . . 53
8. Algoritmo de selección del líder basado en optimización global. 57
9. Algoritmo Parameter-Based Mutation. 60
10. Algoritmo del PSO basado en optimización global. 63
11. Algoritmo del PSO basado en mecanismos de exploración en diversas regiones. 67
12. Algoritmo de la búsqueda dispersa propuesto 74



Introducción

Los algoritmos evolutivos han surgido como resultado de las necesidades por resolver problemas en donde los métodos numéricos no han tenido éxito. Estos algoritmos han sido ampliamente utilizados para resolver problemas difíciles de optimización. Los algoritmos evolutivos surgieron de la abstracción biológica de la evolución siendo Darwin el precursor más importante haciendo notar que la selección natural juega un rol determinante en el proceso evolutivo.

La optimización se define como "el acto de obtener el mejor resultado posible dadas ciertas circunstancias" [69].

En aras de llevar a cabo un proceso de optimización, en este documento presentamos dos algoritmos evolutivos híbridos multiobjetivo basado en *optimización mediante cúmulos de partículas* (PSO) y *búsqueda dispersa*. PSO está inspirado en el comportamiento social de las parvadas de aves, en los bancos de peces e insectos. Este técnica fue propuesta por James Kennedy y Russell Eberhart a mediados de la década de 1990 [54]. PSO es una técnica que ha mostrado ser muy buena tanto en problemas de optimización monoobjetivo como multiobjetivo [2, 16, 54]. PSO mantiene un balance entre dos

características importantes de cualquier algoritmo evolutivo: la exploración y la explotación, debido a los atractores de la partícula que son el mejor registro personal (*pbest*) y el mejor registro global (*gbest*), que hacen referencia al componente cognitivo (aprendizaje que adquiere una partícula debido a su experiencia de vuelo) y al social (conocimiento que se adquiere de forma conjunta del cúmulo de partículas), respectivamente.

La búsqueda dispersa se basa en combinar soluciones de calidad con soluciones dispersas, a partir de un conjunto denominado *conjunto de referencia*. Dicha técnica fue propuesta en los 1970s por Fred Glover [41] y utiliza estrategias para diversificar e intensificar la búsqueda y así obtener nuevas buenas soluciones. El esquema de la búsqueda dispersa es flexible ya que permite el desarrollo de nuevas alternativas para mejorar dicho esquema.

En este documento presentamos dos algoritmos evolutivos realizando nuevas aportaciones tanto para el esquema del PSO como de la búsqueda dispersa. Consideramos que el mecanismo quizá más importante es la selección de líderes dentro del esquema del PSO. Por lo anterior llevamos a cabo dos variantes para realizar dicha selección. Adoptamos un nuevo enfoque dentro de la búsqueda dispersa debido a la flexibilidad que provee este esquema, el cual utiliza el concepto de dispersión (ó diversificación).

1.1 Organización de la Tesis

Capítulo 1. Introducción: Es este capítulo, el cual presenta la introducción al resto de la tesis.

Capítulo 2. Computación Evolutiva: Hacemos una breve revisión histórica del origen de la computación evolutiva así como de sus paradigmas principales y sus elementos más importantes del área de investigación.

Capítulo 3. Optimización Multiobjetivo: Se lista una clasificación breve de un sistema con base en sus componentes. Se presentan los conceptos más importantes de optimización global así como de optimización multiobjetivo y se menciona una clasificación de los algoritmos multiobjetivo.

Capítulo 4. Optimización Mediante Cúmulos de Partículas: Se hace una revisión del surgimiento de la optimización mediante cúmulos de partículas así como la relación que guarda con el modelo cultural descrito por Axelrod; se hace una descripción completa del algoritmo del PSO así como de sus elementos más importantes.

Capítulo 5. Búsqueda Dispersa: Se hace una descripción detallada del algoritmo de la búsqueda dispersa y sus mecanismos más importantes.

Capítulo 6. Descripción del Esquema Propuesto: Se presenta una descripción completa de los dos algoritmos propuestos (ambos basados en PSO) así como del algoritmo de búsqueda dispersa con el que éstos fueron hibridizados.

Capítulo 7. Análisis de Resultados: Se presentan los resultados y se hace un estudio comparativo entre las dos propuestas realizadas y dos algoritmos representativos del estado del arte en el área: NSGA-II y SPEA2.

Capítulo 8. Conclusiones y Trabajo Futuro: Se presentan las conclusiones a las que se llegaron con los resultados obtenidos así como las posibles líneas de investigación a seguir.

Apéndice A. Funciones de Prueba: Se describen las funciones utilizadas en las comparaciones realizadas entre los algoritmos propuestos y los del estado del arte.

Apéndice B. Convergencia del MOPSOSS en las Funciones de Prueba Adoptadas: Se presentan las gráficas de convergencia de las funciones que se emplearon en la validación de uno de los dos algoritmos propuestos.

2

Computación Evolutiva

Durante mucho tiempo la teoría más aceptada sobre el origen de las especies fue el creacionismo, de acuerdo al cual se acepta de manera dogmática que "Dios creó a todas las especies del planeta de forma separada". Adicionalmente, en el creacionismo se tiene la idea de una jerarquización de las especies en donde el hombre ocupa un rango superior, sólo por debajo de Dios.

George Louis Leclerc fue el primero en contradecir abiertamente al "creacionismo" declarando que las especies se originaron entre sí, esbozando la primera teoría evolutiva 100 años antes de que Charles Darwin popularizara sus ideas evolucionistas. Leclerc, en su trabajo *Historie Naturelle*, hacía notar la similitud entre el hombre y los simios, además de proponer la idea de un ancestro común entre estas dos especies. Leclerc creía en los cambios orgánicos pero no describió un mecanismo coherente que fuera responsable de efectuarlos.

En 1801, el zoólogo francés Jean Baptiste Lamarck publica y enfatiza la importancia de la naturaleza en los cambios de las especies explicando un mecanismo responsable de dichos cambios el cual más tarde sería denominado "Lamarckismo". Su idea radica en que un cambio en el ambiente produce

cambios en las necesidades de los organismos, lo que ocasiona que éstos cambien su comportamiento. Esto, a su vez, implica el uso o desuso de ciertos órganos corporales del individuo.

En el siglo XIX, August Weismann [85] formuló una teoría denominada el *germoplasma*, según la cual el cuerpo se divide en células germinales (o germoplasma) que pueden transmitir información hereditaría y en células somáticas (o sotoplasmas), que no pueden hacerlo. Para Weismann, la selección natural era el único mecanismo que podía cambiar el germoplasma (o *genotipo*[1]) y creía que tanto el germoplasma como el medio ambiente podían influenciar al sotoplasma (o *fenotipo*[2]).

En 1859, Charles Robert Darwin [19] publicó uno de los libros más importantes en la historia de la ciencia: *El origen de las especies*, en el cual expone que una especie que no sufriera cambios se volvería incompatible con su medio ambiente ya que éste tiende a cambiar con el transcurrir del tiempo. Además de hacer notar las similitudes entre hijos y padres observada en la naturaleza, Darwin destacó que ciertas características de las especies eran hereditarias, y que de generación a generación ocurren cambios para hacer al nuevo individuo más apto para sobrevivir.

Johann Gregor Mendel realizó una serie exhaustiva de experimentos con plantas de guisantes en los cuales descubrió tres leyes básicas en el paso de una característica de un miembro de una especie a otro:

1. *Ley de Segregación*. Establece que los miembros de cada par de alelos de un gene se separan cuando se producen los gametos durante la meiosis.

2. *Ley de la Independencia*. Establece que los pares de alelos se separan entre sí durante la formación de gametos.

3. *Ley de la Uniformidad*. Establece que cada característica heredada se determina mediante dos factores provenientes de ambos padres, lo cual decide si un cierto gene es dominante o recesivo.

[1]Denota la composición genética de un organismo.
[2]Denota los rasgos físicos de un individuo.

2.1 Neodarwinismo

Si combinamos las propuestas hechas originalmente por la teoría evolutiva de Darwin en combinación con el seleccionismo de August Weismann y la genética de Gregor Mendel, todo esto da origen al paradigma Neodarwiniano, el cual establece [46]:

"La historia de la vasta mayoría de la vida en nuestro planeta puede ser explicada a través de un puñado de procesos estocásticos que actúan sobre y dentro de las poblaciones y especies: la reproducción, la mutación, la competencia y la selección."

La reproducción. Es una proceso mediante el cual los individuos existentes engendran nuevos individuos, proceso sin el cual cualquier especie se extinguiría. Existen dos formas de reproducción: asexual [3] (o vegetativa) y sexual [4] (o generativa).

La mutación. Es una alteración o cambio en la información genética de un ser vivo lo que implica un cambio de características que se presentan espontáneamente la cual se puede transmitir o heredar a la descendencia. La unidad genética capaz de mutar es el gen [5] que forma parte del ADN. David Fogel [37] estableció que en cualquier sistema que se reproduce a sí mismo continuamente y que está en constante equilibrio, la mutación está garantizada.

La competencia. Es un tipo de relación entre varios individuos que tiene lugar en una comunidad de distintas especies para la obtención de recursos, cuando existe una demanda de un recurso común que puede ser limitado. La competencia ha sido de gran importancia en la *evolución* de las especies porque ha influido en la selección.

La selección. La selección es un mecanismo esencial en la evolución, y determina la eficacia de ciertas particularidades en algunos organismos para su supervivencia y reproducción para un cierto ambiente dado.

[3] Un único organismo tiene la capacidad de originar nuevos individuos que son copias genéticamente idénticas, es decir no existe intercambio de material genético (ADN).

[4] A partir de dos progenitores se generan individuos hijos cuyas características resultarán de la combinación del ADN de los mismos.

[5] Unidad de información hereditaria.

La *evolución* no es sino el resultado de estos procesos estocásticos fundamentales que interactúan entre sí en las poblaciones, generación tras generación.

El paradigma neodarwiniano se caracteriza por lo siguiente [64]:

1. La mutación es aleatoria con respecto a la función genética y se repite con una frecuencia razonable.

2. La mutación es la fuente primaria de diversidad, pero este efecto en los cambios de los genes es pequeño, tanto así, que juega un rol menor en la evolución.

3. Debido a las mutaciones que han ocurrido en el pasado, las poblaciones naturales contienen una cantidad suficiente de diversidad genética que no responden a prácticamente ningún tipo de selección.

4. La evolución está determinada principalmente por los cambios del medio ambiente y la selección natural. Debido a que existe suficiente diversidad genética no se requieren nuevas mutaciones en una población para evolucionar en respuesta a los cambios del medio ambiente. Es decir, no hay una relación entre la proporción de las mutaciones y la proporción de los cambios evolutivos.

5. Debido a que las mutaciones tienden a repetirse con una frecuencia razonable, la estructura genética de una población es casi siempre cercana al óptimo para un medio ambiente determinado.

6. Los cambios evolutivos en una especie ocurren gradualmente en respuesta a la selección natural. La macroevolución [6] no es otra cosa que la acumulación de los efectos de la microevolución [7] .

[6] Macroevolución se refiere a los cambios de una especie a gran escala, ocurridos eones de tiempo.

[7] Microevolución se refiere a los cambios que sufre una especie en las frecuencías genéticas en pequeña escala.

2.2 Principales paradigmas de la Computación Evolutiva

La computación evolutiva engloba una serie de técnicas inspiradas biológicamente en la teoría Neodarwiniana. En la actualidad, cada vez es más difícil distinguir las diferencias entre los tipos de técnicas evolutivas existentes, por lo que es más común hablar, de forma genérica, de algoritmos evolutivos.

La simulación de un proceso evolutivo en una computadora requiere:

1. **Codificar las estructuras que se replicarán**. La representación en los algoritmos evolutivos tienen una inspiración biológica. Básicamente, un individuo es representado por una cadena cromosómica, la cual está compuesta por genes [8]. A la codificación de la cadena cromosómica se le denomina *genotipo* mientras que la decodificación de dicha cadena es llamada *fenotipo* en analogía a la composición genética de un organismo y a los rasgos físicos del mismo respectivamente. Finalmente, el átomo de una cadena cromosómica se denomina *alelo* [9].

2. **Operaciones que afecten a los "individuos":**

   - *La cruza*, biológicamente, es un proceso complejo que se da entre parejas de cromosomas los cuales se alinean para fraccionarse en ciertas partes y luego intercambiar fragmentos entre sí. En computación evolutiva, el proceso es similar, aunque mucho más simple.

     En la figura 2.1 mostramos un ejemplo de una cruza de dos puntos usando cadenas binarias.

   - **La mutación** es un operador que es una fuente de diversidad cuando la población ha perdido esta característica. El uso de la mutación es objeto de controversia debido a que es difícil de controlarla ya que no existe una forma definitiva de determinar cuándo se debe aplicar con mayor o menor probabilidad. En los

[8] Gene es una subsección de un cromosoma que (usualmente) codifica el valor de un solo parámetro.

[9] Alelo es la realización de un gene, es decir, cada valor posible que puede adquirir una cierta posición genética.

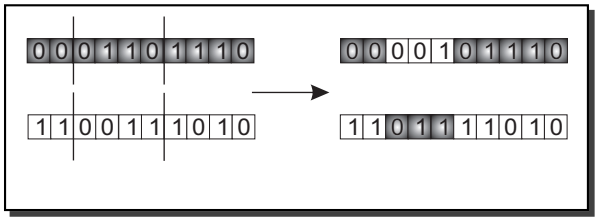


Figura 2.1: Ejemplo de una cruza de dos puntos, usando cadenas binarias.

algoritmos evolutivos, la mutación produce cambios (esporádicos) en un alelo.

En la figura 2.2 mostramos una mutación binaria. En computación evolutiva, las mutaciones se realizan con una cierta probabilidad dada por el usuario. El número de alelos que se mutan se determina con base en una probabilidad. Suele recomendarse el uso de una probabilidad de mutación de: $P_m = 1/L$, donde L es la longitud total de la cadena cromosómica.

Figura 2.2: Ejemplo del operador de mutación en una cadena binaria.

3. **Una función de aptitud**. La aptitud de un individuo se define como la probabilidad de que éste sobreviva para reproducirse o como una función del número de descendientes que éste tiene. En computación evolutiva, la función de aptitud suele ser la función a optimizarse.

4. **Un mecanismo de selección**. La selección es una parte importante de los algoritmos evolutivos haciendo analogía al medio ambiente que determina quiénes son los individuos más áptos para sobrevivir. Algunos algoritmos evolutivos aplican este mecanismo de forma probabilística mientras que otros lo hacen de manera extintiva (los individuos menos aptos desaparecen inmediatamente de la población).

La idea de aplicar los principios Darwinianos para resolver problemas data de los años treinta y precede incluso al invento de la computadora [38]. En 1948, Turing propuso "búsquedas evolutivas o genéticas ". Durante los 1960s

se implementaron tres diferentes tipos de algoritmos evolutivos, desarrollados en lugares distintos y con motivaciones distintas. En Estados Unidos, Lawrence Fogel introduce la **Programación evolutiva** [60, 39]. También en Estados Unidos, John Holland desarrolló los **Algoritmos Genéticos** [50]. Por su parte, en Alemania, Ingo Rechemberg y Hans Paul Schwefel inventaron las **Estrategias Evolutivas** [7].

A continuación describiremos brevemente cada una de estas técnicas.

2.2.1 Programación evolutiva

La Programación Evolutiva fue propuesta el 1960 por Lawrence J. Fogel [60, 39]. En ella la inteligencia se ve como un comportamiento adaptativo, enfatizándose los nexos de comportamiento entre padres e hijos.

El ejemplo clásico de la programación evolutiva es un predictor que evoluciona como una máquina de estados finitos (MEF), la cual es un transductor que puede ser simulado por un alfabeto finito o símbolos de entrada los cuales responden a un alfabeto finito de símbolos de salida. Consiste de un número de estados S y un número de transiciones entre los estados.

En la figura 2.3 mostramos una MEF que consiste de tres estados (A,B y C). Las entradas de la MEF es el alfabeto binario $E = \{0, 1\}$ y las salidas consisten del alfabeto $S = \{a, b\}$. Es decir la MEF transforma una secuencia de entradas a una secuencia de salidas.

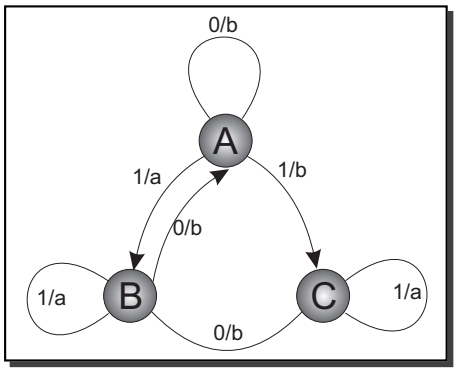


Figura 2.3: Máquina de estados finita.

La transición entre los estados es lo que define las funciones de la máquina de estados; es decir, dependiendo del estado actual y del símbolo de entrada a dicho estado se define un símbolo de salida y el próximo estado de la máquina.

En la programación evolutiva no hay lugar para la recombinación (especies distintas no se recombinan entre sí), pero un mecanismo análogo es la mutación sobre la población actual (vista como una máquina de estados) para generar descendientes. Los tipos de mutación más usados para generar una nueva máquina de estados son:

- Cambiar un símbolo de salida.
- Cambiar una transición de estados (cambiar el próximo estado).
- Añadir un estado.
- Eliminar un estado.
- Cambiar el estado inicial.

En el Algoritmo 1 podemos observar los elementos principales de la programación evolutiva, en donde la selección que se realiza es probabilística; es decir, el individuo menos apto tiene una probabilidad distinta de cero de sobrevivir.

Algoritmo 1: Algoritmo de la programación evolutiva

1 ***Datos de entrada:*** *Población* P
2 ***Resultado:*** *Población* P_{MaxGen}
3 $t \leftarrow 0$
4 $InicializarPoblacion(P_t)$
5 **repeat**
6 $\quad P'_t \leftarrow Mutar(P_t)$
7 $\quad EvaluarAptitud(P'_t)$
8 $\quad P_{t+1} \leftarrow Seleccion(P'_t \cup P_t)$
9 $\quad t \leftarrow t + 1$
10 **until** $MaxGen$

2.2.2 Estrategias evolutivas

En 1964 un grupo de estudiantes alemanes de ingeniería encabezados por Ingo Rechenberg [7] desarrollaron las estrategias evolutivas para resolver problemas hidrodinámicos de alto grado de complejidad. La versión original del algoritmo hacía uso de un solo padre a partir del cual se generaba un solo hijo, y luego el hijo y el padre compiten entre sí, sobreviviendo el más apto (a este tipo de selección se le considera "extintiva" debido a que el peor individuo tiene una probabilidad cero de sobrevivir); el algoritmo original se muestra en el Algoritmo 2 y es denominado $(1 + 1) - EE$, donde $N(0, \sigma)$ es una distribución normal con media cero y desviación estándar σ. Esto es básicamente una perturbación aplicada sobre el padre para generar al hijo.

En 1973 Rechenberg [49] introdujo el concepto de población al proponer una variante al algoritmo original denominada estrategia evolutiva $(\mu + 1) - EE$ donde μ es el número de padres usados para generar un solo hijo, el cual reemplaza al peor padre de la población. En la misma publicación propuso una regla para ajustar la desviación estándar de manera determinista durante el proceso evolutivo de tal forma que el algoritmo convergiera al óptimo. A esta regla se le conoce como "regla del éxito 1/5" y enuncia:

"La razón entre mutaciones exitosas y el total de mutaciones debe ser exactamente 1/5. Si es mayor, entonces debe incrementarse la desviación estándar. Si es menor, entonces debe decrementarse".

Más tarde, Schwefel [78, 79] modificó la estrategia para denominarla $(\mu + \lambda) - EE$ y $(\mu, \lambda) - EE$), donde λ se refiere al número de hijos producidos por los padres, y la diferencia de notación radica en la forma de selección que se realiza; en el primer caso (selección +), los μ mejores individuos obtenidos de la unión de padres e hijos sobreviven; en el segundo caso, solamente los μ mejores hijos sobreviven. En 1996, Thomas Bäck [7] derivó una regla de éxito 1/7 para las $(\mu, \lambda) - EE$.

En las estrategias evolutivas no sólo se afecta a las variables del problema, sino también a los parámetros que utiliza la estrategia (la desviación estándar). A esto se le conoce como *auto-adaptación*, y permite que el algoritmo ajuste por sí mismo sus parámetros.

En las estrategias evolutivas los tipos de recombinación pueden ser:

Algoritmo 2: Algoritmo de la Estrategia Evolutiva $(1+1)-EE$

Datos de entrada: *Población* X^0
Resultado: *Población* X^{MaxGen}
$t \leftarrow 0$
$InicializarPoblacion(\vec{X}^t)$
repeat
    $\vec{y_i}^t \leftarrow \vec{x_i}^t + N(0, \sigma)$
    if $f(\vec{y_i}^t) \leq f(\vec{x_i}^t)$ **then**
        $\vec{x_i}^{t+1} \leftarrow \vec{y_i}^t$
    else
        $\vec{x_i}^{t+1} \leftarrow \vec{x_i}^t$
    end
    $t \leftarrow t + 1$
until *MaxGen*

- *Sexuales* El operador actúa sobre dos individuos elegidos aleatoriamente de la población de padres.
- *Panmíticos* Se elige un padre aleatoriamente y se mantiene fijo. Para cada componente de sus vectores se elige un segundo padre al azar dentro de toda la población de padres.

Las estrategias evolutivas son una abstracción de la evolución a nivel de un individuo. Por esta razón, la recombinación es posible.

2.2.3 Algoritmos genéticos

Los algoritmos genéticos (AG), denominados originalmente "planes reproductivos genéticos", se concibieron a principios de los 1960s [50] por John H. Holland con la finalidad de resolver problemas de aprendizaje de máquina. Inspirado en los autómatas celulares [9] y las redes neuronales [77], Holland percibió el proceso de adaptación en términos de un formalismo en el cual los programas de una población interactúan y mejoran con base en un cierto ambiente que determina lo apropiado de su comportamiento.

Holland publicó en 1975 un libro donde describe su propuesta del "plan reproductivo genético" [50] al que hoy en día se le conoce como "algoritmo genético". Aunque en sus orígenes se pretendió popularizar para aprendizaje de máquina, actualmente es muy popular en optimización.

En los estudios realizados por Holland sobre la adaptación, éste observó [48, 47] que:

- La adaptación ocurre en un ambiente.
- La adaptación es un proceso poblacional.
- Los comportamientos individuales pueden representarse mediante programas.
- Pueden generarse nuevos comportamientos mediante variaciones aleatorias de los programas.
- Las salidas de dos programas normalmente están relacionadas si sus estructuras están relacionadas.

El Algoritmo 3 describe el mecanismo bajo el cual opera un algoritmo genético en su forma más simple. En el algoritmo genético, el operador principal es la cruza y su operador secundario es la mutación. La aptitud la determina la capacidad reproductiva de cada individuo y la población es renovada en su totalidad a cada generación.

En [73] se demuestra que el algoritmo genético requiere de elitismo (mantener intacto al mejor individuo de cada generación) para poder converger al óptimo global. Dicha demostración se realiza con base en un modelado mediante cadenas de Markov utilizando autómatas con probabilidades de transición entre cada estado, en donde cada estado es representado por una generación de la población. En [73] se concluye que los algoritmos genéticos convergen al óptimo global de la función objetivo si la probabilidad de que éste se encuentre en la población tiende a 1 cuando el número de generaciones tiende a infinito.

Un algoritmo genético requiere de los siguientes componentes básicos:

1. Una representación de las soluciones del problema (la representación tradicional es la binaria).

Algoritmo 3: Algoritmo Genético

1 ***Datos de entrada:*** *Población* P
2 ***Resultado:*** *Población* P^{MaxGen}
3 $t \leftarrow 0$
4 $InicializarPoblacion(\vec{P}^t)$
5 $EvaluarAptitud(\vec{P}^t)$
6 **repeat**
7 $\quad \vec{P_s}^t \leftarrow SeleccionarPadres(\vec{P}^t)$
8 $\quad \vec{P_c}^t \leftarrow Cruzar(\vec{P_s}^t)$
9 $\quad \vec{P_m}^t \leftarrow Mutar(\vec{P_c}^t)$
10 $\quad EvaluarAptitud(\vec{P_m}^t)$
11 $\quad \vec{P}^{t+1} \leftarrow \vec{P_m}^t$
12 $\quad t \leftarrow t + 1$
13 **until** *MaxGen*

2. Una forma de crear una población inicial de posibles soluciones (generalmente es un proceso aleatorio).

3. Una función de evaluación que realice el papel del ambiente, clasificando las soluciones en términos de su "aptitud".

4. Operadores genéticos que alteren la composición de los hijos que se producirán para las siguientes generaciones.

5. Valores para los parámetros que utiliza el algoritmo (tamaño de la población, probabilidad de cruza, probabilidad de mutación, número máximo de generaciones, etc.).

Los algoritmos evolutivos son ampliamente utilizados en la actualidad en optimización debido a su buen desempeño (tanto en lo referente a su eficacia como su eficiencia) en problemas de alta complejidad (p. ej., alta no linealidad y dimensionalidad) [34, 30].

3

Optimización Multiobjetivo

3.1 Optimización

Todo sistema tiene tres elementos básicos (entradas, modelo o proceso y salidas). Desde este punto de vista, podemos distinguir entre tres tipos de problemas:

1. *Problemas de optimización.* En los problemas de optimización, el modelo y la salida deseada son conocidas o al menos bien definidas. El problema consiste en encontrar un conjunto de entradas para lograr la o las salidas deseadas. Por ejemplo, en el problema del *agente viajero* el objetivo es visitar un conjunto de ciudades en un orden tal que se minimice un cierto costo (p. ej., la distancia total recorrida). En este problema, las entradas son las ciudades y las distancias entre ellas. La salida es un orden dado para visitarlas.

2. *Problemas de sistemas de identificación.* En estos problemas las entradas y las salidas son bien conocidas. Por ejemplo, en una red neuronal artificial se tienen las características de los objetos o personas a clasificar y se proponen salidas las cuales harán referencia a dichos objetos

o personas, pero el modelo contiene parámetros como los pesos de la red, que son lo que se desean conocer utilizando algún tipo de criterio como minimizar el error de las salidas propuestas.

3. *Problemas de simulación.* En este tipo de problemas conocemos las entradas y el modelo del sistema, y requerimos computar las salidas correspondientes a dichas entradas. Por ejemplo un filtro pasa bajas (deja pasar bajas frecuencias) en los circuitos electrónicos para filtrar señales. Nuestro modelo es un sistema complejo de fórmulas describiendo el comportamiento del circuito. Para una señal dada de entrada este modelo debe computar la señal de salida. El uso de dicho modelo (para comparar dos diseños de circuitos) es mucho más barato que construirlo y medir sus propiedades en el mundo físico.

Nosotros nos enfocaremos en los problemas de optimización, pero para abordarlos debemos primero definir el término. La definición que adoptaremos para los fines de esta tesis es la siguiente:

"*Optimización* es el acto de obtener el mejor resultado posible dadas ciertas circunstancias " [69].

3.2 Optimización global

La optimización global se ocupa de optimizar una función objetivo, bajo ciertas restricciones. Considerando un problema de minimización, las siguientes expresiones definen matemáticamente a un problema de optimización global.

Definición 1. Problema de optimización global:

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

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

$$h_i(\overrightarrow{x}) = 0; \quad i = 1, ..., n$$

Donde $\overrightarrow{x}$ es el vector de solución del problema, $f(\overrightarrow{x})$ es la función objetivo a optimizar, $g_i(\overrightarrow{x})$ son las restricciones de desigualdad y $h_i(\overrightarrow{x})$ son las restricciones de igualdad; las restricciones definen la zona factible del problema (F).

Definición 2. Óptimo global :

Una función $f(\overrightarrow{x}) \in F$ (donde F es la zona factible) posee su óptimo global en el punto $\overrightarrow{x*} \in F$ si y sólo si: $f(\overrightarrow{x*}) \leq f(\overrightarrow{x})$ para toda $\overrightarrow{x} \in F$.

Definición 3. Óptimo local :

Una función $f(\overrightarrow{x}) \in F$ posee un óptimo local (mínimo relativo) en el punto $\overrightarrow{x^l} \in F$ si y sólo si: $f(\overrightarrow{x^l}) \leq f(\overrightarrow{x})$ para todo $\overrightarrow{x}$ a una distancia ϵ de $\overrightarrow{x^l}$.

Es decir, existe un $\epsilon > 0$ tal que para todo $\overrightarrow{x}$ que satisface $|\overrightarrow{x} - \overrightarrow{x^l}| < \epsilon$, $f(\overrightarrow{x^l}) \leq f(\overrightarrow{x})$.

Se denomina *solución óptima* a $\overrightarrow{x*}$ y $f(\overrightarrow{x*})$, donde $\overrightarrow{x*}$ es el punto y $f(\overrightarrow{x*})$ es el valor del punto óptimo.

En la actualidad existen muchas técnicas para resolver diversos tipos de problemas de optimización con características específicas. Por ejemplo, para funciones lineales lo más adecuado es hacer uso del método Simplex. Para optimización no lineal hay métodos directos como la búsqueda aleatoria y métodos no directos como el método del gradiente conjugado [69]. Sin embargo, no siempre es posible aplicar dichas técnicas para resolver algunos problemas de optimización. Por ejemplo, en algunos casos se requiere información de las derivadas de la función objetivo que puede no ser posible obtener.

Existen problemas que no pueden ser resueltos utilizando un algoritmo que requiera tiempo polinomial sino exponencial. Estos problemas se caracterizan por tener un espacio de búsqueda muy grande. Es entonces cuando recurrimos a las denominadas *heurísticas*. La palabra heurística se deriva del griego *heuriskein*, que significa "encontrar" o "descubrir". El significado de

la palabra ha variado históricamente. Actualmente, el término suele usarse como adjetivo, refiriéndose a cualquier técnica que mejore el desempeño en promedio de la solución de un problema, aunque no mejore necesariamente el desempeño en el peor caso [74].

El tipo de heurística adoptada en esta tesis son los algoritmos evolutivos.

3.3 Conceptos de optimización multiobjetivo

La optimización multiobjetivo se ocupa de optimizar simultáneamente un conjunto de dos o más funciones objetivo. Dichas funciones normalmente se encuentran en conflicto unas con otras, por lo que mejorar una función implica empeorar el desempeño de las otras.

Considerando un problema de minimización, las siguientes expresiones definen matemáticamente a un problema de optimización multiobjetivo.

Definición 1. Problema de optimización multiobjetivo:

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

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

las n restricciones de igualdad:

$$h_i(\overrightarrow{x*}) = 0; i = 1, ..., n$$

y optimice el vector de funciones objetivo:

$$\overrightarrow{f}(\overrightarrow{x}) = [f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), ..., f_k(\overrightarrow{x})]$$

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

Definición 2. Óptimo de Pareto:

Vilfredo Pareto en el siglo XIX dio una nueva noción de *optimalidad* para problemas multiobjetivo:

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

El concepto anterior da pie a la obtención de un conjunto de soluciones al que se denomina conjunto de óptimos de Pareto. A las soluciones incluidas en el conjunto de óptimos de Pareto se les conoce como *no-dominados*.

Definición 3. Dominancia de Pareto:

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

El *orden parcial* es una relación binaria especial. Para algún conjunto P y una relación binaria $\leq$ en P. Entonces $\leq$ es un *orden parcial* si es *reflexiva*, *antisimétrica* y *transitiva*, es decir, para todo a, b y c en P, tenemos que:

- *Reflexividad* $a \leq a$.
- *Transitividad* si $a \leq b$ y $b \leq c$ entonces $a \leq c$.
- *Antisimetría* si $a \leq b$ y $b \leq a$ entonces $a = b$.

En otras palabras, para que una solución domine a otra, ésta debe ser estrictamente mejor en al menos un objetivo y no peor en ninguno de los otros.

Definición 4. Frente de Pareto:

Es la representación de las funciones objetivo cuyos vectores son no dominados y además están en el conjunto de óptimos de Pareto.

Para un problema multiobjetivo $\overrightarrow{f}(\vec{x})$ y un conjunto de óptimos de Pareto $P*$, el frente de Pareto ($FP*$) se define como:

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

En forma general, no es posible obtener una expresión analítica de la línea o superficie que contenga a dichos puntos. El procedimiento normal para generar el Frente de Pareto es obtener los puntos factibles Ω y su $f(\Omega)$ correspondiente. Cuando hay un número suficiente de ellos, es entonces posible determinar los puntos no dominados y producir el Frente de Pareto.

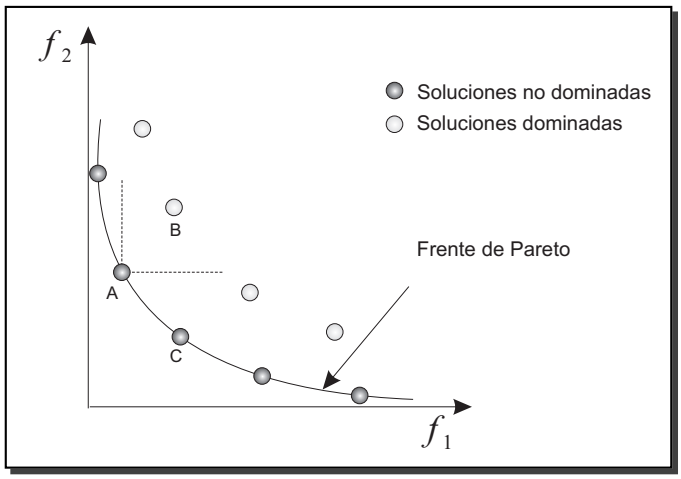


Figura 3.1: Ejemplo de la dominancia de Pareto para dos funciones objetivo.

En la figura 3.1 podemos ver gráficamente la dominancia de Pareto. Por ejemplo, la solucion A *domina* a B, ya que A mejora a la solución B tanto en la función objetivo f_1 como en la función objetivo f_2. La solución C *no domina* a A, ya que C es mejor que la solución A en f_2, pero A es mejor que C en f_1. Las soluciones *no dominadas* forman el frente de Pareto.

El procedimiento usado para evaluar la dominancia de Pareto se muestra en el Algoritmo 4.

3.4 Clasificación de los algoritmos multiobjetivo

Coello Coello [1] propone la siguiente clasificación de los algoritmos evolutivos multiobjetivo (AEMO):

- Métodos basados en funciones agregativas.
- Métodos basados en la población.
- Métodos basados en conceptos de Pareto.

Algoritmo 4: Algoritmo para comparar dos soluciones utilizando el criterio de dominancia de Pareto.

```
1  Datos de entrada: Partícula a⃗, Partícula b⃗
2  Resultado: Dominancia
3  Dominancia=EstadoInicial /*a⃗ es igual a b⃗*/
4  for i=1 to  Numero de Funciones Objetivo do
5      if f_i(a⃗) < f_i(b⃗) then
6          if Dominancia=BDominaA then
7              return ABNoDominados /*Ninguno se domina*/
8          else
9              Dominancia=ADominaB /*A Domina a B*/
10         end
11     end
12     if f_i(b⃗) < f_i(a⃗) then
13         if Dominancia=ADominaB then
14             return ABNoDominados
15         else
16             Dominancia=BDominaA /*B Domina a A*/
17         end
18     end
19 end
```

A continuación describiremos brevemente cada uno de estos tres grupos.

3.4.1 Métodos basados en funciones agregativas

La idea radica en combinar todas las funciones objetivo con operaciones aritméticas (sumas, multiplicaciones, etc.) y reducirlas a solamente una función objetivo. Dichas técnicas se denominan funciones agregativas puesto que combinan o "agregan" todas las funciones objetivo del problema en solo una función.

Un ejemplo de ello es la suma lineal de pesos que se muestra en la ecuación (3.1) donde k es el número de funciones objetivo y $w_i \geq 0$ son los pesos, cada uno de los cuales representa la importancia de cada una de las funciones objetivo.

$$\text{mín} \sum_{i=1}^{k} w_i f_i(\vec{x}) \tag{3.1}$$

Usualmente la suma de pesos debe ser igual a 1, esto es:

$$\sum_{i=1}^{k} w_i = 1 \tag{3.2}$$

Las funciones agregativas pueden ser lineales (como el ejemplo anterior) y no lineales (p. ej., las funciones agregativas adoptadas en teoría de juegos [68, 67], programación por metas [23, 86], alcance de metas [87, 88], y el algoritmo min-max [45, 12]). Estas funciones han sido subestimadas en la literatura de optimización multiobjetivo debido principalmente a las limitaciones que presentan las funciones agregativas lineales (no pueden generar porciones no convexas del frente de Pareto, independientemente de la combinación de pesos utilizada [20]). Nótese sin embargo que no sucede lo mismo con las funciones agregativas no lineales [17] (con las cuales es posible generar porciones cóncavas del frente de Pareto).

3.4.2 Métodos basados en la población

Estas técnicas se caracterizan por utilizar la población de un algoritmo evolutivo para diversificar la búsqueda. En este caso, el concepto de dominan-

cia de Pareto no es incorporado directamente dentro del proceso de selección.

El algoritmo más representativo de este tipo de métodos es el Vector Evaluated Genetic Algorithm (VEGA), propuesto por Schaffer [76]. Básicamente es un AG con una modificación en el mecanismo de selección. En cada generación se divide a la poblacion en k subpoblaciones, para un problema con k funciones objetivo. El tamaño de cada subpoblacion es de M/k donde M es el tamaño de la población. Para cada subpoblación se seleccionan los individuos de acuerdo a la *i-ésima* función objetivo (donde $i = 1, \ldots, k$). Una vez realizado el proceso de selección se crea una nueva población con los individuos seleccionados. Posteriormente se le aplican operadores genéticos (cruza y mutación).

La ventaja de este algoritmo es su sencillez. Por otro lado, VEGA presenta varios problemas, de los cuales el principal es el mecanismo de selección que es opuesto al concepto de dominancia de Pareto. Es decir, si existe un individuo que codifica una buena solución para todos los objetivos, pero no es la mejor en ninguno de ellos, dicha solución es descartada en el proceso de selección.

3.4.3 Métodos basados en conceptos de Pareto

Estos métodos cuentan con un esquema de selección que se basa en el concepto de optimalidad de Pareto. Estos métodos se subdividen en dos generaciones.

La primera generación se distingue por estar basada en el uso de la denominada jerarquización de Pareto (*Pareto ranking*) propuesta por Goldberg [44]. La idea es obtener a las soluciones no dominadas en la población y asignarles una jerarquía (la más alta). Posteriormente, se excluyen estas soluciones y se obtienen las siguientes soluciones no dominadas asignándoles una jerarquía menor, y así sucesivamente hasta jerarquizar a toda la población.

Los algoritmos más representativos de estos métodos son los siguientes: Non-dominated Sorting Genetic Algorithm (NSGA) propuesto por Srinivas y Deb [80], Niched-Pareto Genetic Algorithm (NPGA) propuesto por Horn et. al. [51] y Multiobjective Genetic Algorithm (MOGA) propuesto por Fonseca y Fleming [40].

La segunda generación se distingue por introducir el elitismo en los AEMO. En el contexto de optimización multiobjetivo, el elitismo se refiere normalmente al uso de una población externa (también denominada población secundaria) para retener a los individuos no dominados. En la introducción de dicha población surgen algunas cuestiones:

- ¿Cómo hacer interactuar la población primaria con la población secundaria?
- ¿Cómo determinar cuándo la población secundaria está llena?
- ¿Cómo determinar un criterio adicional para agregar individuos a la población secundaria en lugar de solamente usar la dominancia de Pareto?

Los algoritmos más representativos de esta generación son: Pareto Archive Evolution Strategy (PAES) introducido por Knowles y Corne [55], Niched Pareto Genetic Algorithm 2 (NPGA 2) propuesto por Erickson, Mayer y Horn [35], Pareto Envelope-based Selection Algorithm (PESA) propuesto por Corne y Joshua [18], y el Micro-Genetic Algorithm propuesto por Coello Coello y Toscano Pulido [11, 15], Strength Pareto Evolutionary Algorithm (SPEA) propuesto por Ziztler y Thiele [93].

Un algoritmo no menos importante es el Strength Pareto Evolutionary Algorithm 2 (SPEA2) propuesto por Ziztler Laumanns y Thiele [91]. Este algoritmo cuenta con varios mecanismos entre ellos se encuentra un mecanismo para calcular la aptitud de un individuo de la población basado en el conteo de los individuos que lo dominan y a los que él domina. También utiliza un mecanismo para estimar la densidad basado en la distancia de los vecinos más cercanos, cuenta con un mecanismo para truncar el número de elementos del archivo externo si éste es excedido el cual garantiza que los elementos de los extremos prevalecerán.

Se hace notar que sólo existe un algoritmo de segunda generación de uso extendido que no utiliza población secundaria: el Non-dominated Sorting Genetic Algorithm II (NSGA-II) propuesto por Deb et. al. [24, 25, 28]. El NSGA-II une la población de padres con la de hijos, y de entre ellas dos, selecciona a los mejores individuos (con base en dominancia de Pareto y un

criterio de dispersión). En este tipo de esquema de selección, el elitismo se encuentra implícito en la selección.

4

Optimización Mediante Cúmulos de Partículas

Durante mucho tiempo se ha intentado modelar el comportamiento psicológico, físico y biológico del universo y de los seres de los que se tiene conocimiento. La ciencia de la computación no podría ser la excepción y dentro de ella se encuentran los algoritmos bio-inspirados, los cuales han modelado diversos comportamientos observados en la naturaleza para llevar a cabo la resolución de problemas de optimización. Las colonias de hormigas [31, 32], algoritmos culturales [10, 13, 72], sistemas inmunes [21] son algunos ejemplos de dichos algoritmos bio-inspirados.

Pensar es una actividad social. La cultura humana y la cognición son aspectos de un proceso individual. Las personas aprenden de otras personas no sólo datos sino métodos para procesarlos. Bandura ha explicado teorías acerca del aprendizaje que ocurre cuando los individuos observan el comportamiento de otro [4]:

"Si sólo pudieramos adquirir el conocimiento a través del efecto de las acciones de uno mismo, el proceso del desarrollo del conocimiento cognitivo

y social podría ser en gran medida lento, por no decir extremadamente tedioso .... Afortunadamente, la mayor parte del comportamiento humano es aprendido a través del modelado .... La capacidad de aprender por medio de la observación posibilita ampliar sus conocimientos e ideas sobre las bases de información exhibidas y publicadas por otros."

En este capítulo abordamos las ideas principales de la abstracción de la adquisición del conocimiento y el comportamiento social que dieron origen a la optimización mediante partículas, la cual forma parte fundamental de esta tesis.

4.1 Modelo cultural de Axelrod

¿Por qué existe una gran diversidad de grupos? Por ejemplo en la política podemos observar una gran variedad de partidos políticos. En la academia algunos estudiantes se enfocan a las disciplinas sociales, algunos otros a las áreas físico-matemáticas y otros más a las disciplinas biológicas, por mencionar algunos ejemplos. Dicho comportamiento es explicado en el Modelo Cultural propuesto por Axelrod [3].

El objetivo del modelo de Axelrod es mostrar que la convergencia local no siempre implica una convergencia global de los individuos en una sociedad. La similitud entre pares de individuos pueda resultar en la diseminación de la cultura y no en la homogeneidad de la misma como intuitivamente podría esperarse.

Para ello, Axelrod construye una sociedad artificial (realiza una simulación), en donde los individuos son representados como cadenas de símbolos que representan "rasgos" (o características). El número, la longitud de la cadena y el universo de símbolos son parámetros de entrada del sistema. Por ejemplo, si el individuo contiene cinco características y esas características están definidas numéricamente entre $[0, 1 \ldots, 9]$, entonces un individuo puede ser representado como 42237 y algún otro como 99217. De esta forma se inicializa una matriz bi-dimensional de individuos en la simulación y se lleva a cabo la interacción entre ellos.

Axelrod postula que la probabilidad de la interacción humana es una fun-

ción de similitud de dos individuos: "La idea básica es que es más probable que los individuos que son similares interactúen y lleguen a ser más parecidos" [3]. En el ejemplo citado anteriormente, los dos individuos son iguales en la tercera y quinta posición por lo tanto tienen un 40 % de similitud, es decir tienen un 0.40 de probabilidades de interactuar.

Una interacción de un modelo cultural adaptativo (MCA) ocurre cuando un individuo adopta una característica de los elementos en los que difiere del otro individuo. Se selecciona un individuo adyacente ya sea horizontal o vertical de forma aleatoria. Si se satisface el criterio de similitud, el individuo seleccionado puede cambiar alguna de sus características en la misma posición de la cadena del vecino. Esto es, si los dos individuos citados anteriormente interactúan, el individuo 42237 puede tomar el valor 1 del vecino, y entonces las características que tendrá el individuo serán 42217. El elemento modificado es seleccionado aleatoriamente de los elementos en los que difiere.

La tabla 4.1 muestra una población generada aleatoriamente [54] con 100 individuos en un arreglo bi-dimensional de 10x10 para realizar la simulación. Cada individuo tiene cuatro vecinos con los cuales puede interactuar. Los principios y los finales de las filas y columnas se consideran adyacentes.

27217	74924	31157	53671	22660	37316	07959	57666	33206	92725
66219	08226	26707	45600	48767	39481	62784	89859	27792	35492
37262	66163	89178	60968	91098	19937	62103	07562	03500	13864
87746	66209	94122	72784	03593	16646	19776	87819	22160	48185
16880	09713	76057	30843	92125	41152	74156	98801	64760	00144
86287	66161	23271	46773	53014	44442	25424	98309	32553	16678
90624	65685	68785	32385	90770	24676	68806	25347	16640	30602
98681	11402	57304	68003	16943	01041	44693	63237	76040	61075
52249	30617	91425	92780	82342	30467	19721	84117	96595	55215
79949	70851	29089	89311	19176	67653	95954	64805	51332	74301

Tabla 4.1: Población inicial de 100 individuos en un arreglo bi-dimensional para efectuar la simulación utilizando el paradigma de Axelrod.

En la Tabla 4.2 se muestra el resultado al realizar la simulación de la sociedad. La simulación muestra que a medida que los individuos convergen, los grupos de individuos divergen. Y esto se da principalmente por la influencia de los vecinos. Por ejemplo, si un individuo A decide interactuar con

22233	22233	22233	22233	22233	22233	22233	22233	22233	22233
22233	22233	22233	22233	22233	22233	22233	22233	22233	22233
22233	65955	65955	22233	22233	22233	22233	35588	35588	35588
22233	22233	22233	22233	22233	22233	22233	35588	35588	35588
22233	22233	22233	22233	22233	22233	22233	22233	22233	22233
22233	22233	22233	22233	22233	22233	22233	22233	22233	22233
22233	22233	22233	22233	22233	22233	22233	13157	22233	22233
22233	22233	22233	22233	22233	22233	22233	13157	22233	22233
22233	22233	22233	22233	22233	22233	22233	22233	22233	22233
22233	22233	22233	22233	22233	22233	22233	22233	22233	22233

Tabla 4.2: Resultado de la simulación utilizando una función estocástica de similitud (paradigma de Axelrod).

otro vecino B y éste hace que A modifique alguna de sus características, al momento de interactuar con otro vecino llamémosle C, A ya habrá cambiado alguna característica que quizás tenía en común con C. Este comportamiento hace que los individuos discrepen en algunas características y que los grupos de individuos se polaricen.

4.2 Proceso de aprendizaje

El aprendizaje no sólo se da de una persona a otra. Un individuo puede adquirir el conocimiento por algunos otros medios como los libros, medios de comunicación, Internet, etc. Sin embargo el conocimiento y las ideas se transmiten de una persona a otra, por lo tanto la población converge a un proceso óptimo. Kennedy y Eberhart [54] describen un sistema que opera simulando el proceso de aprendizaje en tres niveles:

- Los individuos aprenden localmente de sus vecinos. Las personas son conscientes de la interacción con sus vecinos, recabando conocimientos de ellos y compartiendo los suyos. El aprendizaje local social es fácilmente medible y es un fenómeno bien documentado.

- La difusión del conocimiento a través del aprendizaje social resulta en un proceso emergente a nivel grupal. Este fenómeno sociológico, económico o político es visto regularmente en creencias, actitudes, comportamientos y otros atributos a través de individuos en una población.

- El proceso de conocimiento optimizado cultural. Todas las interacciones son locales, los conocimientos e innovaciones son transmitidos por la cultura desde quien lo origina hasta alguien más distante, impulsando la combinación de varias innovaciones que resultan en métodos de mejora. Este efecto global es, en gran parte, transparente para quienes interactúan en el sistema quienes son los que resultan beneficiados.

4.3 Sistemas de cúmulos

En un documento de preguntas y respuestas del Instituto Santa Fe acerca de simulación de sistemas de cúmulos se encuentra una descripción de "cúmulo" (*swarm*):

"Usamos el término *cúmulo* en el sentido general para referirnos a una colección estructurada de agentes interactuando. El ejemplo clásico es un cúmulo de abejas, pero esta metáfora puede ser extendida a otros sistemas con una arquitectura similar. Una colonia de hormigas puede ser vista como un cúmulo cuyos agentes individuales son hormigas. Una bandada de aves es un cúmulo cuyos agentes son las aves; el tráfico es un cúmulo de autos. Un sistema inmune es un cúmulo de células y moléculas. Una economía es un cúmulo de agentes económicos. Aunque la noción de cúmulo sugiere un movimiento colectivo en el espacio (como un cúmulo de aves) estamos interesados en todos los tipos de comportamiento colectivo y no sólo en un movimiento espacial."

Esta descripción de *swarm* es la más aceptada en la literatura especializada y la que utilizaremos en esta tesis.

4.3.1 Principios de los cúmulos de partículas

En [52] Kennedy propone tres principios sociocognitivos basados en el modelo cultural adaptativo de Axelrod:

- *Evaluar*. La tendencia a evaluar estimula a calificar algo como positivo o negativo, atractivo o repulsivo. Es quizás el comportamiento más ubicuo, característico de organismos vivos. Por ejemplo, las bacterias llegan a ser agitadas, corriendo y cayéndose cuando el ambiente es dañino. El

aprendizaje no puede ocurrir a menos que el organismo pueda evaluar. Es decir, puede discernir características del ambiente que atraen y características que lo rechazan. Desde este punto de vista, el aprendizaje puede ser definido como un cambio que permite al organismo mejorar el comportamiento de un individuo dentro de su ambiente.

- *Comparar*. La teoría de comparación social propuesta por Festinger [36], describe algunas de las formas que las personas usan como un estándar para poder medirse con respecto a otras, y cómo las comparaciones con respecto a otras pueden servir como un tipo de motivación para aprender y cambiar. La teoría de Festinger en su forma original no enuncia alguna manera de ser fácilmente probada o rechazada, y algunas de las predicciones generadas por la teoría no han sido confirmadas. Pero en general, ha servido como una columna vertebral para subsecuentes teorías socio-psicológicas. En casi todas las cosas que pensamos y hacemos, nos juzgamos a través de comparaciones con otros, ya sea evaluando nuestro aspecto general, riqueza, humor, inteligencia u otros aspectos de opinión y habilidad. En el Modelo Cultural Adaptativo y en los Cúmulos de Partículas, se compara a los individuos con respecto a sus vecinos con una medida concisa y los imita solamente si son mejores a ellos.

- *Imitar*. Se podría pensar que imitar podría estar en cualquier parte de la naturaleza; es como una forma efectiva de aprender a hacer cosas. Lorenz [58] ha puntualizado que algunos animales son capaces de realizar una imitación real. Él afirma que sólo los humanos y algunas aves son capaces de aprender mediante la imitación. En *The Cultural Origins of Human Cognition* [82], Michael Tomasello argumenta que algunos tipos de aprendizaje social ocurren en chimpancés, pero el verdadero aprendizaje al imitar, si es que ocurre del todo, es raro. Por ejemplo, un individuo utiliza un objeto como una herramienta. El objeto puede llamar la atención de otro individuo y el segundo individuo puede usar el mismo objeto, pero de diferente manera. La imitación verdadera se da en la sociedad humana y es el punto central de la adquisición y conservación de las habilidades mentales.

4.3.2 Algoritmo de la optimización mediante cúmulos de partículas

Los tres principios propuestos por Kennedy [52] puden ser combinados y simplificados en programas computacionales para desafiar ambientes complejos, resolviendo problemas difíciles de optimización.

La optimización mediante cúmulos de partículas (*Particle Swarm Optimization*, o PSO) es una técnica desarrollada por Russell Eberhart y James Kennedy en 1995 [53], inspirada por el comportamiento social de las bandadas de aves o bancos de peces.

PSO comparte algunas similitudes con los algoritmos de Computación Evolutiva como los Algoritmos Genéticos. El sistema es inicializado con una población de soluciones aleatorias y trata de localizar óptimos, actualizando la velocidad de las partículas en cada generación. Sin embargo, a diferencia de los Algoritmos Genéticos, PSO no tiene operadores evolutivos como la cruza o la mutación.

En PSO las soluciones (también denominadas partículas), vuelan o se mueven en el espacio de búsqueda guiadas por la partícula que ha encontrado la mejor solución hasta ese momento.

Cada partícula mantiene el camino de las coordenadas en el espacio del problema las cuales son asociadas con la mejor solución (mejor aptitud) almacenada. Dicho valor es llamado *pbest* (*personal best*). Otro valor que es de interés para el PSO es el mejor valor obtenido por una partícula entre sus vecinos; esta posición es denominada *lbest* (*local best*). Cuando una partícula es considerada por toda la población topológicamente como vecino, entonces es el mejor valor global y es llamado *gbest* (*global best*).

El algoritmo original de PSO fue realizado simulando un modelo social simple. El PSO fue diseñado para simular la búsqueda de comida. El ave encuentra comida a través de la cooperación social con otras aves que se encuentran alrededor de él (con sus vecinos). El algoritmo original usa la siguiente expresión para determinar la velocidad de una partícula:

$$v_{id} = v_{id} + c_1 \cdot U(0,1) \cdot (p_{id} - x_{id}) + c_2 \cdot U(0,1) \cdot (p_{gd} - x_{id}) \tag{4.1}$$

$$x_{id} = x_{id} + v_{id} \tag{4.2}$$

donde c_1 y c_2 son constantes positivas y $U(0,1)$ es una función aleatoria uniforme en el rango $[0,1]$; $\vec{x_i} = (x_{i1}, x_{i2}, ..., x_{iDV})$ representa la i-ésima partícula, DV es el número de variables que tiene el problema; $\vec{p_i} = (p_{i1}, p_{i2}, ..., p_{iDV})$ representa la mejor posición previa de la partícula; el símbolo g representa el índice de la mejor partícula de la población; $\vec{v_i} = (v_{i1}, v_{i2}, ..., v_{iDV})$ representa la razón de cambio (velocidad) de la posición de la partícula i.

La ecuaciones (4.1) y (4.2) describen la trayectoria de vuelo de una población de partículas. La ecuación (4.1) describe cómo la velocidad es dinámicamente actualizada. La ecuación (4.2) actualiza la posición de "vuelo" de la partícula. La ecuación (4.1) consiste de tres partes. La primera parte es el "momento" (la cantidad de movimiento). La velocidad no puede ser cambiada abruptamente, sino que es cambiada a partir de la velocidad actual. La segunda parte es la parte "cognitiva" la cual representa la parte pensante que aprende a partir de la experiencia que posee del vuelo. La tercera parte es la "social" que representa la colaboración entre las partículas, y que aprende de la experiencia de vuelo del grupo.

El algoritmo 5 muestra los mecanismos principales del PSO, donde $MaxIter$ es el número de iteraciones que efectuará el algoritmo, $NumPart$ es el número de partículas, $G(\vec{x})$ es la función que evalúa la aptitud de la partícula y P es la mejor solución encontrada por la partícula hasta el momento.

Las topologías comúnmente usadas son la versión global y la versión local del PSO. En la figura 4.2 podemos observar un ejemplo de una topología global en donde se consideran como vecinos a toda la población. En la versión global, cada partícula vuela a través del espacio de búsqueda con una velocidad que es dinámicamente ajustada de acuerdo a la mejor velocidad alcanzada hasta el momento por todas las partículas.

En la figura 4.1 se muestra un ejemplo de una topología local en donde cada partícula tiene solamente dos vecinos.

Algoritmo 5: Algoritmo del PSO

Datos de entrada: *Una lista de partículas* $\vec{x}$
Resultado: *Una lista de partículas* $\vec{p}$
Inicializar una población de partículas con posiciones y velocidades aleatorias en la dimensión D del espacio del problema
repeat
  for $i = 1$ *to* $NumPart$ **do**
    if $G(\vec{x_i}) > G(\vec{p_i})$ **then**
      for $d = 1$ *to* NV **do**
        $p_{id} = x_{id}$;
      end
    end
    $g = i$; *//Valor arbitrario*
    for $j =$*indices_de_los_vecinos* **do**
      if $G(\vec{p_j}) > G(\vec{p_g})$ **then**
        $g = j$; *//g es el índice del mejor individuo en el //vecindario*
      end
    end
    for $d = 1$ *to* NV **do**
      $v_{id} = v_{id} + c_1 \cdot rand() \cdot (p_{id} - x_{id}) + c_2 \cdot Rand() \cdot (p_{gd} - x_{id})$
      $x_{id} = x_{id} + v_{id}$
    end
  end
until $MaxIter$

En la versión local del PSO la velocidad de cada partícula se actualiza de acuerdo al pbest y a la mejor partícula obtenida hasta el momento dentro del vecindario. Generalmente el vecindario de cada partícula está definido como la partícula topológicamente más cercana a la partícula de cada lado. La versión global del PSO puede ser considerada una versión local del PSO siendo el vecindario de cada partícula toda la población.

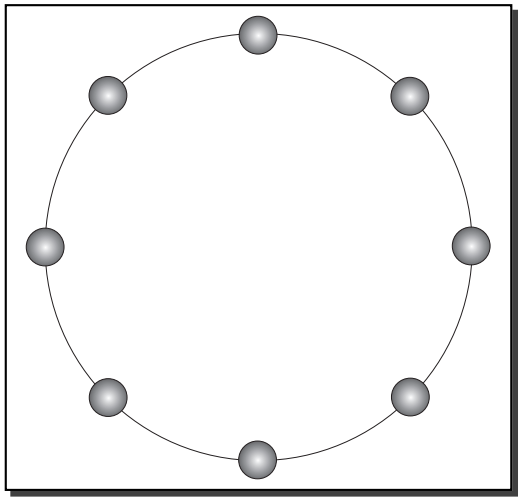


Figura 4.1: Topología de anillo.

PSO ha sido aplicado satisfactoriamente en varias áreas de investigación y aplicaciones. Existe mucha evidencia empírica de la eficiencia y efectividad del PSO en una amplia gama de problemas de optimización [54].

4.3.3 Trabajos relacionados

A continuación presentamos las propuestas más representativas realizadas a la fecha con PSO para resolver problemas de optimización multiobjetivo, los cuales son los problemas de nuestro interés.

- MOPSO propuesto por Benitez [2]. Propone tres métodos para seleccionar al líder basado en dominancia de Pareto a partir de un archivo externo. El primer método promueve la diversidad, el segundo método promueve convergencia, y el tercero (basado en probabilidades) promueve tanto diversidad como convergencia.

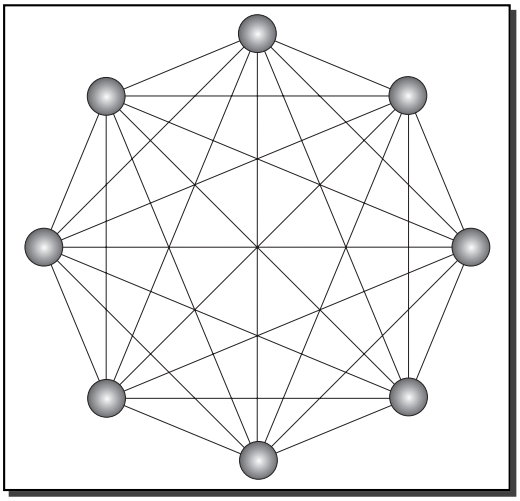


Figura 4.2: Topología con vecinos completamente conectados.

- AWPSO propuesto por Mahfouf [59]. Propone un mecanismo para modificar la velocidad, la cual se incrementa con el número de iteraciones. Utiliza una función agregativa de pesos para seleccionar la partícula personal como la partícula global. Realiza un ordenamiento de las soluciones no-dominadas para seleccionar las partículas de una iteración a otra.

- POPSO propuesto por Baumgartner [6]. La propuesta hace uso de un función agregativa lineal para resolver un problema de optimización multiobjetivo. El cúmulo de partículas se divide en n subcúmulos de igual tamaño. Cada subcúmulo utiliza un conjunto de pesos diferentes hacia la dirección al líder del subcúmulo.

- DPSO propuesto por Bartz-Beielstein [5]. Introduce elitismo dentro del PSO y propone un mecanismo para seleccionar y eliminar las partículas del archivo con base en una aptitud tanto de selección como de eliminación. La selección se realiza con base en la influencia de la partícula para generar diversidad. La aptitud de eliminación se asigna sólo si el tamaño del archivo sobrepasa al máximo valor permisible.

- PS-EA propuesto por Srinivas [81]. Proponen un híbrido combinando conceptos de estrategias evolutivas y PSO. PS-EA reemplaza la ecuación del PSO con un mecanismo de auto-adaptación, utilizando un

árbol de probabilidades y ajustes dinámicos basados en la velocidad de convergencia de la partícula.

- MOPSO propuesto por Coello y Lechuga [14]. Se basa en la idea de tener un repositorio en el cual cada partícula almacena su experiencia de vuelo después de cada iteración. El repositorio es utilizado para obtener un líder que guíe la búsqueda. Utiliza un operador de mutación sobre la partícula y sobre los rangos de las variables.

- OMOPSO propuesto por Reyes y Coello [71]. Proponen un algoritmo basado en conceptos de Pareto y utilizan un estimador de densidad sobre el vecino más cercano para la selección del líder. Hacen uso de dos archivos externos, uno para almacenar los líderes utilizados en la iteración actual y medir el rendimiento de los mismos al efectuar el vuelo y el segundo archivo es utilizado para almacenar las soluciones finales.

- OMOPSO con herencia propuesto por Reyes y Coello [70]. En esta propuesta se incorpora el concepto de aptitud heredada dentro de un MOPSO. Se Calcula un factor de aptitud con base en una distancia euclidiana entre la partícula actual, la partícula previa y la partícula líder. Posteriormente, se hereda la aptitud a la partícula nueva con el factor calculado previamente. El objetivo de este enfoque es reducir el número total de evaluaciones realizadas por el algoritmo.

5

Búsqueda Dispersa

5.1 Orígenes de la búsqueda dispersa

La búsqueda dispersa (o *scatter search*) (BD) se originó en los 1960s, y se basa en la combinación de reglas de decisión, principalmente para problemas de secuenciación, así como en la combinación de restricciones. La premisa de BD consiste en que se pueden obtener nuevas y mejores soluciones a partir de combinar soluciones de buena calidad con soluciones que poseen diversidad.

La BD fue propuesta por primera vez por Fred Glover en 1977 [41] en una conferencia celebrada en Austin, Texas y fue presentada como una heurística para programación entera. Glover describe a la BD como un método que utiliza una sucesión de coordenadas iniciales para generar nuevas soluciones. En la propuesta original, las soluciones son predeterminadas (no son aleatorias) a fin de garantizar que se cubre ampliamente el espacio de búsqueda. La BD enfoca la exploración de forma sistemática relativa a un conjunto de soluciones de referencia.

La propuesta original tiene las siguientes características [41]:

- Combina dos o más soluciones de un conjunto denominado *conjunto de referencia* con el objetivo de generar centroides.
- Genera soluciones en la línea que une a dos soluciones dadas.
- Al combinar soluciones se deben seleccionar pesos adecuados, en vez de tomarlos al azar.
- Se deben realizar combinaciones tanto convexas como no convexas de las soluciones.
- La distribución de soluciones en un requerimiento del método, por lo que los puntos deben de estar dispersos.

Dicha propuesta no fue retomada sino hasta los 1990s. En 1994, Glover [42] propone el uso de BD para problemas de optimización con espacios de búsqueda continuos, problemas combinatorios y problemas binarios. Glover hibridiza conceptos de BD con la búsqueda tabú (o Tabu Search) como un mecanismo para dirigir la búsqueda a regiones no exploradas del espacio de búsqueda. En esta propuesta, la BD se describe como un método que genera sistemáticamente un conjunto de puntos dispersos a partir de la elección de puntos de un conjunto de referencia. Asimismo se introduce el concepto de combinación ponderada como el mecanismo principal para generar nuevas soluciones. Se enfatizan las búsquedas lineales entre dos soluciones y se utilizan pesos para muestrear sobre dicha línea. En esta descripción se propone la combinación de soluciones de calidad y de soluciones con diversidad. El método incluye un componente de intensificación que consiste en realizar un mayor muestreo sobre la línea que ha producido mejores soluciones.

En los problemas binarios como el de la mochila (*knapsack*), que consiste en llenar un saco de una capacidad fija con la mayor cantidad posible de objetos valiosos, se propone un mecanismo de combinación basado en votos en donde se definen reglas para que una solución "vote" para que sus características aparezcan en la solución que se esté construyendo.

En 1998, Glover [43] publica un esquema de la BD en el cual se recopilan y simplifican las ideas básicas expuestas en trabajos anteriores. Este documento tuvo una gran difusión y actualmente es una referencia estándar de la búsqueda dispersa. En la siguiente sección ilustraremos el esquema de la búsqueda dispersa propuesta por Glover [43] en este artículo.

5.2 Esquema de la búsqueda dispersa

El esquema de la búsqueda dispersa que ha sido adoptado como estándar lo podemos observar en la figura 5.1, y consiste de los siguientes elementos:

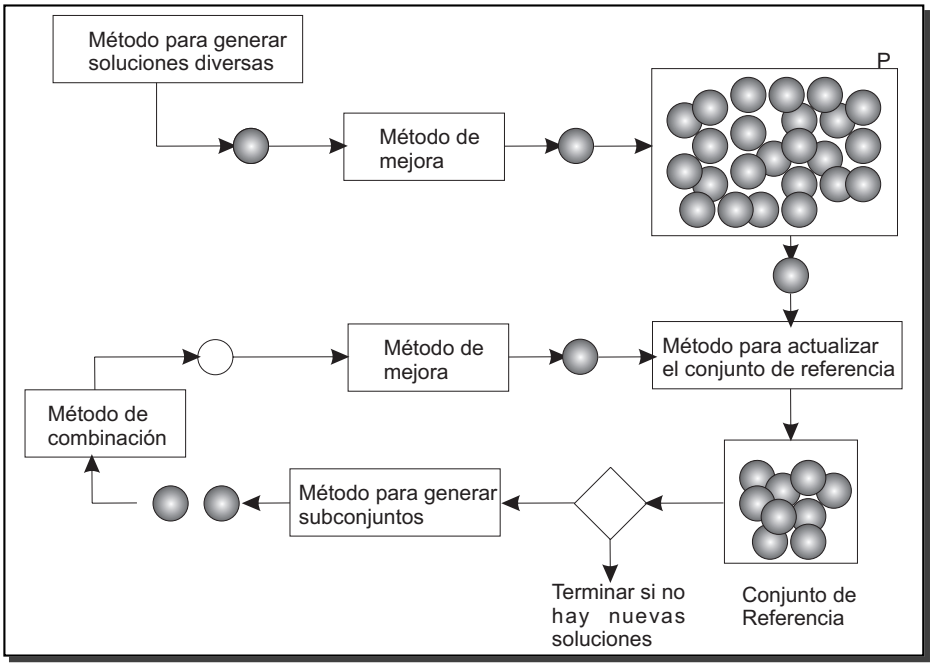


Figura 5.1: Esquema de la búsqueda dispersa

1. Método para generar soluciones diversas. El método se basa en generar un conjunto, denominado P, de soluciones diversas (aproximadamente 100 soluciones), del cual obtenemos alrededor de 10 soluciones para conformar el conjunto de referencia.

2. Método de mejora. Básicamente es un método de búsqueda local para mejorar las soluciones, tanto del conjunto de referencia como de las combinadas.

3. Método para crear y actualizar el conjunto de referencia. A partir del conjunto de soluciones diversas P se extrae el conjunto de referencia según el criterio de contener soluciones de calidad y soluciones diversas.

En cuanto a la actualización del conjunto, las soluciones fruto de la combinaciones entre las soluciones del conjunto de referencia pueden entrar y reemplazar a alguna solución en caso de que las mejore.

4. Método para generar subconjuntos. Este método especifica la forma en que se seleccionan los subconjuntos para aplicarles el método de combinación.

5. Método de combinación. La BD se basa en combinar todas las soluciones del conjunto de referencia. La solución o soluciones que se obtienen en esta combinación pueden ser introducidas inmediatamente al conjunto de referencia (actualización dinámica) o almacenarlas y esperar a combinar todas las soluciones (actualización estática).

El esquema de la BD es muy flexible, de tal forma que cada elemento puede ser implementado en una gran variedad de formas diferentes.

5.3 Algoritmo de la búsqueda dispersa

El algoritmo de la BD consiste de un método para generar soluciones diversas y se utiliza para inicializar la búsqueda generando un conjunto al que denominamos P. Básicamente lo que se hace en este método es subdividir el rango de cada variable para tener una muestra de soluciones que estén lo mejor distribuidas posibles.

El método de mejora, como su nombre lo indica, intentará mejorar las soluciones que se tengan como entrada a dicho método. El método más utilizado es el propuesto por Nelder y Mead el cual se conoce como *simplex* [65]. Éste es un método determinista de optimización no lineal, que no requiere del gradiente de la función objetivo y que suele ser muy efectivo. Sin embargo, a pesar de ello, al igual que las demás técnicas de programación matemática, el método de Nelder-Mead puede quedar atrapado en óptimos locales con cierta facilidad.

Para construir el conjunto de referencia debemos incluir soluciones tanto de diversidad como de calidad las cuales son utilizadas para generar nuevas soluciones. Primero escogemos las mejores b_1 soluciones del conjunto P. Después utilizamos un criterio de selección de mínimos y máximos para obtener

Algoritmo 6: Algoritmo de la Búsqueda Dispersa

Datos de entrada: *Una lista de soluciones* P
Resultado: *Una lista de soluciones* $RefSet$
Hacer $P = \phi$. *Utilizar el método para generar soluciones diversas para construir una solución y el método de mejora para intentar mejorarla; sea* x *la solución obtenida. Si* $x \notin P$ *entonces añadir* x *a* P*; en otro caso, rechazar* x. *Repetir esta etapa hasta que* P *tenga un tamaño predefinido.*
Construir el conjunto de referencia $RefSet = \left\{x^1, \ldots, x^b\right\}$ *con las* b_1 *mejores soluciones de* P *y las* b_2 *soluciones de* P *más diversas a las ya incluidas.*
Evaluar las soluciones en $RefSet$ *y ordenarlas de mejor a peor con respecto a la función objetivo. Denotamos por* x^b *a la mejor de todas ellas.*
Hacer $NuevaSolucion = TRUE$
while $NuevaSolucion$ **do**
    Generar $NuevosSubconjuntos$ *con el método para generar subconjuntos*
    Hacer $NuevaSolucion = FALSE$
    while $NuevosSubconjuntos \neq \phi$ **do**
        Seleccionar el siguiente subconjunto s *tal que* $s \in NuevosSubconjuntos$
        Aplicar el método de combinación a s *para obtener una o más soluciones*
        Aplicar el método de mejora a cada solución obtenida por combinación. Sea x *la solución mejorada:*
        if $f(x) < f(x^b)$ *y* $x \notin RefSet$ **then**
            Hacer $x^b = x$ *y reordenar* $RefSet$
            Hacer $NuevaSolucion = TRUE$
        end
        Eliminar s *de* $NuevosSubconjuntos$
    end
end

a las mejores soluciones de diversidad. Almacenamos los valores mínimos de las distancias euclidianas desde el conjunto P al conjunto de referencia. Después seleccionamos la solución con el valor máximo considerando a todos los valores mínimos a fin de conformar el conjunto de referencia. Este proceso es efectuado b_2 veces para obtener el tamaño deseado del conjunto de referencia ($b = b_1 + b_2$).

5.4 Trabajos relacionados

La búsqueda dispersa, a pesar de haber tenido sus orígenes en los 1970s, ha tenido un gran auge en la última década, en la que, como se ha comentado con anterioridad, se ha aplicado a problemas binarios y combinatorios principalmente. En los últimos años se ha intensificado también su uso en problemas con espacios continuos y, más recientemente, en problemas de optimización multiobjetivo.

A continuación revisamos brevemente los principales trabajos reportados en la literatura especializada que han utilizado el esquema de búsqueda dispersa para el tipo de problemas que nos atañe (problemas de optimización multiobjetivo).

- SSPMO propuesto por Molina, Laguna y Martí [61]. Este método contiene dos fases. La primera fase consiste en generar un conjunto inicial de puntos denominados "puntos eficientes" y en ella se aplica la primera fase del MOAMP (Multiobjective Metaheuristic using an Adaptive Memory Procedure) que consiste en enlazar $(p + 1)$ funciones objetivo mediante una búsqueda tabú, donde cada objetivo es seleccionado en turnos y en donde cada búsqueda inicia con el último punto de la búsqueda previa. La segunda fase consiste en aplicar el esquema de la búsqueda dispersa el cual se basa en crear el conjunto de referencia seleccionando la mejor solución para cada uno de los k objetivos del conjunto inicial denominado P para cada una de las funciones objetivo. El resto del conjunto de referencia se obtiene seleccionando las soluciones en P que maximicen la distancia a las soluciones existentes en el conjunto de referencia. Una vez creado dicho conjunto, se aplica una combinación lineal para mejorar las soluciones.

- MOSS propuesto por Ricardo Beausoleil [8]. En este algoritmo se propone hibridizar la búsqueda dispersa con la búsqueda tabú. La búsqueda tabú es utilizada durante el método de generación de diversidad y en la fase de actualización del conjunto de referencia dentro del esquema de la búsqueda dispersa.

- SSMO propuesto por Nebro, Luna y Alba [62]. Este es un algoritmo en donde se incorporan conceptos típicos del dominio de optimización multiobjetivo como la dominancia de Pareto, distancia de agrupamiento (la distancia de agrupamiento de una solución x es el promedio de las diferencias de los valores de las funciones objetivo entre dos soluciones adyacentes a la solución x. La población del algoritmo se ordena de acuerdo a cada objetivo para encontrar a las soluciones adyacentes, donde las soluciones límite tienes valores infinitos) y jerarquización de Pareto (a las soluciones no dominadas se les asigna la mayor jerarquía. Después ese conjunto se excluye obteniéndose nuevamente a las soluciones no dominadas de la población restante a las cuales se les asigna una jerarquía menor y así sucesivamente). Dichos conceptos son aplicados en el método de mejora y en el método para actualizar el conjunto de referencia dentro del esquema de la búsqueda dispersa. Para el método de mejora se utiliza una mutación y una prueba de dominancia de Pareto con el objetivo de obtener al mejor individuo. Para actualizar y construir el conjunto de referencia se analizan dos estrategias. La primera estrategia hace uso de los conceptos de jerarquización y distancia de agrupamiento para ordenar a la población y de esta forma obtener al mejor individuo. La segunda estrategia utiliza una técnica de clustering la cual consiste en obtener p centroides del conjunto compuesto por los individuos con mejor jerarquía.

- abYSS propuesto por Nebro, Luna, Dorronsoro, Alba y Beham [63]. El algoritmo utiliza conceptos del área de optimización multiobjetivo tales como la dominancia de Pareto, población secundaria, distancia de agrupamiento y un estimador de densidad.

  Para actualizar el conjunto de referencia se realiza una prueba de dominancia de Pareto con todos los elementos dentro del conjunto de referencia y se remueven del conjunto a aquellas soluciones que son dominadas por la nueva solución. Si la nueva solución domina a alguna otra, entonces es añadida al conjunto de referencia.

La distancia de agrupamiento le permite al algoritmo calcular la densidad de dispersión de las soluciones que se encuentran en la población secundaria para cada función objetivo.

6

Descripción del Esquema Propuesto

En este capítulo se expone una descripción completa de la propuesta de esta tesis. Para abordarla nos basamos en dos objetivos:

1. Realizar un algoritmo que pueda aproximarse al frente de Pareto verdadero efectuando un número relativamente bajo de evaluaciones de la función de aptitud.

2. Realizar una metaheurística utilizando búsqueda dispersa, la cual pueda competir (en cuanto a calidad de las soluciones obtenidas) con los algoritmos representativos del estado del arte en el área.

Partiendo de los objetivos señalados anteriormente, abordamos el algoritmo del PSO como una técnica que permite aproximar a las soluciones al frente de Pareto verdadero, ya que existe mucha evidencia empírica de la eficiencia y efectividad del PSO en una amplia gama de problemas de optimización [54].

Este capítulo se divide en dos partes. En la primera, se describe al algoritmo del PSO. En la segunda, se proporciona una descripción de la búsqueda dispersa. Para el algoritmo del PSO se presentan dos diferentes variantes.

6.1 Algoritmos del PSO

La ecuación de vuelo del algoritmo del PSO hasta el momento se compone de dos elementos importantes: el mejor desempeño personal de la partícula (*pbest*) y el mejor desempeño, ya sea dentro del vecindario (*lbest*) o en toda la población (*gbest*), dependiendo de la topología adoptada.

En el campo de optimización global resulta simple seleccionar a dichas partículas debido a que podemos comparar dos soluciones y determinar cuál es la mejor, dado que la aptitud de cada partícula está dada por la única función objetivo que se tiene.

Sin embargo, en el campo de optimización multiobjetivo no es tan sencillo determinar la aptitud de una partícula porque se tiene más de una función a optimizar. Sin embargo, se cuenta con el concepto de dominancia de Pareto, el cual nos permite la comparación entre soluciones.

En la siguiente sección abordamos un enfoque para optimización global para resolver problemas de optimización multiobjetivo, y en la sección 6.1.2 se presentará una propuesta basada en explorar diversas regiones del frente de Pareto obtenido a cada iteración y actualizando la partícula personal con base en una prueba de dominancia de Pareto.

6.1.1 Mecanismo de selección basado en optimización global

El algoritmo del PSO fue originalmente propuesto para optimización global. En esta sección proponemos una extensión del algoritmo original para problemas de optimización multiobjetivo.

6.1.1.1 Población secundaria

La población secundaria en este contexto se refiere a la población que contiene a las soluciones no dominadas utilizando los conceptos de Pareto definidos en la sección 3.3. Para fines de nuestro algoritmo, utilizar una población muy grande representa un costo computacional alto, el cual se puede disminuir utilizando un concepto propuesto por primera vez en [57] y denominado *dominancia-ϵ*. Con la dominancia-ϵ se busca controlar la cantidad de soluciones no dominadas que se generarán, así como garantizar su buena distribución.

Presuponiendo un problema de minimización, la siguiente expresión define el concepto de *Dominancia-ϵ*.

Definición. Dominancia-ϵ:

Sea $\vec{a}, \vec{b} \in \mathbf{Y}$ (donde $\mathbf{Y}$ es el conjunto de soluciones no dominadas). Se dice que $\vec{a}$ ϵ-domina a $\vec{b}$ para algún $\epsilon > 0$, denotado como $\vec{a} \prec_\epsilon \vec{b}$, si

$$(1-\epsilon)\cdot a_i \leq b_i \quad \forall i \in \{1,\ldots,k\} \tag{6.1}$$

En otras palabras, la *dominancia-ϵ* consiste en asignar a una solución una posición en el espacio $n-dimensional$, donde n es el número de funciones objetivo que contenga el problema a optimizar. La ecuación 6.2, asigna dicho espacio a una solución. Esta expresión ha sido utilizada, por ejemplo, en el algoritmo $\epsilon - MOEA$ [27] y en el ϵ-MyDE [75].

$$f_i^\epsilon(\vec{x}) = \begin{cases} \lfloor (f_i(\vec{x}) - f_i^{min})/\epsilon_i \rfloor & \text{para minimización} \\ \lceil (f_i(\vec{x}) - f_i^{min})/\epsilon_i \rceil & \text{para maximización} \end{cases} \tag{6.2}$$

En la figura 6.1 observamos la diferencia entre la dominancia clásica de Pareto y la dominancia-ϵ. Esta última tiene la capacidad de dominar soluciones que se encuentren en el frente de Pareto verdadero con un cierto valor de ϵ.

El Algoritmo 7 describe la forma en la que se agregan o eliminan soluciones a la población secundaria. En la línea 5 realizamos una prueba de dominancia, la cual se efectúa como se describió en el Algoritmo 4, con la diferencia que en lugar de comparar con $f_i(\vec{x})$ comparamos con respecto a $f_i^\epsilon(\vec{x})$. En la línea 12 del algoritmo 7, si la solución X es igual a P_i significa

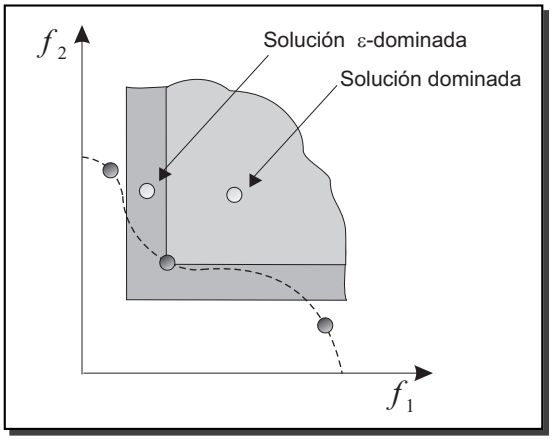


Figura 6.1: Ilustración del concepto de dominancia y dominancia-ϵ.

que se encuentran en el mismo hipercubo, como se observa en la figura 6.2, en donde la solución A se encuentra en el mismo hipercubo que la solución B. Si esta condición se cumple, se realiza una segunda prueba de dominancia pero ahora utilizando a $f_i(\vec{x})$. Si las soluciones comparadas son no dominadas se recurre a un criterio adicional. Se calcula la esquina del hipercubo con la ecuación 6.3. En las líneas 23 y 24 del algoritmo 7 se calculan las distancia euclidianas de las soluciones $\vec{X}$ y $\vec{P}_i$ a la esquina obtenida. Para ello hacemos uso de la ecuación 6.4. La solución más cercana a la esquina del hipercubo es mantenida en la población secundaria DE.

$$C_i = f_i^{\epsilon}(\vec{x}) \cdot \epsilon_i \tag{6.3}$$

$$\mathrm{DE}(x_i^{(1)}, x_i^{(2)}) = \sqrt{\sum_{i=1}^{M} (x_i^{(1)} - x_i^{(2)})^2} \tag{6.4}$$

Algoritmo 7: Algoritmo para agregar soluciones a la población secundaria.

Datos de entrada: *Partícula* $\vec{X}$
Resultado: *Una lista de soluciones* $\vec{P}$
for *Cada partícula* $\vec{P}_i \in$ *población secundaria* **do**
    $PruebaDominancia\text{-}\epsilon(\vec{X}, \vec{P}_i)$
    if $\vec{X}$ *domina a* $\vec{P}_i$ **then**
        $\vec{P}_i \leftarrow Dominada$
    else
        if $\vec{P}_i$ *domina*$-\epsilon$ *a* $\vec{X}$ **then**
            $Terminar$
        else
            if $\vec{X} = \vec{P}_i$ **then**
                $PruebaDominancia(\vec{X}, \vec{P}_i)$
                if $\vec{X}$ *domina a* $\vec{P}_i$ **then**
                    $\vec{P}_i \leftarrow Dominada$
                else
                    if $\vec{P}_i$ *domina a* $\vec{X}$ || $\vec{P}_i = \vec{X}$ **then**
                        $Terminar$
                    else
                        /*$NumFx$ es el número de funciones objetivo.*/
                        for $i \leftarrow 1$ **to** $NumFx$ **do**
                            $Esquina[i] \leftarrow f_i^{\epsilon}(\vec{X}) \cdot \epsilon_i$
                        end
                        $d1 \leftarrow DE(\vec{X}, \vec{Esquina})$
                        $d2 \leftarrow DE(\vec{P}_i, \vec{Esquina})$
                        if $d1 < d2$ **then**
                            $\vec{P}_i \leftarrow Dominada$
                        else
                            $Terminar$
                        end
                    end
                end
            end
        end
    end
end
Remover a las partículas de la población secundaria marcadas como dominadas
Agregar a la partícula $\vec{X}$ *dentro de la población secundaria*

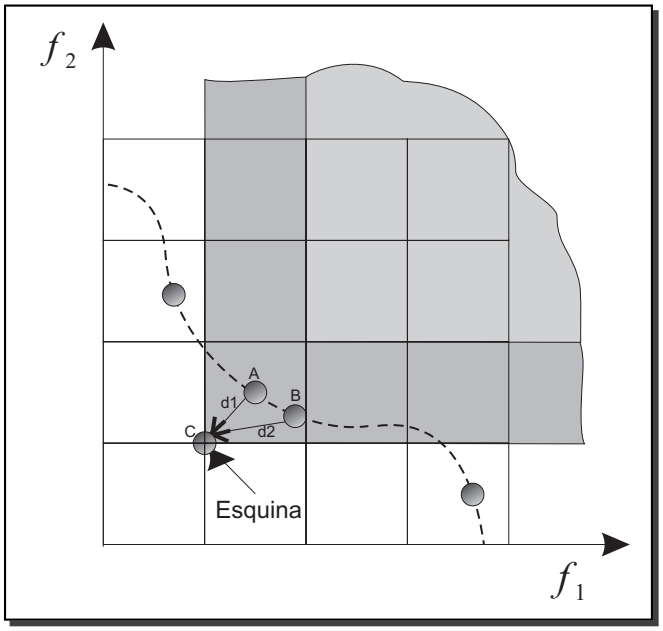


Figura 6.2: Ilustración de dos soluciones ϵ-dominadas iguales.

6.1.1.2 Selección del líder

La propuesta de selección del líder se realiza con base en una topología local y considerando la evaluación de una función objetivo a la vez. Para evaluar la función objetivo se consideran dos casos:

1. En el primer caso la función objetivo que se considera es aquella que corresponde a la $i-$ésima función que se esté optimizando en ese momento, como se muestra en la ecuación (6.5).

2. Para el segundo caso se utiliza una función agregativa no lineal; esto se expresa en la ecuación (6.6).

Lo que se realiza es optimizar cada una de las funciones un número determinado de evaluaciones. La idea es intensificar la búqueda de la mejor solución para una sola función objetivo a fin de obtener $f_i^*(\vec{x})$ como se observa el la figura 6.3. La imagen del lado izquierdo corresponde al comportamiento deseado al optimizar a $f_1(\vec{x})$. La imagen del lado decrecho corresponde a optimizar sobre la función $f_2(\vec{x})$.

Una vez que se obtiene a las mejores soluciones para cada función objetivo ($f_i^*(\vec{x})$), se optimiza la función agregativa no lineal (ecuación 6.6) con la finalidad de obtener a la mejor solución compromiso entre las soluciones que optimizan a cada una de las funciones objetivo. En la figura 6.4 se observa el comportamiento deseado de las partículas de la población al optimizar en la dirección de la función agregativa.

$$G(x) = f_i(\vec{x}) \tag{6.5}$$

$$G(x) = \sum_{i=1}^{N} \frac{|f_i^*(\vec{x}) - f_i(\vec{x})|}{|f_i^*(\vec{x})|} \tag{6.6}$$

En el Algoritmo 8 se describe el proceso de selección del líder. Se obtiene un líder dentro de una topología local *(lbest)*. Dicho líder se obtiene de la población secundaria (descrita en la sección anterior). Primero se calculan las distancias euclidianas entre la población secundaria y la partícula $\vec{X}$. Se ordenan las distancias calculadas de menor a mayor y se obtiene a la mejor partícula con respecto a la función objetivo que se esté optimizando dentro del vecindario determinado, para ello se requiere definir el tamaño del vecindario.

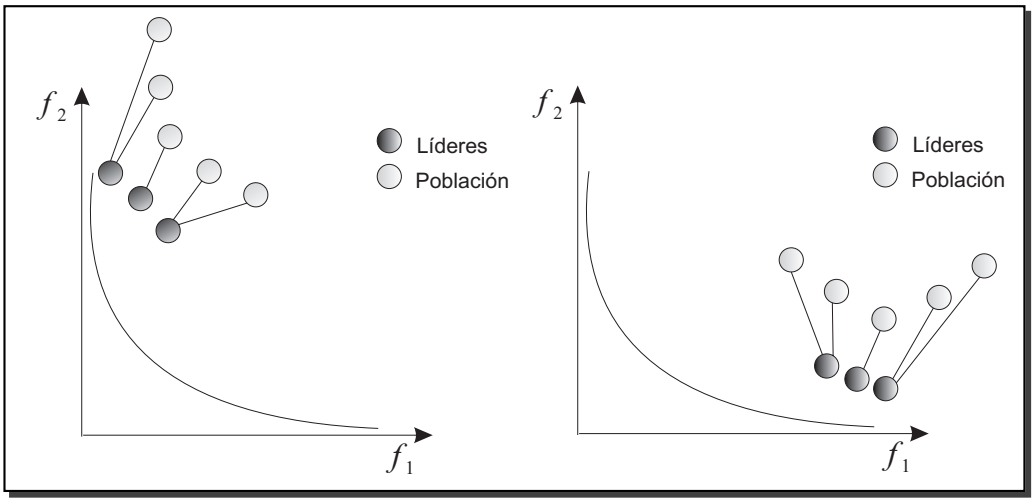


Figura 6.3: Selección del líder utilizando una función objetivo a la vez.

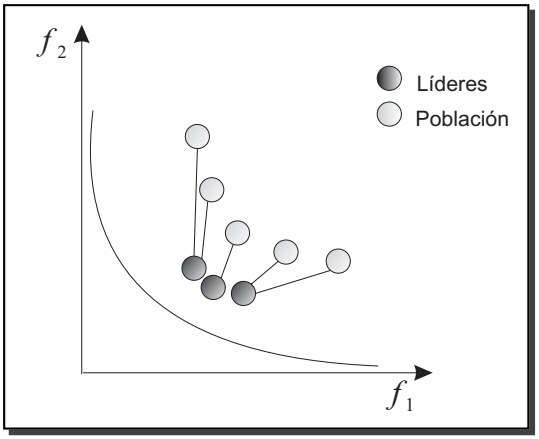


Figura 6.4: Selección del líder utilizando una función agregativa no lineal.

Algoritmo 8: Algoritmo de selección del líder basado en optimización global.

Datos de entrada: *Partícula* $\vec{X}$
Resultado: *Partícula líder* $PopSecundaria[g]$
/*Obtener la distancia euclidiana de la partícula $\vec{X}$ con todas las partículas de la población secundaria.*/
for *Cada partícula* $\vec{P}_i \in$ *población secundaria* **do**
    $Distancias[i][1] \leftarrow i$
    $Distancias[i][2] \leftarrow DistanciaEuclidiana(\vec{X}, \vec{P}_i)$
end
/*Ordenar las distancias de menor a mayor utilizando el algoritmo Quick Sort*/
$DistanciasOrdenadas \leftarrow QuickSort(Distancias)$
$g \leftarrow DistanciasOrdenadas[1][1]$
for $i \leftarrow 2$ **to** *Número de vecinos* **do**
    $pos\_particula \leftarrow DistanciasOrdenadas[i][1]$
    if $G(PopSecundaria[pos\_particula]) < G(PopSecundaria[g])$ **then**
        $g \leftarrow pos\_particula$
    end
end
return *(*$PopSecundaria[g]$*)*

6.1.1.3 Recombinación BLX-α

En la mayoría de los problemas de optimización se tiene la restricción de los límites de las variables. Para tratar con esta restricción utilizamos un operador de recombinación. Cuando se utiliza un operador de recombinación lo deseable es contar con dos características fundamentales: un grado de exploración y otro de explotación. Los operadores que cuentan con estas dos características son operadores robustos.

El operador de recombinación BLX-α funciona de la siguiente manera: Sean $c_i^1, c_i^2 \in [a_i, b_i]$ dos partículas a combinar con $\alpha_i = \text{mín}\{c_i^1, c_i^2\}$ y $\beta_i = \text{máx}\{c_i^1, c_i^2\}$, considerando un intervalo de acción $[a_i, b_i]$, lo anterior origina tres intervalos o regiones: $[a_i, \alpha_i]$,$[\alpha_i, \beta_i]$ y $[\beta_i, b_i]$ (véase la figura 6.5). Estas regiones se clasifican en zonas exploratívas y explotativas.

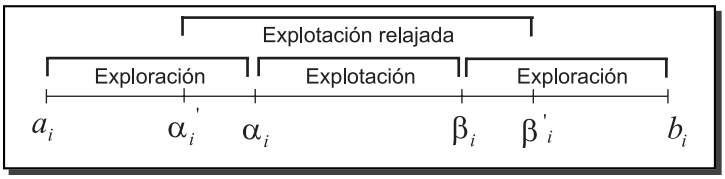


Figura 6.5: Intervalos de acción del operador de recombinación BLX-α.

Se denomina zona *explotativa* debido a que algún valor g_i generado por un operador de cruza en este intervalo cumple:

$$\text{máx}\{|g_i - \alpha_i|, |g_i - \beta_i|\} \leq |\alpha_i - \beta_i| \tag{6.7}$$

En consecuencia, se denomina zona *explorativa* a la región que no cumple la relación 6.7.

La región $\alpha_i^{'}, \beta_i^{'}$ se obtiene con las ecuaciones (6.8) y (6.9) respectivamente, donde $I = \beta_i - \alpha_i$.

$$\alpha_i^{'} = \alpha_i - I \cdot \alpha \tag{6.8}$$
$$\beta_i^{'} = \beta_i - I \cdot \alpha \tag{6.9}$$

La región delimitada por los extremos $\alpha_i^{'}$ y $\beta_i^{'}$ se considera zona de *explotación relajada*, ya que existe un grado de exploración y/o explotación el

cual es asignado al operador de recombinación.

BLX-α genera un nuevo vector $\vec{G} = (g_i, g_2, \ldots, g_d)$, donde g_i es un valor aleatorio uniforme en el intervalo $[\alpha_i^{'}, \beta_i^{'}]$.

Nomura [66] realiza un análisis de dicha recombinación con base en una función de densidad de probabilidad del cromosoma antes y después de aplicar el operador, asumiendo una población infinita. Los autores observan que BLX-α provee una buena dispersión de los valores cuando $\alpha > (\sqrt{3} - 1)/2$; en caso contrario, se reduce la dispersión. También observan que utilizar un valor de $\alpha = 0$ hace que la varianza de la distribución de los cromosomas disminuya (reduciendo la dispersión). Sin embargo, utilizar un valor de $\alpha = 0.5$ hace que la varianza de la distribución de la variable aleatoria se incremente (aumentando la dispersión), la cual es relativa a la posición de los padres.

6.1.1.4 Mutación

La mutación, como se hizo mención anteriormente, es un operador que provee diversidad dentro de la población.

En esta tesis hacemos uso de la mutación denominada *Parameter-Based Mutation* (PBM) propuesta por Deb [22, 26]. Utiliza una distribución de probabilidad polinomial para crear una solución y en la vecindad de una variable padre x.

El algoritmo que describe al PBM se muestra en el Algoritmo 9, donde $[x^u, x^l]$, son los límites superiores e inferiores permisibles de la variable x, $U(0,1)$ es un valor aleatorio uniforme entre 0 y 1, η_m es el índice de distribución para la mutación (determina la cantidad de perturbación de la variable) y debe ser un valor positivo. En esta propuesta utilizamos un valor de $\eta_m = 10$.

Algoritmo 9: Algoritmo Parameter-Based Mutation.

Datos de entrada: *Un valor de la variable x, límite superior x^u, límite inferior x^l*
Resultado: *Un valor y*
$\delta 1 = (x - x^l)/(x^u - x^l)$
$\delta 2 = (x^u - x)/(x^u - x^l)$
$\delta = min(\delta 1, \delta 2)$
$u = U(0,1)$
if $u \leq 0.5$ **then**
    $\bar{\delta} = [2u + (1-2u)(1-\delta)^{\eta_m+1}]^{\frac{1}{\eta_m+1}} - 1$
else
    $\bar{\delta} = 1 - [2(1-u) + 2(u-0.5)(1-\delta)^{\eta_m+1}]^{\frac{1}{\eta_m+1}}$
end
$\Delta_{max} = x^u - x^l$
$y = x + \bar{\delta}\Delta_{max}$

6.1.1.5 Descripción del algoritmo.

El algoritmo propuesto se describe en el Algoritmo 10. Primero generamos una población de partículas aleatoriamente, el siguiente paso es optimizar a la primera función objetivo $f_1(\vec{x})$ un número determinado de evaluaciones. Para cada partícula aplicamos los siguientes mecanismos:

1. $ObtenerLiderLocal(\vec{x_i})$ obtiene al líder que guiará la búsqueda de la partícula $i-$ésima, utilizando el criterio de selección de líder propuesto en la sección 6.1.1.2 (en la página 54). Como se mencionó anteriormente se considera el valor de la función objetivo a optimizar de las partículas vecinas que se encuentran en la población secundaria para determinar a dicho líder. La partícula que se encuentre dentro del vecindario con el mejor valor de la función objetivo es seleccionada.

2. Aplicar la ecuación de vuelo del PSO (ecuación 4.1 en la página 36) y actualizar la posición de la partícula (ecuación 4.2 en la página 36).

3. Si la partícula se sale de las dimensiones permisibles en el espacio de las variables, utilizamos la recombinación BLX-α descrita en la sección 6.1.1.3. La recombinación la efectuamos entre la partícula personal (es la mejor partícula encontrada por la $i-$ésima partícula) y la partícula elegida como líder.

4. Utilizamos el operador de mutación descrito en la sección 6.1.1.4, para evitar que la partícula se quede atrapada en un óptimo local. Como se mencionó anteriormente la mutación provee diversidad dentro de la población. Aplicamos dicho operador con una cierta probabilidad de mutación. En [28] se recomienda un valor de $p_m = 1/NumVariables$.

5. La función $Evaluar(\vec{x_i}, fprincipal)$, evalúa las funciones objetivo, además de determinar el valor de $G(\vec{x_i})$ de la partícula en base a las ecuaciones (6.5) y (6.6) dependiendo de la función que se esté optimizando. Si la función a optimizar el la $NumFunciones + 1$ se utiliza la función agregativa (ecuación 6.6) la cual trata de optimizar una función que obtenga un conjunto de soluciones compromiso entre las funciones objetivo del problema. También es este punto se determina la localidad de la partícula dentro del hipercubo para realizar las pruebas de la dominancia-ϵ.

6. Intentamos agregar la partícula $\vec{x_i}$ a la población secundaria con el mecanismo descrito en la sección 6.1.1.1.

7. Utilizamos la función $G(\vec{x_i})$ para comparar la nueva partícula $\vec{x_i}$ con la partícula personal (*pbest* o $\vec{p_i}$). Si la nueva partícula $\vec{x_i}$ es mejor, entonces $\vec{p_i}$ toma el valor de $\vec{x_i}$.

Una vez que se realizan los pasos anteriores un determinado número de evaluaciones cambiamos de función a optimizar.

6.1.2 Mecanismo de selección basado en exploraciones de diversas regiones dentro de la población secundaria

En esta sección describimos un algoritmo el cual se distingue por la selección de los líderes de la población en base al conjunto de soluciones objetivo que se esté optimizando. Por otro lado, se utiliza un criterio de selección dentro de una topología global a cada iteración del algoritmo. Además se actualiza la partícula personal en base a un criterio de dominancia de Pareto.

6.1.2.1 Selección del líder

Para proponer esta selección nos basamos en el hecho de que la cooperación es quizás la característica más importante de los cúmulos de partículas. El explorar diferentes regiones al mismo tiempo nos podría llevar a obtener mejores soluciones de manera rápida.

La idea de la siguiente propuesta radica, a diferencia de la propuesta realizada en la sección 6.1.1 (en la página 50), que podemos obtener mejores resultados si se realizan exploraciones en diferentes regiones del espacio de búsqueda. Las regiones que se exploran están dadas por el número de funciones objetivo y la localización de los mejores resultados obtenidos para cada objetivo además de la ubicación de una partícula ideal.

Una de las características que sobresalen es el uso de una población de partículas muy pequeña (proponemos una población de cinco partículas). En la figura 6.6 mostramos el mecanismo de selección de líderes para cada partícula. Las partículas seleccionadas se obtienen de la población secundaria (descrita anteriormente). Las partículas denominadas líderes, tienen la

Algoritmo 10: Algoritmo del PSO basado en optimización global.

Datos de entrada: *Una lista de partículas* $\vec{x}$
Resultado: *Población secundaria (una aproximación al conjunto de óptimos de pareto)*
Inicializar una población de partículas con posiciones aleatorias en la dimensión del espacio de búsqueda del problema
for $fprincipal \leftarrow 1$ *to* $NumFunciones + 1$ **do**
    repeat
        for $i \leftarrow 1$ *to* $NumParticulas$ **do**
            $\vec{p_l} \leftarrow ObtenerLiderLocal(\vec{x_i})$
            for $d \leftarrow 1$ *to* $NumVariables$ **do**
                $v_{id} \leftarrow v_{id} + c_1 \cdot U(0,1) \cdot (p_{id} - x_{id}) + c_2 \cdot U(0,1)(p_{ld} - x_{id})$
                $x_{id} \leftarrow x_{id} + v_{id}$
            end
            if $\vec{x_i} \notin$ *espacio de búsqueda* **then**
                $\vec{x_i} \leftarrow BLX - \alpha(\vec{p_i}, \vec{p_l})$
            end
            if $U(0,1) < p_m$ **then**
                $Mutar(\vec{x_i})$
            end
            $Evaluar(\vec{x_i}, fprincipal)$
            if $\vec{x_i}$ *no es* ϵ*-dominado* **then**
                Agregar la solución $\vec{x_i}$ *a la población secundaria*
            end
            if $G(\vec{x_i}) > G(\vec{p_i})$ **then**
                for $d \leftarrow 1$ *to* $NumVariables$ **do**
                    $p_{id} \leftarrow x_{id}$
                end
            end
        end
    until $MaxIter$
end

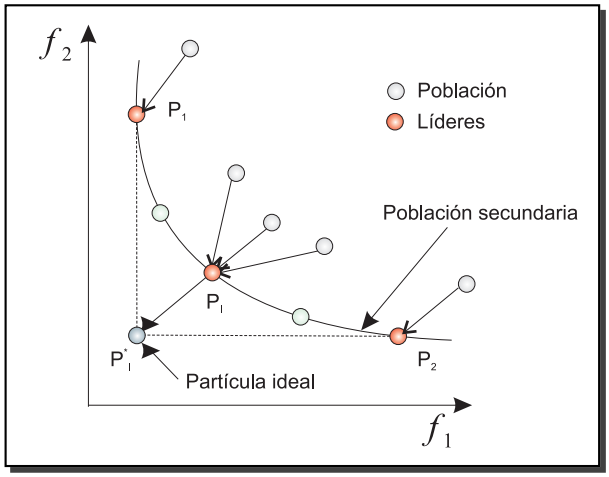


Figura 6.6: Selección del líder creando un vector ideal.

características de elegirse de acuerdo a cada uno de los objetivos que contenga el problema. Adicionalmente creamos una partícula de forma artificial (partícula ideal) para elegir una partícula dentro de la población secundaria (una partícula real) que se acerque más a la partícula ideal.

La mejor solución compromiso entre las diferentes funciones objetivo a optimizar dentro de la población secundaria la obtenemos de la siguiente forma:

1. Obtener las mejores partículas que optimicen mejor a cada función objetivo hasta el momento: $P_1 = \{f_1^*, f_2, \ldots, f_n\}$, $P_2 = \{f_1, f_2^*, \ldots, f_n\}$, $P_n = \{f_1, f_2, \ldots, f_n^*\}$.

2. Construir la partícula ideal con los mejores valores de cada función objetivo: $P_I^* = \{f_1^*, f_2^*, \ldots, f_n^*\}$.

3. Se elige a la partícula dentro de la población secundaria que se acerque más utilizando una distancia euclidiana a P_I^*, esta partícula se etiqueta como P_I.

De esa forma obtenemos a las partículas líderes que guiarán la búsqueda de mejores soluciones.

6.1.2.2 Descripción del algoritmo

En el algoritmo 11 presentamos la propuesta para optimizar en diferentes regiones del espacio de búsqueda. A continuación describimos los mecanismos del algoritmo:

1. Inicializamos una población de partículas de manera aleatoria.
2. La función $ObtenerLideres()$ obtiene un conjunto de partículas dentro de la población secundaria y las almacena en una lista denominada L, la cardinalidad de la lista L es del número de funciones objetivo a optimizar más uno, la partícula adicional se refiere a la partícula ideal. El mecanismo de selección de los líderes se describió anteriormente en la sección 6.1.2.1.
3. La función $ObtenerLider(i)$ devuelve el índice del líder correspondiente a la partícula $i-$ésima dentro de la lista L. Es decir, para las primeras n partículas (donde n es el número de funciones objetivo) devuelve la mejor partícula que optimiza a la $i-$ésima función objetivo. Para las partículas restantes se les asigna la partícula más cercana a la partícula ideal, con la finalidad de intensificar la búsqueda en la zona en donde se encuentra la solución compromiso entre las funciones que se estén optimizando.
4. Aplicar la ecuación de vuelo del PSO (ecuación 4.1) y actualizar la posición de la partícula (ecuación 4.2).
5. Si la partícula se sale de las dimensiones permisibles en el espacio de las variables. Utilizamos la recombinación BLX-α descrita en la sección 6.1.1.3. La recombinación la efectuamos entre la partícula personal y la partícula líder que le corresponde.
6. Utilizamos el operador de mutación descrito en la sección 6.1.1.4 con una cierta probabilidad.
7. La función $Evaluar(\vec{x_i})$, evalúa las funciones objetivo. En esta función se determina la localidad de la partícula dentro del hipercubo para realizar las pruebas de la dominancia-ϵ.
8. Se realiza una prueba de dominancia (dominancia-ϵ). Si la partícula no es dominada-ϵ se actualiza la partícula personal al valor de $\vec{x_i}$ y se agrega a la población secundaria.

6.2 Algoritmo de búsqueda dispersa

La búsqueda dispersa es un mecanismo que ha tenido una relevancia importante dentro de los algoritmos evolutivos en los últimos años. En esta tesis haremos uso de este método para realizar una mayor exploración en las zonas donde el PSO no tuvo mucho éxito. De tal forma que se logren acercar más las soluciones al verdadero frente de Pareto.

En la figura 6.7 ilustramos el esquema de la propuesta realizada en cuanto a la búsqueda dispersa se refiere. El mecanismo para crear el conjunto de soluciones iniciales es reemplazado por el algoritmo del PSO descrito en la sección anterior. Adicionalmente proponemos un mecanismo para obtener soluciones dispersas. El método para actualizar el conjunto de referencia toma una semilla (una solución dispersa) para generar el conjunto de referencia, posteriormente el método para generar subconjuntos crea los subconjuntos a partir del conjuntos de referencia, el método de combinación hace uso de la recombinación BLX-α descrito en la sección 6.1.1.3, después se aplica un método de mejora el cual intentará mejorar la solución generada por el método de recombinación (se aplica este método a todos los subconjuntos generados). Se vuelve a repetir el proceso para generar el conjunto de referencia y los pasos subsecuentes hasta que se hayan utilizado todas las soluciones dispersas.

A continuación se describen de manera detallada los mecanismos principales de la búsqueda dispersa realizados en esta tesis.

6.2.1 Conjunto de soluciones iniciales

En el esquema de la búsqueda dispersa descrito en la seccion 5.2 existe una etapa inicial para generar un conjunto de soluciones diversas y así crear un conjunto denominado conjunto P. Al realizar un híbrido como es el caso en esta tesis existe una etapa que debe servir como enlace entre los dos algoritmos. Esa etapa es la de crear el conjunto P. Para ello usamos el conjunto de soluciones obtenidas por el PSO. Con el algoritmo del PSO obtenemos una población de partículas secundaria y una población terciaria.

La población secundaria está conformada por las soluciones no dominadas y la población terciaria se conforma por las soluciones dominadas en un

Algoritmo 11: Algoritmo del PSO basado en mecanismos de exploración en diversas regiones.

Datos de entrada: *Una lista de partículas* $\vec{x}$
Resultado: *Población secundaria (una aproximación al conjunto de óptimos de pareto)*
Inicializar una población de partículas con posiciones aleatorias en la dimensión del espacio de búsqueda del problema
$ObtenerLideres()$
repeat
  for $i \leftarrow 1$ *to* $NumParticulas$ **do**
    $g \leftarrow ObtenerLider(i)$
    for $d \leftarrow 1$ *to* $NumVariables$ **do**
      $v_{id} \leftarrow v_{id} + c_1 \cdot U(0,1) \cdot (p_{id} - x_{id}) + c_2 \cdot U(0,1)(L_{gd} - x_{id})$
      $x_{id} \leftarrow x_{id} + v_{id}$
    end
    if $\vec{x_i} \notin$ *espacio de búsqueda* **then**
      $\vec{x_i} \leftarrow BLX - \alpha(\vec{p_i}, \vec{L_g})$
    end
    if $U(0,1) < p_m$ **then**
      $Mutar(\vec{x_i})$
    end
    $Evaluar(\vec{x_i})$
    if $\vec{x_i}$ *es no es dominado-*ϵ **then**
      for $d \leftarrow 1$ *to* $NumVariables$ **do**
        $p_{id} \leftarrow x_{id}$
      end
      Agregar la solución $\vec{x_i}$ *a la población secundaria*
    end
  end
  $ObtenerLideres()$
until $MaxIter$

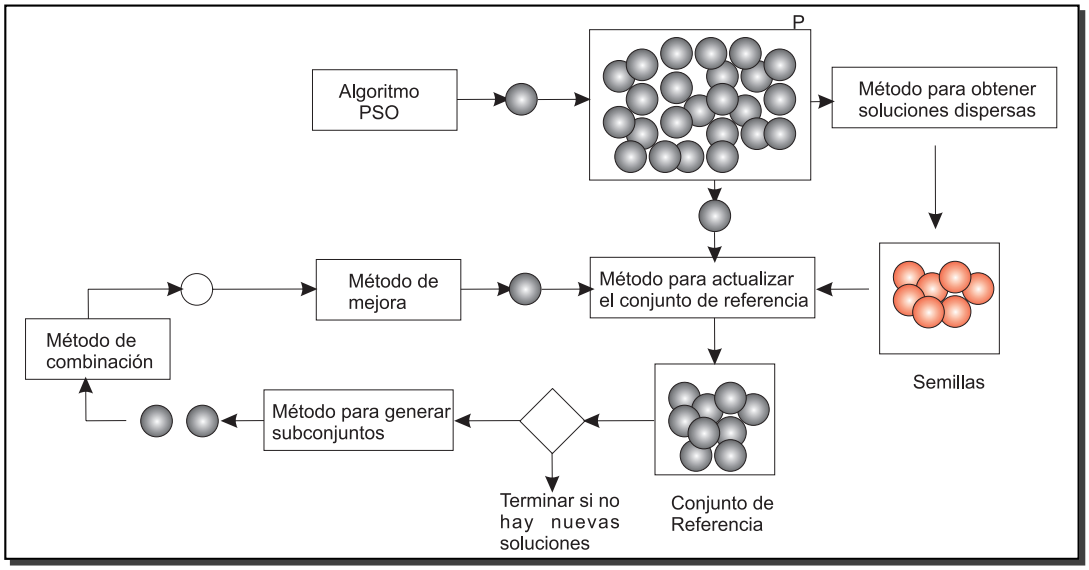


Figura 6.7: Esquema propuesto de la búsqueda dispersa.

segundo nivel; es decir, las soluciones que salen de la población secundaria se van a la población terciaria con una prueba de dominancia de Pareto. Todas estas soluciones conforman el conjunto de enlace, el denominado conjunto P.

En la figura 6.8 se ilustran las dos poblaciones y por consiguiente al nuevo conjunto P.

6.2.2 Método para obtener soluciones dispersas

Para obtener soluciones dispersas se utilizó un criterio de optimalidad en donde se hace uso de nociones de optimización compromiso [33, 89]. Dado un punto ideal el cual está compuesto de los mejores valores encontrados en cada función objetivo, la programación compromiso asume que es preferible un punto que se encuentre lo más cerca del punto ideal sobre aquellos que están alejados.

Una medida de la "solución más cercana a la ideal" es la denominada métrica L_q (donde $1 \leq q \leq \infty$) definida en la ecuación 6.10 la cual es una generalización de la distancia euclidiana (con $q = 2$).

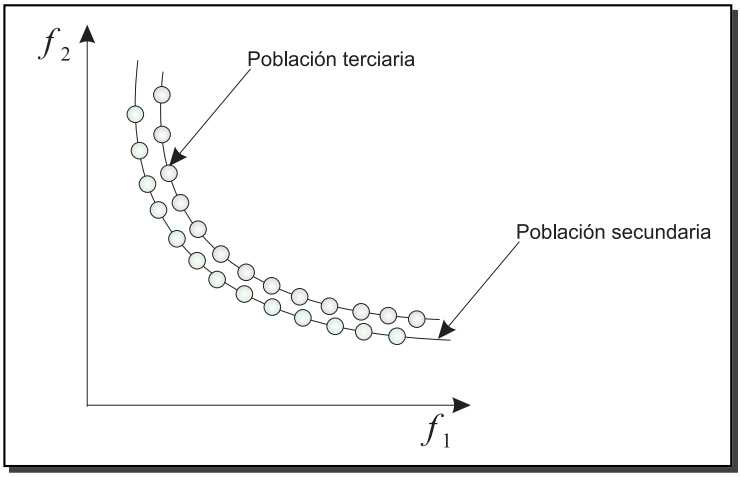


Figura 6.8: Conjunto de soluciones iniciales. El conjunto P es la unión de la población secundaria y terciaria.

El valor de q indica el tipo de distancia. Para $q = 1$, todas las desviaciones son tomadas en cuenta en proporción a la dirección de las magnitudes, la que se considera una "utilidad grupal". Para $2 \leq q \leq \infty$, las desviaciones grandes acarrean grandes pesos en la distancia L_q. Para $p = \infty$ la desviación más grande es la única que se toma en cuenta lo que se considera una "utilidad individual". L_1 y L_∞ son métricas lineales, no así para el resto de la familia. Se prefiere el uso de la métrica L_∞ porque muestra un mejor balance entre las soluciones obtenidas. La métrica L_∞ da como resultado un criterio global de mín − máx.

$$L_q(f^{(1)}, f^{(2)}) = \left[\sum_{i=1}^{n} \left| f_i^{(1)}(\vec{x}) - f_i^{(2)}(\vec{x}) \right|^q \right]^{1/q} \tag{6.10}$$

$$L_\infty(\vec{x}) =_{i=1,\ldots,n}^{\text{máx}} \left\{ \frac{f_i^{\text{máx}}(\vec{x}) - f_i(\vec{x})}{f_i^{\text{máx}}(\vec{x}) - f_i^{\text{mín}}(\vec{x})} \right\} \tag{6.11}$$

La distancia L_∞ da como resultado la máxima desviación relativa comparando una solución en las n dimensiones. El óptimo mín − máx compara la desviaciones relativas de las máxima separaciones y da como resultado la mínima.

Generalizando la ecuación 6.11 para obtener el conjunto de puntos dispersos obtenemos la expresión 6.12:

$$\max_{\forall u \in U} \left\{ \min_{\forall v \in V} \left\{ \max_{i=1,\ldots,n} \left\{ \frac{|f_{vi}(\vec{x}) - f_{ui}(\vec{x})|}{f_i^{\max}(\vec{x}) - f_i^{\min}(\vec{x})} \right\} \right\} \right\} \tag{6.12}$$

En la expresión 6.12 el conjunto U contiene a las soluciones no dominadas y el conjunto V contiene a las soluciones dispersas. Inicializamos el conjunto V con las mejores soluciones para cada función objetivo. Posteriormente utilizamos la expresión 6.12 para obtener las siguientes soluciones dispersas, tantas como se deseen.

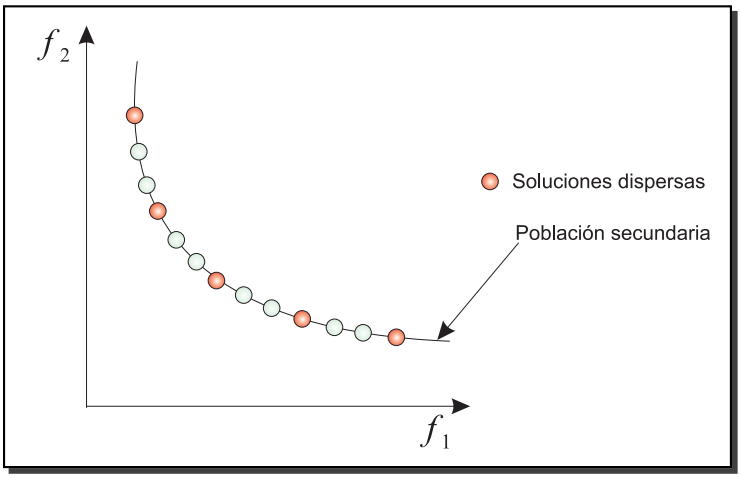


Figura 6.9: Soluciones dispersas.

En la figura 6.9 los puntos denominados soluciones dispersas nos servirán para crear el conjunto de referencia.

6.2.3 Método para actualizar el conjunto de referencia

Este es uno de los métodos más relevantes de la búsqueda dispersa. Nuestra propuesta va enfoncada en mantener el concepto de dispersión durante toda la búsqueda. Este método se basa en tener un conjunto de soluciones dispersas. Dicho conjunto es el que se ilustra como *semillas* en la figura 6.7.

Para explicar el mecanismo haremos referencia a la figura 6.10. Utilizamos dos poblaciones: una población secundaria y una población terciaria. Obtenemos una solución dispersa del conjunto denominado *semillas*. A partir del conjunto P (el cual está compuesto por una población secundaria y una terciaria) seleccionamos las m soluciones más cercanas a la semilla elegida (utilizando una distancia euclidiana). Las regiones marcadas que encierran a las soluciones en la figura 6.10 ilustran el conjunto de referencia obtenido.

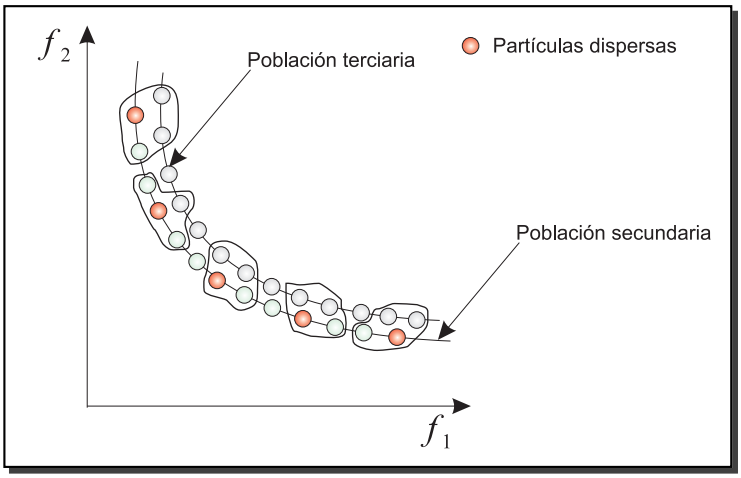


Figura 6.10: Selección de soluciones para crear el conjunto de referencia

Este proceso es realizado hasta que se vacíe el conjunto *Semillas* y entonces, se vuelve a obtener dicho conjunto.

6.2.4 Método para generar subconjuntos

Este método es el encargado de determinar cómo se llevará a cabo la combinación de las soluciones, es decir determina los subconjuntos que se van a generar para que en una etapa posterior se les aplique un operador de recombinación.

En esta propuesta se generan subconjunto de dos soluciones. De tal forma que si tenemos p soluciones en el conjunto de referencia obtendremos $\frac{p\cdot(p-1)}{2}$ subconjuntos.

6.2.5 Método de combinación

En esta fase se utiliza el método de recombinación BLX-α descrito en la sección 6.1.1.3 (en la página 58), que como ya se ha mencionado es un método que tiene la característica de explorar y explotar la vecindad en donde se encuentran las soluciones a combinar dicha característica está determinada por un valor de α [66].

Este método es aplicado a los subconjuntos de dos soluciones generados en la etapa anterior, de tal manera que obtendremos $\frac{p\cdot(p-1)}{2}$ descendientes del conjunto de referencia.

6.2.6 Metodo de mejora

El método de mejora básicamente es un mecanismo de búsqueda local. Este mecanismo se utiliza para mejorar las soluciones obtenidas como producto de la combinación de las soluciones que se encuentran en el conjunto de referencia.

Aprovechamos la flexibilidad del esquema de la búsqueda dispersa para utilizar un mecanismo aleatorio que permita en un caso dado obtener soluciones aún cuando las soluciones que se obtengan sean óptimos locales.

Como método de mejora hacemos uso de la mutación basada en parámetros descrito en la sección 6.1.1.4. En un algoritmo en donde la mayoría de sus elementos son sistemáticos es importante agregar un componente que

provea diversidad como lo es la mutación.

La principal desventaja del esquema original es que las soluciones que en un inicio son soluciones de diversidad y de calidad eventualmente quedan soluciones de calidad y es probable que se pierda la diversidad que se tenía en un principio.

6.2.7 Descripción del algoritmo

A continuación describimos los elementos que conforman el Algoritmo 12, propuesto para la búqueda dispersa:

1. La función $ObtenerSolucionesDispersas()$ genera la lista *semillas* descrito en el esquema 6.7 el cual es obtenido como se describió en la sección 6.2.2 como *método para obtener soluciones dispersas.*

2. La variable $MaxSolucionesDispersas$ es un parámetro del algoritmo el cual determina el número de *semillas* que deseamos.

3. La función $ObtenerSolucionDispersa()$ devuelve la $n-$ésima semilla para generar el conjunto de referencia y la remueve de la lista *semillas*.

4. La función $ActualizarConjuntoReferencia()$ es el *método para actualizar el conjunto de referencia* el cual obtiene el conjunto utilizando la semilla adquirida en el paso anterior como se describió en la sección 6.2.3 en donde se obtienen $TamConjRef$ soluciones que conformarán dicho conjunto.

5. Las líneas 10 y 11 del algoritmo en conjunto son el *método para generar subconjuntos* descrito en la sección 6.2.4.

6. La función $BLX-\alpha()$ es el *método de combinación* descrito en la sección 6.2.5. Dicho método se aplica sobre la $i-$ésima y $j-$ésima solución del conjunto de referencia.

7. La función $Mutar()$ es el *método de mejora* descrito en la sección 6.2.6 el cual se trata de una búsqueda local sobre la solución obtenida por el método de combinación.

8. Finalmente se realiza una prueba de dominancia (dominancia-ϵ). Si la solución no es dominada-ϵ se agrega a la población secundaria y las soluciones que pudieran salir de la población secundaria se intentarán agregar a una población terciaria.

Algoritmo 12: Algoritmo de la búsqueda dispersa propuesto

```
1  Datos de entrada: Población secundaria y una población terciaria
2  Resultado: Población secundaria (una aproximación al conjunto de óptimos de pareto)
3  repeat
4      ObtenerSolucionesDispersas()
5      for n = 1 to MaxSolucionesDispersas do
6          pl = ObtenerSolucionDispersa()
7          //Método para actualizar el conjunto de referencia
8          ActualizarConjuntoReferencia(pl)
9          for i = 1 to TamConjRef do
10             for j = i + 1 to TamConjRef do
11                 //Método de combinación de soluciones
12                 x⃗ = BLX − α(pobConjRef(i), pobConjRef(j))
13                 //Método de mejora
14                 x⃗ = Mutar(x⃗)
15                 if x⃗ no es ε-dominado then
16                     Agregar la solución x⃗ a la población secundaria
17                 end
18             end
19         end
20     end
21 until MaxIter
```

7

Análisis de Resultados

En este capítulo presentamos los resultados obtenidos con las dos propuestas descritas en el capítulo anterior (sección 6.1.1 en la página 50 y sección 6.1.2 en la página 62).

Denominamos *MOPSOSS v1* al algoritmo con el mecanismo de selección basado en optimización global (descrito en la sección 6.1.1 en la página 50) en conjunción con el algoritmo de búsqueda dispersa (descrito en la sección 6.2 en la página 66). Denominamos *MOPSOSS v2* al algoritmo con el mecanismo basado en exploraciones de diversas regiones dentro de la población secundaria (descrito en la sección 6.1.2 en la página 62) junto con el algoritmo de la búsqueda dispersa (descrito en la sección 6.2 en la página 66).

Con el objeto de poder realizar una comparación entre los diferentes algoritmos hacemos uso de las métricas descritas en la siguiente sección:

7.1 Métricas

7.1.1 Distancia generacional invertida (DGI)

La distancia generacional (DG) fue propuesta por Van Veldhuizen y Lamont [83, 84]. La DG es una métrica que indica qué tan lejos o cerca están los puntos obtenidos por un algoritmo multiobjetivo del verdadero frente de Pareto.

La ecuación 7.1 define la métrica. El valor de n es la cardinalidad del conjunto de soluciones encontradas por el algoritmo propuesto, d_i es la distancia euclidiana entre la n-ésima solución encontrada por el algoritmo y la solución más cercana dentro del verdadero frente de Pareto.

$$\mathrm{DG} = \sqrt{\sum_{i=1}^{n} d_i^2} \tag{7.1}$$

La DGI utiliza el mismo concepto. Sin embargo, a diferencia de la DG, en este caso se utiliza como base el verdadero frente de Pareto. El valor de n es la cardinalidad del conjunto de soluciones del verdadero frente de Pareto, d_i es la distancia euclidiana entre la n-ésima solución del verdadero frente de Pareto y la solución más cercana dentro de las soluciones obtenidas por el algoritmo propuesto en el espacio de las funciones. Un valor de cero significa que todas las soluciones obtenidas por el algoritmo pertenecen al conjunto de óptimos de Pareto.

7.1.2 Distribución (D)

Esta métrica fue propuesta por Deb [28]. Mide la distribución de las soluciones obtenidas por el algoritmo propuesto. La ecuación 7.2 define la métrica.

$$D = \frac{d_f + d_l + \sum_{i=1}^{n-1} \left|\bar{d} - d_i\right|}{d_f + d_l + (n-1) \cdot \bar{d}} \tag{7.2}$$

donde d_i es la distancia euclidiana entre las soluciones adyancentes (las soluciones son ordenadas de acuerdo a cada una de las funciones objetivo), $\bar{d}$ es

la media de esas distancias, d_f y d_l son las distancias euclidianas a los extremos de las soluciones del frente de Pareto verdadero como una penalización al algoritmo en caso de no obtener las soluciones que conforman los extremos del verdadero frente de Pareto. En esta métrica un valor de cero indica que las soluciones están espaciadas de forma equidistante y que el algoritmo encontró las soluciones de los extremos del frente de Pareto verdadero.

7.1.3 Cobertura (C)

Esta métrica fue propuesta por Zitzler, Deb y Thiele [90]. Esta métrica compara un conjunto de soluciones con respecto a otro, utilizando la dominancia de Pareto. La métrica se define en la ecuación 7.3.

$$C(X', X'') = \frac{|\{a'' \in X''; a' \in X' : a' \prec a''\}|}{|X''|} \tag{7.3}$$

Si todos los puntos en X' dominan o son iguales a todos los puntos de X'', esto implica que $C(X', X'') = 1$. Si ningún punto de X' domina a algún punto del conjunto X'' entonces $C(X', X'') = 0$. Cuando $C(X', X'') = 1$ y $C(X'', X') = 0$ entonces se dice que el conjunto X' es mejor que X''.

7.2 Comparación de resultados

La premisa fundamental de esta tesis es tener una alta convergencia al verdadero frente de Pareto en los algoritmos que se proponen. Debido a ello se efectuaron diversas pruebas a fin de determinar el umbral crítico que debía utilizarse para diseñar nuestro algoritmo híbrido. En estos experimentos se determinó que para $7,000$ o más evaluaciones de la función de aptitud, tanto el NSGA-II como el SPEA2 son capaces de converger al verdadero frente de Pareto de los problemas ZDT. De tal forma, era importante que nuestro híbrido no llegase a ese número de evaluaciones. Tras realizar múltiples experimentos, determinamos que con $4,000$ evaluaciones de la función de aptitud lográbamos conseguir un nivel de competitividad adecuado, a fin de que nuestro algoritmo pudiera superar al NSGA-II y SPEA2.

Evidentemente, si nuestro algoritmo ejecuta un mayor número de evaluaciones, su desempeño mejora, pero deja de ser competitivo con respecto al NSGA-II y al SPEA2. De realizarse menos evaluaciones, nuestro algoritmo

produce resultados pobres, dado que el motor de búsqueda no logra generar soluciones suficientemente cerca del verdadero frente de Pareto. Resta, sin embargo, estudiar a más detalle el balance a realizarse entre el uso del motor de búsqueda y la búsqueda dispersa, pues de ahí podrían derivarse mayores ahorros en lo que al número total de evaluaciones se refiere.

En esta sección describiremos el estudio comparativo que realizamos entre las dos versiones de nuestro algoritmo (MOPSOSS v1 y MOPSOSS v2), y dos algoritmos representativos del estado del arte en optimización evolutiva multiobjetivo: SPEA2 [91] y NSGA-II [24, 25, 28]. Ambos algoritmos se describieron en la sección 3.4.3 (en la página 25).

Los parámetros utilizados en los algoritmos de nuestro estudio se muestran en la tabla 7.1, donde P es el tamaño de la población, $P_{\text{máx}}$ es el número máximo de soluciones que se pueden almacenar en la población secundaria, $G1_{\text{máx}}$ es el número de generaciones del algoritmo (para nuestra propuesta, $G1_{\text{máx}}$ indica el número de generación efectuadas por el PSO), P_c es la probabilidad de cruza, P_m es la probabilidad de mutación, $(w, c_1, c2)$ son los parámetros de vuelo del PSO, el parámetro α es el factor utilizado para la recombinación $BLX-\alpha$. Los parámetros $TamRefSet, SD, G2_{\text{máx}}$ de la segunda fase se refieren al algoritmo de la búsqueda dispersa; donde $TamRefSet$ es el tamaño del conjunto de referencia, SD es el número de soluciones dispersas y $G2_{\text{máx}}$ es el número de generaciones de la búsqueda dispersa.

En resumen, realizamos $4,000$ evaluaciones de la función objetivo para cada algoritmo. Los resultados obtenidos corresponden a la realización de 30 ejecuciones para cada uno de los problemas de prueba.

Para nuestro estudio comparativo adoptamos un conjunto de funciones de prueba descritas en el apéndice A.

Para cada función de prueba se muestran dos tablas:

1. En la primera tabla mostramos las métricas DGI y D. Para cada métrica obtenemos la media, el mejor valor obtenido, el peor, la desviación estándar y la mediana.

2. La segunda tabla corresponde a la métrica de cobertura, la cual utilizamos para comparar cada uno de los algoritmos contra todos los demás,

Parámetro	MOPSOSS v1	MOPSOSS v2	SPEA2	NSGA-II
P	5	5	100	100
$P_{\text{máx}}$	100	100	100	100
$G1_{\text{máx}}$	400	400	40	40
P_c	—	—	0.8	0.8
P_m	1/N	1/N	1/N	1/N
w	0.1	0.1	—	—
c_1	1.1	1.1	—	—
c_2	1.6	1.6	—	—
α	0.5	0.5	—	—
$TamRefSet$	4	4	—	—
SD	7	7	—	—
$G2_{\text{máx}}$	333	333	—	—

Tabla 7.1: Parámetros utilizados en los algoritmos para realizar las pruebas.

obteniendo un porcentaje en promedio del conjunto de soluciones que domina un algoritmo así como el porcentaje en promedio de soluciones que lo dominan.

Para tener un criterio adicional de comparación, mostramos las gráficas que se encuentran en la mediana con respecto a la métrica DGI para cada algoritmo.

Por otra parte obtenemos el porcentaje de efectividad de nuestras propuestas con base en la DGI con la fórmula siguiente:

$$\text{PE} = \left(1 - \frac{(\text{Media} - \text{Mejor})}{(\text{Peor} - \text{Mejor})}\right) \cdot 100 \tag{7.4}$$

7.2.1 Función de Kursawe

En la tabla 7.2 podemos observar que el algoritmo SPEA2 en la métrica DGI es mejor para todos los criterior empleados. Sin embargo, nuestra propuesta MOPSOSS v1 resulta ser mejor en cuanto a la distribución de las soluciones obtenidas (métrica D). En la tabla 7.3 se observa que el SPEA2 domina en promedio a un mayor porcentaje de las soluciones con el 24.33 % y es dominado por el 12.66 % de las soluciones.

En la figura 7.1 podemos ver que los 4 algoritmos son capaces de llegar al verdadero frente de Pareto con tan solo 4,000 evaluaciones de la función objetivo. Sin embargo, el SPEA2 tiene una ventaja marginal sobre las demás propuestas.

El porcentaje de efectividad para el MOPSOSS v1 es del 51 % mientras que para el MOPSOSS v2 es del 57 %.

Métrica	Algoritmo	Media	Mejor	Peor	Des. Estándar	Mediana
DGI	MOPSOSS v.1	0.005408	0.004852	0.005987	0.000334	0.005417
	MOPSOSS v.2	0.005296	0.004429	0.006471	0.000447	0.005307
	SPEA2	0.002843	0.002608	0.003359	0.000153	0.002841
	NSGA-II	0.003650	0.003229	0.004263	0.000280	0.003652
D	MOPSOSS v.1	0.392255	0.324460	0.447802	0.028808	0.392120
	MOPSOSS v.2	0.397889	0.336636	0.458496	0.025702	0.399704
	SPEA2	0.411025	0.344483	0.475073	0.035399	0.413897
	NSGA-II	0.438382	0.387781	0.545397	0.037092	0.438305

Tabla 7.2: Resultados de las métricas DGI y D para la función de Kursawe

C	MOPSOSS v.1	MOPSOSS v.2	SPEA2	NSGA-II	Dominan
MOPSOSS v.1	———	0.160998	0.136000	0.213667	16.66 %
MOPSOSS v.2	0.152804	———	0.131667	0.216667	16.33 %
SPEA2	0.211638	0.229026	———	0.306333	24.33 %
NSGA-II	0.156135	0.173833	0.123667	———	14.66 %
Son Dominados	17.36 %	18.33 %	12.66 %	24.55 %	———

Tabla 7.3: Resultados de la métrica de cobertura para la función de Kursawe

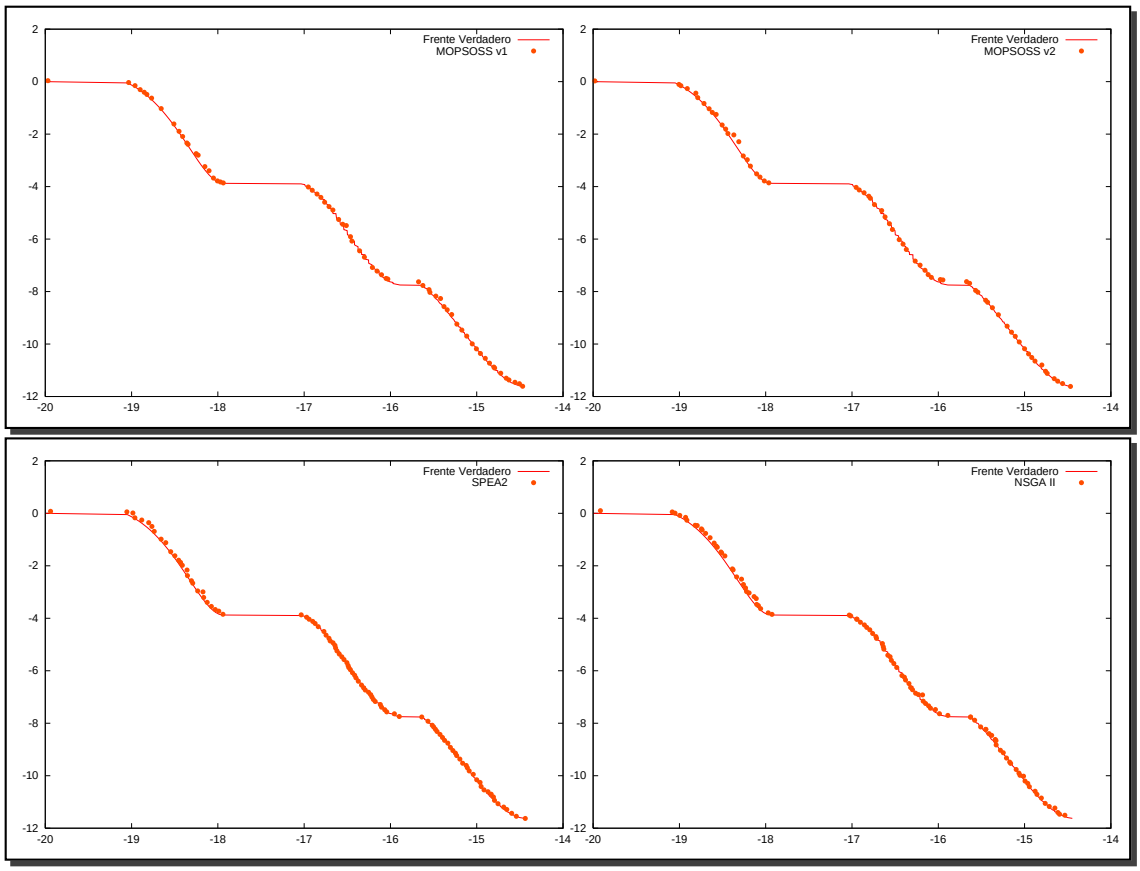


Figura 7.1: Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función de Kursawe

7.2.2 Función ZDT1

En la tabla 7.4 podemos notar que nuestra propuesta MOPSOSS v2 es mejor en todos los criterios que las otras propuestas en la métrica DGI. En cuanto a la distribución de las soluciones nuestra propuesta es mejor con excepción a la desviación estándar en donde el NSGA-II es marginalemente mejor. Esto no significa que nuestro algoritmo sea peor que el NSGA-II, sino que el NSGA-II se acerca al mismo resultado en las 30 corridas realizadas.

En la tabla 7.5 notamos que en cuanto a la cobertura el MOPSOSS v2 domina en promedio al 89.37 % y es dominado solamente por el 0.16 % lo que nos da una idea de qué tan bueno resultó este algoritmo para este problema y con respecto a los algoritmos evaluados.

Gráficamente podemos observar en la figura 7.2 que nuestra propuesta MOPSOSS v2 se acerca más al verdadero frente de Pareto que los demás algoritmos. Esto se puede constatar tanto gráficamente como con las métricas empleadas.

El porcentaje de efectividad para el MOPSOSS v1 es del 53 % mientras que para el MOPSOSS v2 es del 81 %.

Métrica	Algoritmo	Media	Mejor	Peor	Des. Estándar	Mediana
DGI	MOPSOSS v.1	0.022124	0.010463	0.035660	0.006758	0.022379
	MOPSOSS v.2	0.001851	0.000748	0.006460	0.001180	0.001993
	SPEA2	0.006551	0.004030	0.008888	0.001322	0.006557
	NSGA-II	0.009791	0.006058	0.013745	0.001904	0.009780
D	MOPSOSS v.1	0.570925	0.419279	0.753562	0.072896	0.568313
	MOPSOSS v.2	0.319018	0.221882	0.514988	0.063441	0.318180
	SPEA2	0.545766	0.453909	0.654551	0.047688	0.544490
	NSGA-II	0.544460	0.486246	0.636038	0.038006	0.544236

Tabla 7.4: Resultados de las métricas DGI y D para la función ZDT1

C	MOPSOSS v.1	MOPSOSS v.2	SPEA2	NSGA-II	Dominan
MOPSOSS v.1	———	0.000000	0.000000	0.014621	0.33 %
MOPSOSS v.2	0.972076	———	0.843024	0.866286	89.37 %
SPEA2	1.000000	0.004938	———	0.906372	58.22 %
NSGA-II	0.972139	0.000000	0.063916	———	63.70 %
Son Dominados	98.13 %	0.16 %	30.23 %	59.57 %	———

Tabla 7.5: Resultados de la métrica de cobertura para la función ZDT1

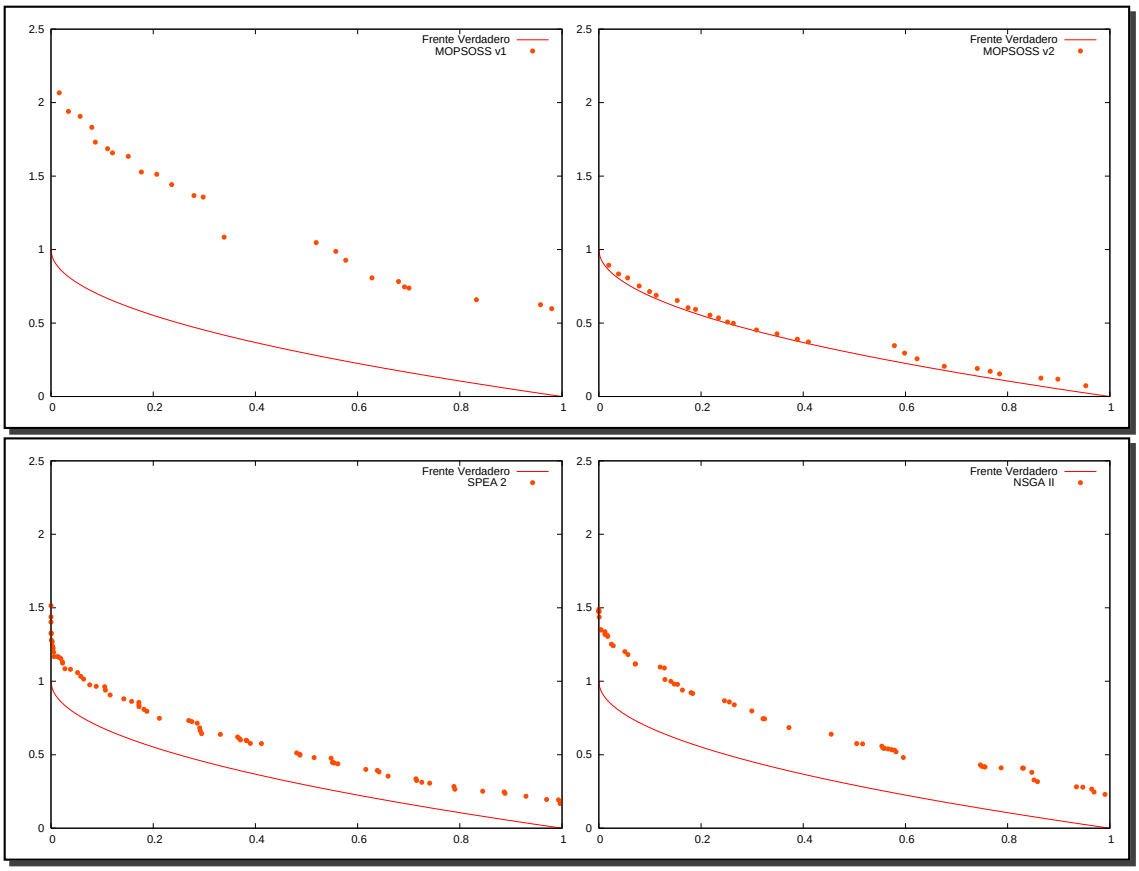


Figura 7.2: Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función ZDT1

7.2.3 Función ZDT2

En cuanto a las métricas que utilizamos para medir la efectividad de cada uno de los algoritmos, podemos notar que con la métrica DGI la propuesta MOPSOSS v2 es mejor en la mayoría de los criterios. Sin embargo, el MOPSOSS v1 tiene una mejor mediana lo que se refleja gráficamente. Esto se da por la similitud de los resultados obtenidos por estas dos propuestas.

En cuanto a la distribución, MOPSOSS v2 obtuvo los mejores resultados los cuales son marginales con respecto a MOPSOSS v1. Para la métrica de cobertura MOPSOSS v2 domina al 65.47 % y es dominado por el 3.44 % de las soluciones. Estos resultados, una vez más, son semejantes a los que obtuvo el MOPSOSS v1.

Gráficamente podemos notar (ver figura 7.3) que las dos propuestas que realizamos (MOPSOSS v1 y MOPSOSS v2) se acercan más al verdadero frente de Pareto que los otros 2 algoritmos.

El porcentaje de efectividad para el MOPSOSS v1 es del 83 % mientras que para el MOPSOSS v2 es del 82 %.

Métrica	Algoritmo	Media	Mejor	Peor	Des. Estándar	Mediana
DGI	MOPSOSS v.1	0.006310	0.000999	0.032753	0.008661	0.004404
	MOPSOSS v.2	0.005231	0.000886	0.024573	0.006408	0.005068
	SPEA2	0.024139	0.013275	0.046733	0.010513	0.023956
	NSGA-II	0.022366	0.015249	0.047607	0.006438	0.022100
D	MOPSOSS v.1	0.419970	0.266854	1.000000	0.171484	0.416093
	MOPSOSS v.2	0.398447	0.254967	0.793154	0.110873	0.399111
	SPEA2	0.756452	0.535991	1.000000	0.139502	0.758064
	NSGA-II	0.721361	0.560834	1.043846	0.110210	0.723506

Tabla 7.6: Resultados de las métricas DGI y D para la función ZDT2

C	MOPSOSS v.1	MOPSOSS v.2	SPEA2	NSGA-II	Dominan
MOPSOSS v.1	———	0.103255	0.833460	0.921912	61.95 %
MOPSOSS v.2	0.301456	———	0.773371	0.889465	65.47 %
SPEA2	0.000000	0.000000	———	0.645189	21.50 %
NSGA-II	0.000000	0.000000	0.092821	———	3.09 %
Son Dominados	10.04 %	3.44 %	56.65 %	81.88 %	———

Tabla 7.7: Resultados de la métrica de cobertura para la función ZDT2

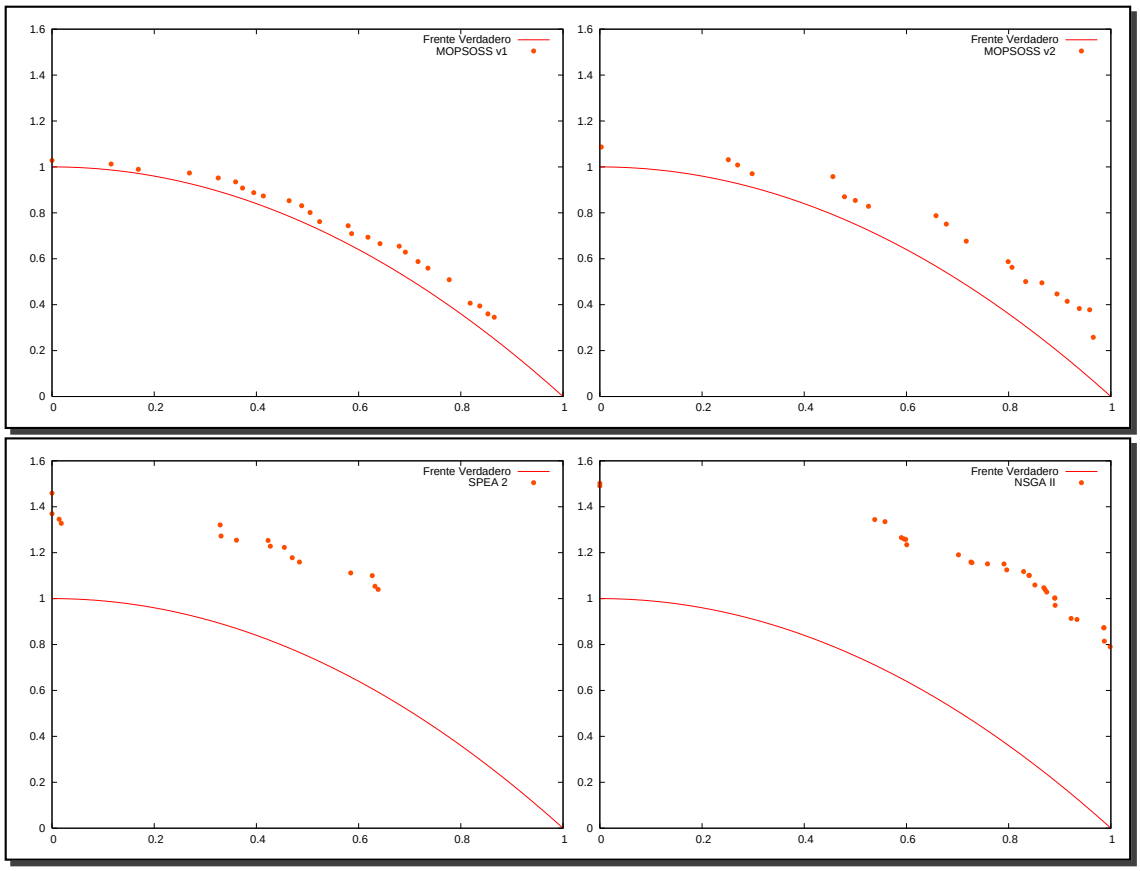


Figura 7.3: Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función ZDT2

7.2.4 Función ZDT3

En la tabla 7.8 vemos que nuestra propuesta MOPSOSS v2 es mejor en 3 de los 5 criterios adoptados en la métrica DGI. En cuanto a la distribución nuestro esquema MOPSOSS v2 gana en 4 de los 5 criterios, con excepción de la desviación estándar. Sin embargo en la métrica de cobertura observamos que nuestra propuesta en promedio domina al 91.02 % y es dominado por el 1.84 %. Lo que muestra un mejor comportamiento que los demás algoritmos.

En la figura 7.4 vemos que nuestra propuesta MOPSOSS v2 se acerca más que los otros algoritmos e incluso es capaz de encontrar algunos puntos del verdadero frente de Pareto en la corrida correspondiente a la mediana.

El porcentaje de efectividad para el MOPSOSS v1 es del 64 % mientras que para el MOPSOSS v2 es del 78 %.

Métrica	Algoritmo	Media	Mejor	Peor	Des. Estándar	Mediana
DGI	MOPSOSS v.1	0.054282	0.023741	0.108657	0.017922	0.052979
	MOPSOSS v.2	0.006417	0.002035	0.022034	0.003864	0.006419
	SPEA2	0.011707	0.009823	0.014001	0.001138	0.011742
	NSGA-II	0.015514	0.011695	0.020015	0.002024	0.015453
D	MOPSOSS v.1	0.770705	0.660748	0.883693	0.053017	0.770547
	MOPSOSS v.2	0.646252	0.556406	0.809878	0.063519	0.650965
	SPEA2	0.789491	0.717838	0.923188	0.047798	0.790142
	NSGA-II	0.749287	0.681987	0.825549	0.035358	0.749000

Tabla 7.8: Resultados de las métricas DGI y D para la función ZDT3

C	MOPSOSS v.1	MOPSOSS v.2	SPEA2	NSGA-II	Dominan
MOPSOSS v.1	——	0.002052	0.000000	0.001818	0.74 %
MOPSOSS v.2	0.974427	——	0.855196	0.901110	91.02 %
SPEA2	1.000000	0.041723	——	0.930539	65.71 %
NSGA-II	0.995579	0.011749	0.032508	——	34.65 %
Son Dominados	98.99 %	1.84 %	29.58 %	61.11 %	——

Tabla 7.9: Resultados de la métrica de cobertura para la función ZDT3

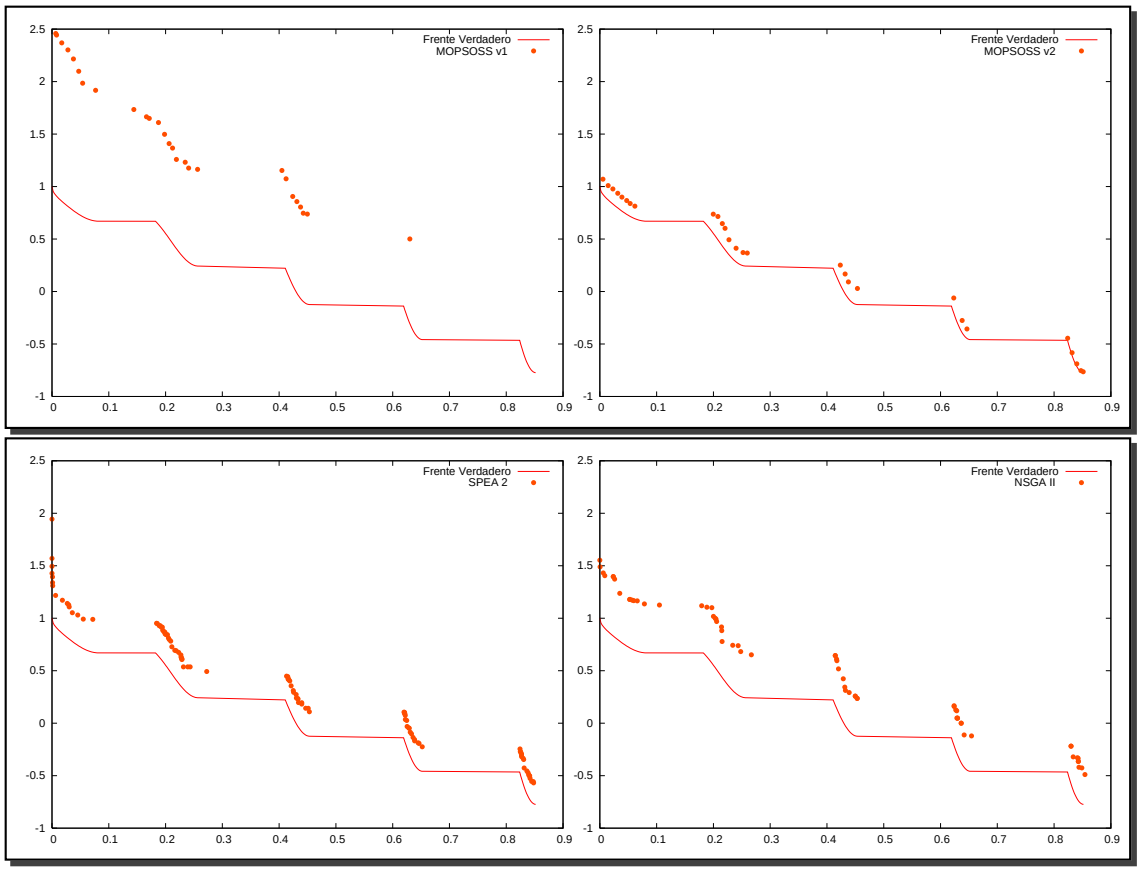


Figura 7.4: Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función ZDT3

7.2.5 Función ZDT4

En la tabla 7.10 observamos que nuestras propuestas MOPSOSS v1 y MOPSOSS v2 tienen resultados similares. Sin embargo, MOPSOSS v2 es marginalmente mejor para la métrica DGI así como para la métrica de distribución. En cuanto a la métrica de cobertura, MOPSOSS v2 obtuvo el mejor resultado dominando en promedio 37.94 % y fue dominado en el 11.06 %.

La función ZDT4 es quizás la función más difícil del conjunto de pruebas ZDTs adoptado, por lo que con el número de evaluaciones definido, MOPSOSS v2 se acerca más rápido que los demás algoritmos al frente de Pareto como lo podemos observar en la figura 7.5.

El porcentaje de efectividad para el MOPSOSS v1 es del 56 % mientras que para el MOPSOSS v2 es del 62 %.

Métrica	Algoritmo	Media	Mejor	Peor	Des. Estándar	Mediana
DGI	MOPSOSS v.1	0.123785	0.027767	0.245527	0.061951	0.122481
	MOPSOSS v.2	0.106260	0.037401	0.217377	0.042089	0.103348
	SPEA2	0.447195	0.284478	0.665949	0.100032	0.451893
	NSGA-II	0.429756	0.196690	0.792868	0.130436	0.449456
D	MOPSOSS v.1	0.708651	0.488364	0.907290	0.103044	0.707851
	MOPSOSS v.2	0.699258	0.558554	0.844436	0.077166	0.699954
	SPEA2	0.978972	0.901893	1.000000	0.034082	0.985019
	NSGA-II	0.984066	0.933141	1.002513	0.019319	0.980628

Tabla 7.10: Resultados de las métricas DGI y D para la función ZDT4

C	MOPSOSS v.1	MOPSOSS v.2	SPEA2	NSGA-II	Dominan
MOPSOSS v.1	——	0.331922	0.242365	0.241501	27.19 %
MOPSOSS v.2	0.658257	——	0.242365	0.241501	37.94 %
SPEA2	0.011333	0.000000	——	0.833289	38.06 %
NSGA-II	0.000000	0.000000	0.077186	——	2.57 %
Son Dominados	22.31 %	11.06 %	18.72 %	43.87 %	——

Tabla 7.11: Resultados de la métrica de cobertura para la función ZDT4

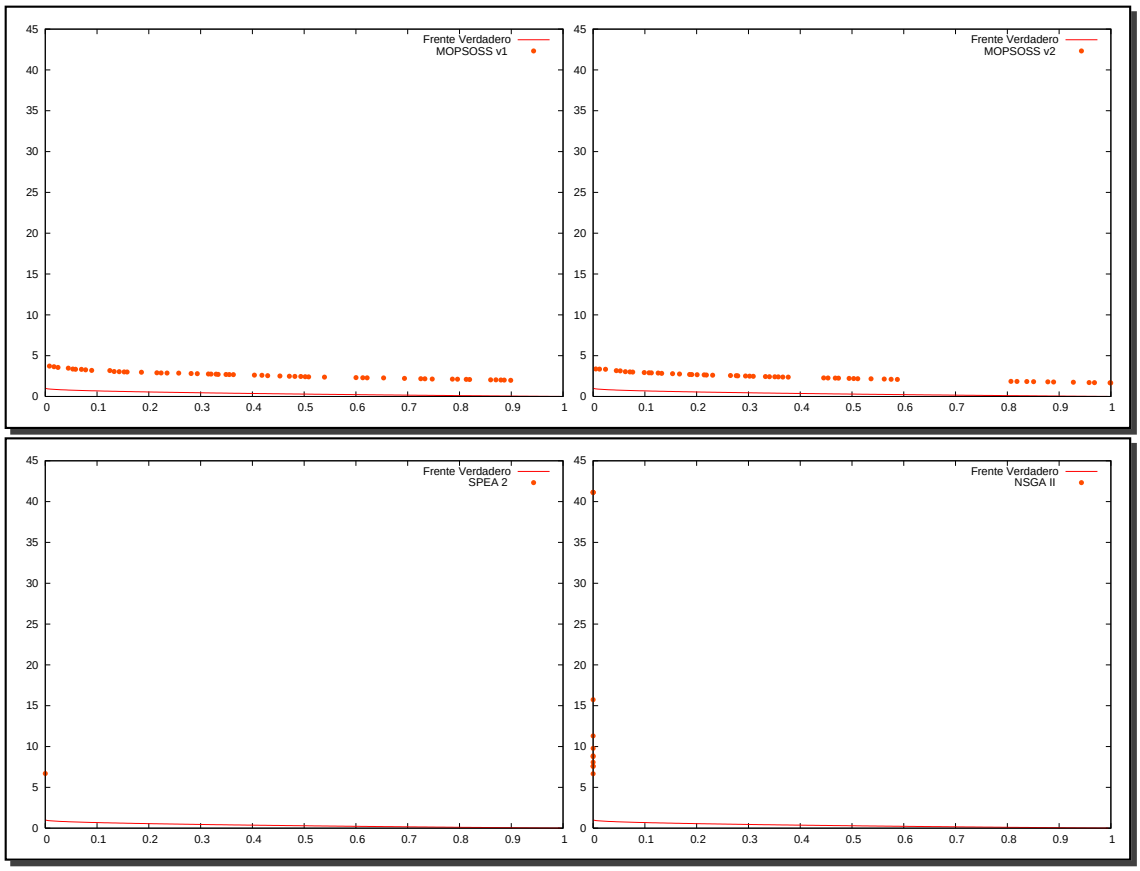


Figura 7.5: Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función ZDT4

7.2.6 Función ZDT6

Para la función ZDT6 en la métrica DGI MOPSOSS v2 mostró ser mejor en todos los criterios utilizados. En cuanto a la dispersión, MOPSOSS v2 sólo es mejor en 3 de los 5 criterios. El SPEA2 posee el mejor resultado en lo que corresponde a la peor corrida y una mejor desviación estándar con respecto a la dispersión. En cuanto a la métrica de cobertura, MOPSOSS v2 mostró ser mejor dominando en promedio al 67.65 % de las soluciones y tan sólo fue dominado por el 1.94 % siendo notoriamente superior a los demás algoritmos.

En la figura 7.6 podemos corroborar lo anterior observando que nuestra propuesta MOPSOSS v2 se aproxima más rápido al verdadero frente de Pareto en la corrida correspondiente a la media.

El porcentaje de efectividad para el MOPSOSS v1 es del 41 % mientras que para el MOPSOSS v2 es del 77 %.

Métrica	Algoritmo	Media	Mejor	Peor	Des. Estándar	Mediana
DGI	MOPSOSS v.1	0.004351	0.001212	0.006510	0.001294	0.004240
	MOPSOSS v.2	0.001021	0.000429	0.003052	0.000575	0.000999
	SPEA2	0.030126	0.026141	0.035226	0.002482	0.030025
	NSGA-II	0.042071	0.032913	0.049729	0.004102	0.041350
D	MOPSOSS v.1	0.572103	0.412958	1.011084	0.151638	0.572823
	MOPSOSS v.2	0.426199	0.226490	1.048401	0.155724	0.432013
	SPEA2	0.785340	0.633627	0.901921	0.062847	0.789782
	NSGA-II	0.874745	0.704567	1.000000	0.077845	0.873403

Tabla 7.12: Resultados de las métricas DGI y D para la función ZDT6

C	MOPSOSS v.1	MOPSOSS v.2	SPEA2	NSGA-II	Dominan
MOPSOSS v.1	———	0.058255	0.620076	0.546102	40.81 %
MOPSOSS v.2	0.872116	———	0.620076	0.537542	67.65 %
SPEA2	0.000000	0.000000	———	0.993915	33.13 %
NSGA-II	0.000000	0.000000	0.002564	———	0.08 %
Son Dominados	29.07 %	1.94 %	41.41 %	69.25 %	———

Tabla 7.13: Resultados de la métrica de cobertura para la función ZDT6

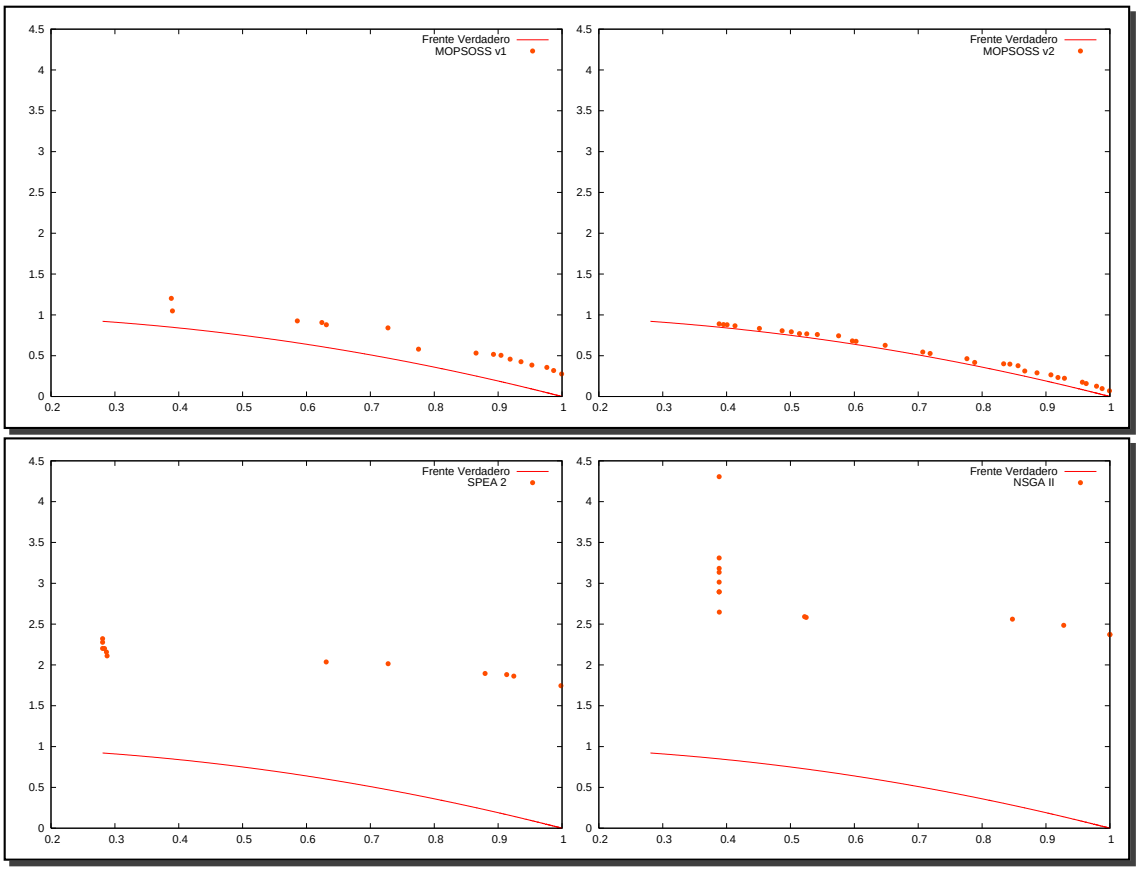


Figura 7.6: Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función ZDT6

7.2.7 Función DTLZ1

La función DTLZ1 es la primera de las 4 funciones de prueba de 3 objetivos que utilizaremos en esta tesis.

En la tabla 7.14 vemos que tanto en la métrica DGI como en la dispersión es mejor nuestro algoritmo MOPSOSS v2. En la tabla 7.15 vemos que en la métrica de cobertura nuestra propuesta MOPSOSS v2 domina en promedio al 59.14 % de las soluciones y es dominado por el 10.67 %.

En la figura 7.7 vemos que las soluciones se encuentran tan lejanas del verdadero frente de Pareto que este ni siquiera alcanza a verse (ver Apéndice B). Aún así, nuestra propuesta MOPSOSS v2 se acerca más rápido al frente de Pareto que los otros algoritmos.

El porcentaje de efectividad para el MOPSOSS v1 es del 51 % mientras que para el MOPSOSS v2 es del 63 %.

Métrica	Algoritmo	Media	Mejor	Peor	Des. Estándar	Mediana
DGI	MOPSOSS v.1	0.527411	0.286171	0.775936	0.141230	0.541021
	MOPSOSS v.2	0.394229	0.241844	0.658855	0.118162	0.407520
	SPEA2	0.686993	0.302719	1.013826	0.187351	0.691259
	NSGA-II	0.731898	0.337627	1.253174	0.206284	0.729736
D	MOPSOSS v.1	0.994811	0.988378	1.009364	0.004467	0.994824
	MOPSOSS v.2	0.993969	0.985480	1.002286	0.004006	0.993447
	SPEA2	0.998157	0.996791	0.999123	0.000510	0.998113
	NSGA-II	0.998136	0.996359	0.999561	0.000810	0.998020

Tabla 7.14: Resultados de las métricas DGI y D para la función DTLZ1

C	MOPSOSS v.1	MOPSOSS v.2	SPEA2	NSGA-II	Dominan
MOPSOSS v.1	———	0.195496	0.657763	0.665837	50.63 %
MOPSOSS v.2	0.428127	———	0.663665	0.682625	59.14 %
SPEA2	0.120557	0.069679	———	0.299719	16.32 %
NSGA-II	0.093340	0.055108	0.439898	———	19.60 %
Son Dominados	21.39 %	10.67 %	58.70 %	54.93 %	———

Tabla 7.15: Resultados de la métrica de cobertura para la función DTLZ1

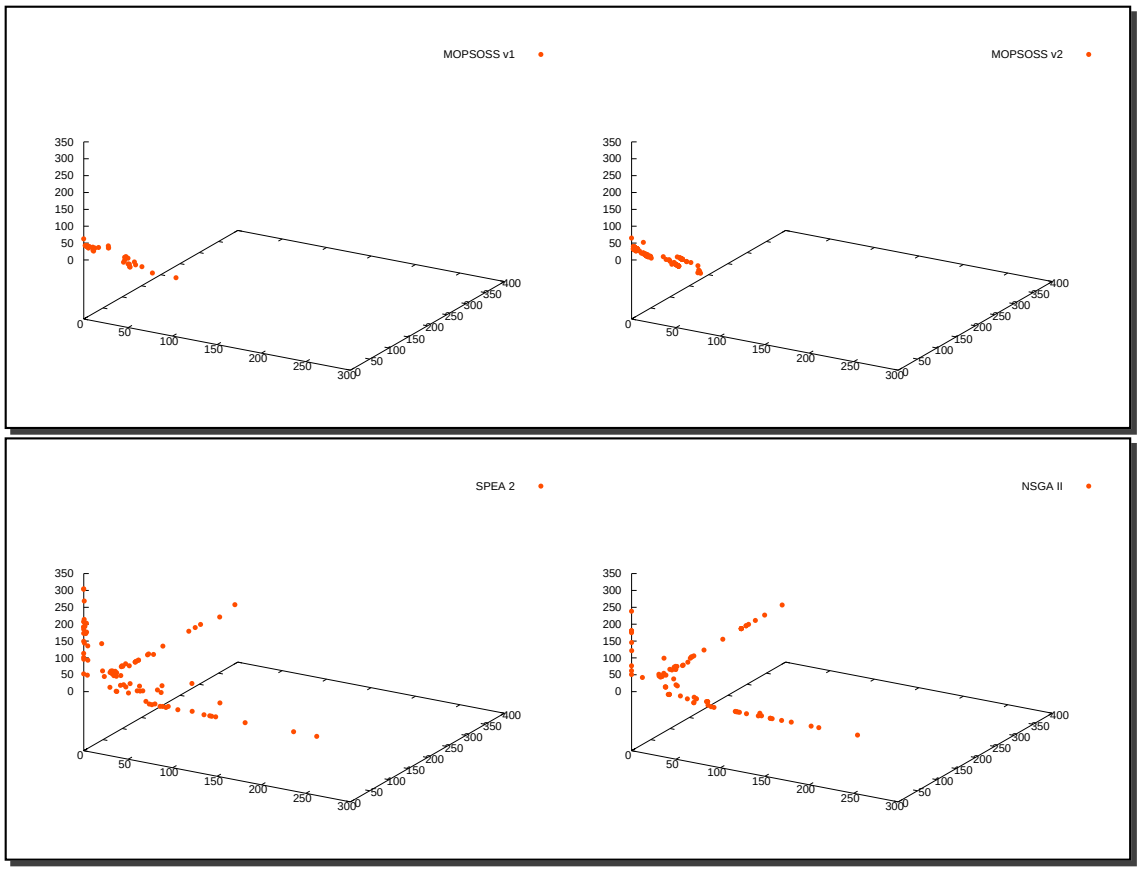


Figura 7.7: Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función DTLZ1

7.2.8 Función DTLZ2

En la tabla 7.16 vemos que para la métrica DGI el SPEA2 es mejor en todos los criterios adoptados. Sin embargo, el NSGA-II presenta una mejor dispersión de las soluciones. En la tabla 7.17 observamos que en cuanto a la cobertura, al algoritmo que le va mejor es a nuestra propuesta MOPSOSS v2 dominando en promedio al 27.29 % y siendo dominado por el 7.46 %

En la figura 7.8 podemos ver que los 4 algoritmos llegan sin problemas al verdadero frente de Pareto lo cual podemos constatar en el Apéndice B. Sin embargo, en cuanto a distribución de las soluciones vemos que el SPEA2 obtiene un mejor frente de Pareto en la corrida correspondiente a la mediana.

El porcentaje de efectividad para el MOPSOSS v1 es del 61 % mientras que para el MOPSOSS v2 es del 74 %.

Métrica	Algoritmo	Media	Mejor	Peor	Des. Estándar	Mediana
DGI	MOPSOSS v.1	0.000593	0.000416	0.000872	0.000128	0.000590
	MOPSOSS v.2	0.000635	0.000431	0.001216	0.000185	0.000638
	SPEA2	0.000360	0.000323	0.000384	0.000014	0.000360
	NSGA-II	0.000424	0.000382	0.000498	0.000031	0.000424
D	MOPSOSS v.1	0.647575	0.568676	0.764825	0.054528	0.643580
	MOPSOSS v.2	0.760046	0.597533	0.894202	0.068670	0.760009
	SPEA2	0.446576	0.371231	0.567421	0.047186	0.446347
	NSGA-II	0.215535	0.188645	0.279647	0.020232	0.213748

Tabla 7.16: Resultados de las métricas DGI y D para la función DTLZ2

C	MOPSOSS v.1	MOPSOSS v.2	SPEA2	NSGA-II	Dominan
MOPSOSS v.1	——	0.088648	0.160418	0.091000	11.33 %
MOPSOSS v.2	0.349529	——	0.288754	0.180667	27.29 %
SPEA2	0.268992	0.069686	——	0.051333	12.99 %
NSGA-II	0.228866	0.065657	0.087013	——	12.71 %
Son Dominados	28.24 %	7.46 %	17.87 %	10.76 %	——

Tabla 7.17: Resultados de la métrica de cobertura para la función DTLZ2

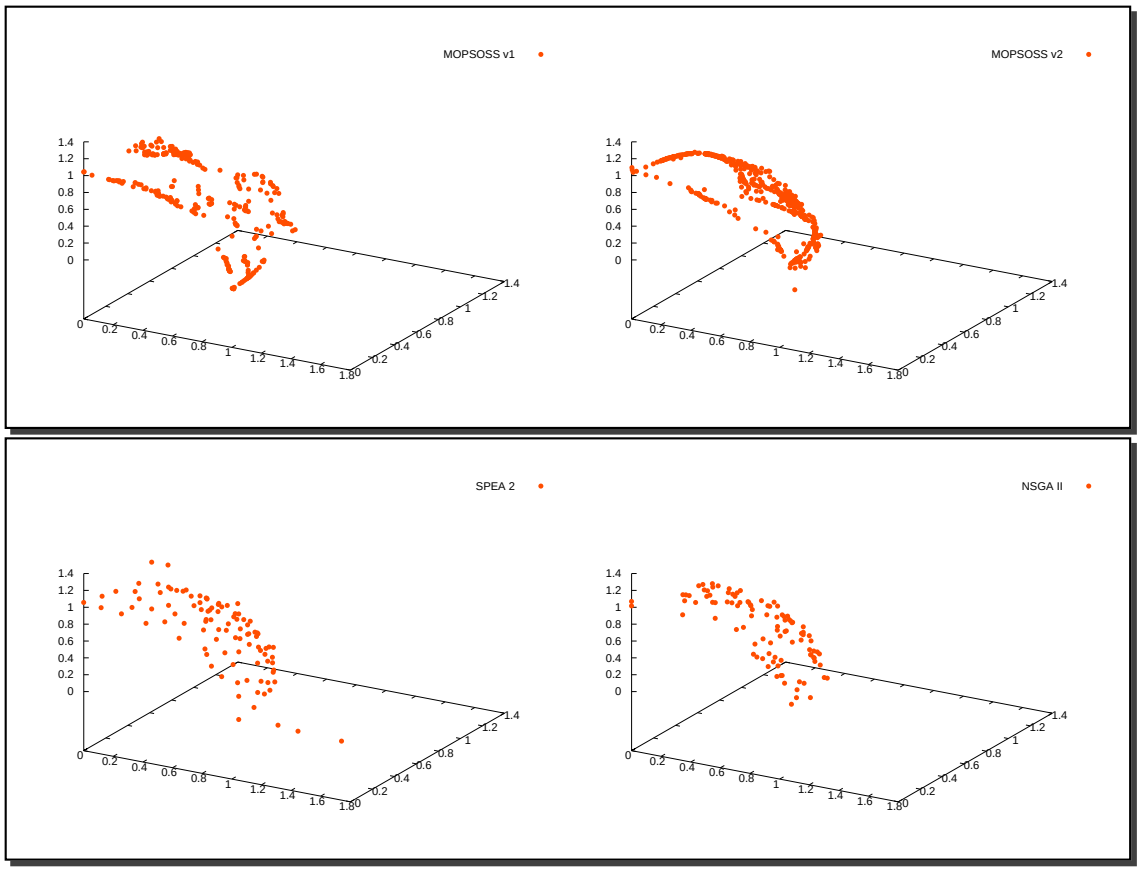


Figura 7.8: Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función DTLZ2

7.2.9 Función DTLZ3

El la tabla 7.18 vemos que en la métrica DGI nuestra propuesta MOPSOSS v2 gana en todos los criterios adoptados. Para la métrica de distribución, MOPSOSS v2 gana en 3 de los 5 criterios adoptados, obtiene los mejores resultados en la media y en la mediana así como en la mejor corrida. En cuanto a la cobertura se refiere, como se muestra en la tabla 7.19, observamos que el MOPSOSS v2 domina en promedio al 44.52 % de las soluciones y es dominado por el 8.92 % de las soluciones.

En la figura 7.9 podemos observar que el MOPSOSS v2 se acerca más rápido que los demás algoritmos al verdadero frente de Pareto.

El porcentaje de efectividad para el MOPSOSS v1 es del 54 % mientras que para el MOPSOSS v2 es del 47 %.

Métrica	Algoritmo	Media	Mejor	Peor	Des. Estándar	Mediana
DGI	MOPSOSS v.1	0.958198	0.500484	1.498925	0.267526	0.961696
	MOPSOSS v.2	0.834905	0.273790	1.323319	0.234381	0.824312
	SPEA2	1.473229	0.771439	1.949341	0.284647	1.471541
	NSGA-II	1.422879	0.810014	2.185355	0.269040	1.426260
D	MOPSOSS v.1	0.996722	0.990753	1.001049	0.002883	0.996829
	MOPSOSS v.2	0.996468	0.987014	1.002165	0.002995	0.996543
	SPEA2	0.999184	0.998473	0.999584	0.000292	0.999172
	NSGA-II	0.999241	0.998637	0.999636	0.000251	0.999228

Tabla 7.18: Resultados de las métricas DGI y D para la función DTLZ3

C	MOPSOSS v.1	MOPSOSS v.2	SPEA2	NSGA-II	Dominan
MOPSOSS v.1	——	0.242243	0.442398	0.468419	38.43 %
MOPSOSS v.2	0.342565	——	0.476700	0.516568	44.52 %
SPEA2	0.057788	0.021817	——	0.283965	12.11 %
NSGA-II	0.093870	0.003646	0.469664	——	18.90 %
Son Dominados	16.46 %	8.92 %	46.28 %	42.29 %	——

Tabla 7.19: Resultados de la métrica de cobertura para la función DTLZ3

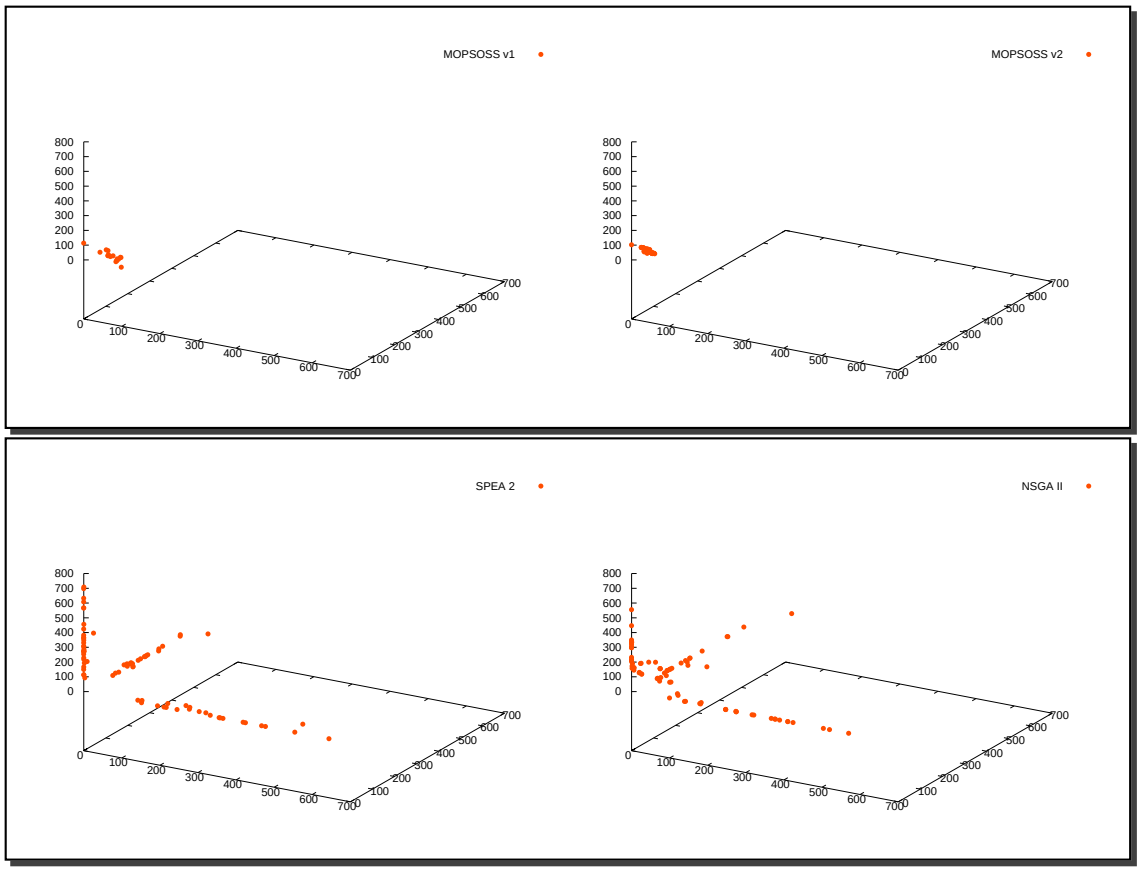


Figura 7.9: Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función DTLZ3

7.2.10 Función DTLZ4

Pare la fnución de prueba DTLZ4, en la tabla 7.20 vemos que el SPEA2 gana en 4 de los 5 criterios para la métrica DGI mientras que el NSGA II gana en el criterio restante. En la métrica de distribución tenemos que el SPEA2 es mejor en todos los criterios. En la tabla 7.21 observamos que el SPEA2 resultó ser muy superior a los demás algoritmos dominando en promedio al 86.55 % y no fue dominado por ninguna solución.

En la figura 7.10 podemos confirmar los resultados obtenidos, el SPEA2 llega al frente de Pareto en las gráficas correspondientes a las medianas, mientras que los otros algoritmos se quedan más lejos.

El porcentaje de efectividad para el MOPSOSS v1 es del 64 % mientras que para el MOPSOSS v2 es del 39 %.

Métrica	Algoritmo	Media	Mejor	Peor	Des. Estándar	Mediana
DGI	MOPSOSS v.1	0.026293	0.020050	0.037300	0.004800	0.026443
	MOPSOSS v.2	0.022155	0.012062	0.028557	0.004298	0.021654
	SPEA2	0.004709	0.001082	0.010026	0.004329	0.001272
	NSGA-II	0.009695	0.005443	0.014339	0.002568	0.009784
D	MOPSOSS v.1	0.800506	0.634688	1.000000	0.144785	0.786777
	MOPSOSS v.2	0.734538	0.558397	1.000000	0.116815	0.747598
	SPEA2	0.426918	0.346352	0.559859	0.045831	0.429730
	NSGA-II	0.725140	0.528815	0.971702	0.089823	0.721240

Tabla 7.20: Resultados de las métricas DGI y D para la función DTLZ4

C	MOPSOSS v.1	MOPSOSS v.2	SPEA2	NSGA-II	Dominan
MOPSOSS v.1	——	0.128452	0.000000	0.026274	5.1 %
MOPSOSS v.2	0.324122	——	0.000000	0.081590	13.52 %
SPEA2	0.897371	0.892922	——	0.806436	86.55 %
NSGA-II	0.519367	0.432912	0.000000	——	31.74 %
Son Dominados	58.02 %	48.47 %	0.00 %	30.47 %	——

Tabla 7.21: Resultados de la métrica de cobertura para la función DTLZ4

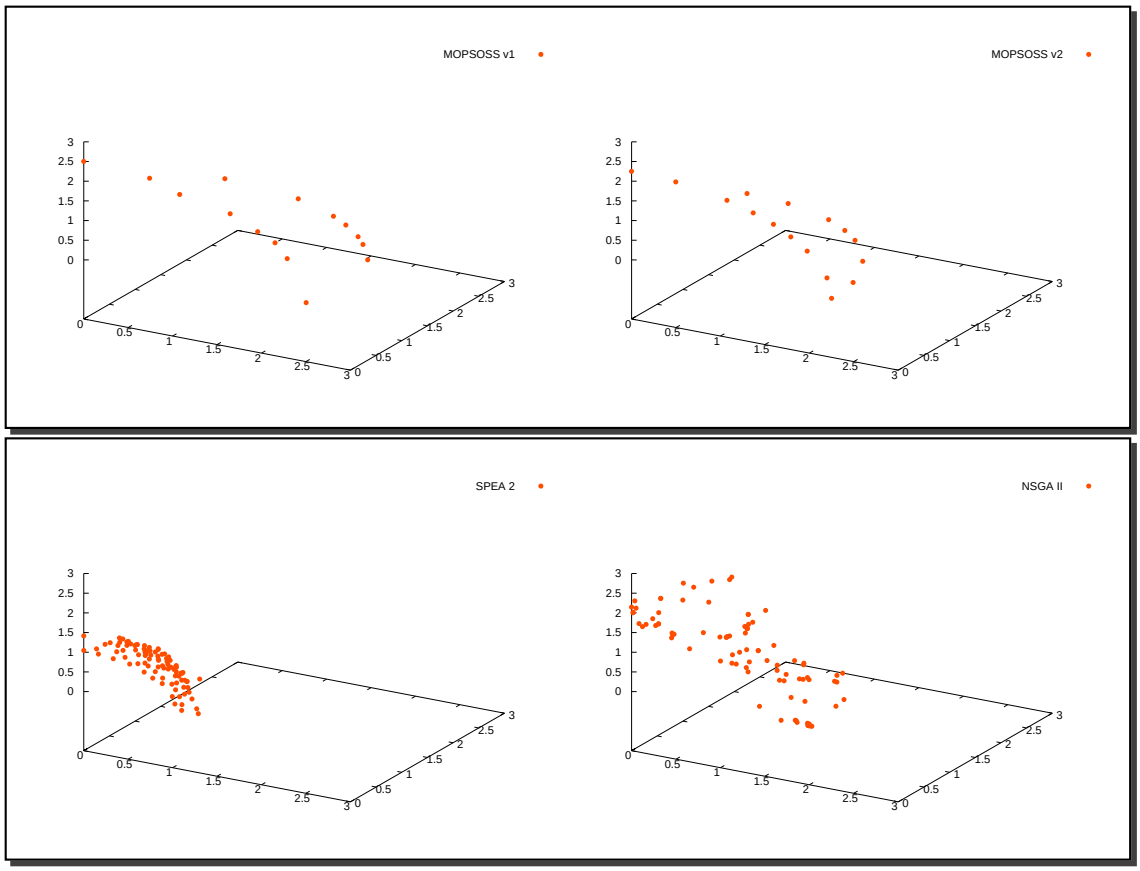


Figura 7.10: Frente de Pareto de la mediana generado por MOPSOSS v1, MOPSOSS v2, SPEA2 y NSGA-II para la función DTLZ4

7.3 Conclusiones sobre los resultados obtenidos

Nuestra propuesta MOPSOSS v2 resultó ser mejor que el MOPSOSS v1, al menos en el conjunto de pruebas adoptado. El MOPSOSS v2 mostró ser contundente para el conjunto de funciones ZDTs así como para las funciones DTLZ1 y DTLZ3, y en las funciones de KURSAWE y DTLZ2 mostró ser competitivo ganando en el menos una métrica. La única función para la cual no pudimos competir fue la DTLZ4. En general MOPSOSS v2 fue mejor que el NSGA II así como el SPEA2 para la mayoría de los problemas de prueba presentados (7 de los 10 problemas).

Nuestra propuesta MOPSOSS v2 resultó ser muy competitiva con respecto a los algoritmos SPEA2 y el NSGA II. En la tesis realizamos $4,000$ evaluaciones ya que los algoritmos contra los que realizamos las pruebas suelen tener una rápida convergencia y decidimos competir en cuanto a la velocidad de convergencia se refiere.

En términos generales, nuestra propuesta MOPSOSS v2 resultó ser un algoritmo consistente y muy competitivo sobre el conjunto de funciones adoptado en nuestro estudio. Las características de las funciones de prueba adoptadas se describen en el Apéndice A. Las gráficas de convergencia de nuestro MOPSOSS v2 se muestran en el Apéndice B.

8

Conclusiones y Trabajo Futuro

8.1 Conclusiones

Con el análisis de resultados realizado en esta tesis podemos concluir que hemos diseñado un algoritmo de optimización multiobjetivo *robusto* con mecanismos novedosos y sobre todo funcionales (MOPSOSS v2), que si bien en algunos problemas de prueba no supera a los algoritmos más importantes del estado del arte como el NSGA-II y el SPEA2, sí lo hace en la mayoría de los problemas estudiados en esta tesis. A continuación se mencionan los puntos más relevantes que podemos concluir de este trabajo de tesis:

1. El PSO tiene un mejor comportamiento utilizando una topología global que una topología local.

2. Es más eficiente hacer búsquedas en diferentes objetivos a la vez que optimizar sobre uno solo de ellos.

3. PSO es una técnica de optimización difícil de controlar debido a los parámetros de vuelo, pero una vez que se logra tener una mejor comprensión del papel que juegan sus parámetros, presenta una convergencia muy acelerada.

4. La forma en que dirigimos las soluciones hacia los óptimos en la versión MOPSOSS v2 nos da muy buenos resultados. Para los problemas de tres funciones objetivo, la partícula ideal juega un papel muy importante. Suelen encontrarse mejores resultados a través de la partícula ideal que usando las partículas que optimizan sobre cada función objetivo. Se intuye en primera instancia que esto se debe a que le damos un mayor peso, dedicando más partículas en la dirección de la partícula ideal.

5. Un mecanismo que no se enfatiza demasiado, pero que es muy importante, es el cómo reposicionar a la partícula una vez que ha excedido los límites de los valores de las variables; el mecanismo que utilizamos en el PSO es una recombinación BLX-α entre la partícula líder y la partícula personal. Esto evita irnos a los extremos de las variables haciendo que el algoritmo tenga resultados tendenciosos si es que las soluciones se encuentran en los límites.

6. La mutación es un mecanismo imprescindible dentro del PSO debido a que provee diversidad dentro de la población.

7. La búsqueda dispersa, a pesar de ser una técnica poco utilizada en el área, nos dio muy buenos resultados gracias a sus mecanismos de búsqueda local que nos ayudan a dirigirnos a óptimos globales.

8. Se modificó el esquema original de la búsqueda dispersa, a fin de aprovechar los conceptos de dispersión y utilizarlos de una forma diferente pero, a nuestra consideración, más eficiente.

9. La búsqueda dispersa es un mecanismo que vale la pena investigar más y en torno al cual se requieren más propuestas ya sus conceptos básicos son muy buenos y flexibles.

8.2 Trabajo Futuro

Consideramos que existen mecanismos de nuestro algoritmo que se podrían diseñar de forma que fuesen más eficientes:

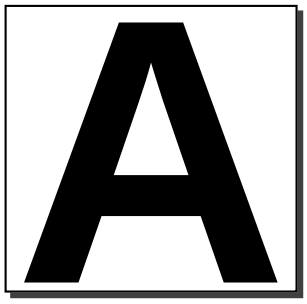
1. Un mecanismo que ajuste los parámetros del PSO de manera automática.

2. Crear un mecanismo de selección para manejar los casos en los que el límite de la población se ha alcanzado y tenemos un exceso de soluciones no dominadas. En esos casos, se puede ayudar a la distribución de soluciones a lo largo del frente de Pareto, a través de los mecanismos del filtrado de la población secundaria.

3. Analizar los diferentes mecanismos de recombinación existentes y validar con cuál se comporta mejor nuestro algoritmo.

4. En el esquema de la búsqueda dispersa analizar otros tipos de búsqueda local y recombinación.

5. Agregar un mecanismo para el manejo de restricciones.



Funciones de prueba

A.1 Función de Kursawe

Esta función fue propuesta por Kursawe [56] y consiste en minimizar:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x})] \tag{A.1}$$

donde:

$$f_1(\vec{x}) = \sum_{i=1}^{n-1} \left[-10 \exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2} \right) \right]$$

$$f_2(\vec{x}) = \sum_{i=1}^{n} \left[|x_i|^{0.8} + 5\sin(x_i)^3 \right]$$

para $n = 3$ y $x_i \in [-5, 5]$.

Este problema tiene un frente de Pareto desconectado y el conjunto de óptimos de Pareto tambíen es desconectado. Posee 4 regiones desconectadas.

A.2 Función ZDT1

Esta función fue propuesta por Zitzler [92] y consiste en minimizar:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x})] \tag{A.2}$$

donde:

$$f_1(\vec{x}) = x_1$$
$$f_2(\vec{x}) = g(\vec{x}) \cdot h(f_1, g)$$
$$h(f_1, g) = 1 - \sqrt{f_1/g(\vec{x})}$$
$$g(\vec{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$

para $n = 30$ y $x_i \in [0, 1]$. Tiene un frente de Pareto convexo y conectado.

A.3 Función ZDT2

Esta función fue propuesta por Zitzler [92] y consiste en minimizar:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x})] \tag{A.3}$$

donde:

$$f_1(\vec{x}) = x_1$$
$$f_2(\vec{x}) = g(\vec{x}) \cdot h(f_1, g)$$
$$g(\vec{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$
$$h(f_1, g) = 1 - [f_1/g(\vec{x})]^2$$

para $n = 30$ y $x_i \in [0, 1]$. Posee un frente de Pareto no convexo y conectado.

A.4 Función ZDT3

Esta función fue propuesta por Zitzler [92] y consiste en minimizar:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x})] \tag{A.4}$$

donde:

$$f_1(\vec{x}) = x_1$$
$$f_2(\vec{x}) = g(\vec{x}) \cdot h(f_1, g)$$
$$h(f_1, g) = 1 - \sqrt{f_1/g(\vec{x})} - [f_1/g(\vec{x})] \sin(10\pi f_1)$$
$$g(\vec{x}) = 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i$$

para $n = 30$ y $x_i \in [0, 1]$. Tiene un frente de Pareto discontinuo.

A.5 Función ZDT4

Esta función fue propuesta por Zitzler [92] y consiste en minimizar:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x})] \tag{A.5}$$

donde:

$$f_1(\vec{x}) = x_1$$
$$f_2(\vec{x}) = g(\vec{x}) \cdot h(f_1, g)$$

$$h(f_1, g) = 1 - \sqrt{f_1/g(\vec{x})}$$
$$g(\vec{x}) = 1 + 10(n-1) + \sum_{i=2}^{n} \left[x_i^2 - 10\cos(4\pi x_i)\right]$$

para $n = 10$, $x_1 \in [0, 1]$ y $x_i \in [-5, 5]$. Posee un frente de Pareto convexo, conectado y existen 21^9 frentes de Pareto locales.

A.6 Función ZDT6

Esta función fue propuesta por Zitzler [92] y consiste en minimizar:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x})] \tag{A.6}$$

donde:

$$f_1(\vec{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$$
$$f_2(\vec{x}) = g(\vec{x}) \cdot h(f_1, g)$$
$$h(f_1, g) = 1 - [f_1/g(\vec{x})]^2$$
$$g(\vec{x}) = 1 + 9\left\{\left[(\sum_{i=2}^{10} x_i)/9\right]^{0.25}\right\}$$

para $n = 10$ y $x_i \in [0, 1]$. Posee un frente de Pareto no covexo y continuo.

A.7 Función DTLZ1

Esta función fue propuesta por Deb [29] y consiste en minimizar:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x})] \tag{A.7}$$

donde:

$$f_1(\vec{x}) = \frac{1}{2}x_1x_2\,[1 + g(\vec{x})]$$
$$f_2(\vec{x}) = \frac{1}{2}x_1(1 - x_2)\,[1 + g(\vec{x})]$$
$$f_3(\vec{x}) = \frac{1}{2}(1 - x_1)\,[1 + g(\vec{x})]$$
$$g(\vec{x}) = 100\left\{12 + \sum_{i=3}^{12}(x_i - 0.5)^2 - \cos\left[20\pi(x_i - 0.5)\right]\right\}$$

para $n = 12$ y $x_i \in [0, 1]$. Tiene un frente de Pareto lineal y contiene $11^n - 1$ frentes de Pareto locales.

A.8 Función DTLZ2

Esta función fue propuesta por Deb [29] y consiste en minimizar:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x})] \tag{A.8}$$

donde:

$$f_1(\vec{x}) = \cos\left(\frac{\pi}{2}x_1\right)\cos\left(\frac{\pi}{2}x_2\right)[1+g(\vec{x})]$$
$$f_2(\vec{x}) = \cos\left(\frac{\pi}{2}x_1\right)\sin\left(\frac{\pi}{2}x_2\right)[1+g(\vec{x})]$$
$$f_3(\vec{x}) = \sin\left(\frac{\pi}{2}x_1\right)[1+g(\vec{x})]$$
$$g(\vec{x}) = \sum_{i=3}^{12}(x_i - 0.5)^2$$

para $n = 12$ y $x_i \in [0, 1]$. Tiene un frente de Pareto convexo y conectado.

A.9 Función DTLZ3

Esta función fue propuesta por Deb [29] y consiste en minimizar:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x})] \tag{A.9}$$

donde:

$$f_1(\vec{x}) = \cos\left(\frac{\pi}{2}x_1\right)\cos\left(\frac{\pi}{2}x_2\right)[1+g(\vec{x})]$$
$$f_2(\vec{x}) = \cos\left(\frac{\pi}{2}x_1\right)\sin\left(\frac{\pi}{2}x_2\right)[1+g(\vec{x})]$$
$$f_3(\vec{x}) = \sin\left(\frac{\pi}{2}x_1\right)[1+g(\vec{x})]$$
$$g(\vec{x}) = 100\left\{12 + \sum_{i=3}^{12}(x_i - 0.5)^2 - \cos\left[20\pi(x_i - 0.5)\right]\right\}$$

para $n = 12$ y $x_i \in [0, 1]$. Posee un frente de Pareto no convexo y contiene $3^n - 1$ frentes de Pareto locales.

A.10 Función DTLZ4

Esta función fue propuesta por Deb [29] y consiste en minimizar:

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x})] \tag{A.10}$$

donde:

$$f_1(\vec{x}) = \cos\left(\frac{\pi}{2}x_1^\alpha\right)\cos\left(\frac{\pi}{2}x_2^\alpha\right)[1 + g(\vec{x})]$$

$$f_2(\vec{x}) = \cos\left(\frac{\pi}{2}x_1^\alpha\right)\sin\left(\frac{\pi}{2}x_2^\alpha\right)[1 + g(\vec{x})]$$

$$f_3(\vec{x}) = \sin\left(\frac{\pi}{2}x_1^\alpha\right)[1 + g(\vec{x})]$$

$$g(\vec{x}) = \sum_{i=1}^{12}(x_i - 0.5)^2$$

para $n = 12$, $\alpha = 100$ y $x_i \in [0, 1]$. Tiene un frente de Pareto convexo y conectado.

B

Convergencia del MOPSOSS en las Funciones de Prueba Adoptadas

En este apéndice presentamos las gráficas de nuestra propuesta MOPSOSS v2 con la finalidad de validar la convergencia de nuestro algoritmo al frente de Pareto verdadero en un determinado número de evaluaciones de la función objetivo.

Para las funciones de prueba de Kursawe, ZDT1, ZDT2, ZDT3, ZDT6 y DTLZ2 realizando tan solo $7,000$ evaluaciones nuestra propuesta converge al frente de Pareto verdadero, mientras que para la función ZDT4 se necesita realizar al menos $30,000$ evaluaciones. Las funciones que presentaron mayor dificultad son las DTLZ1, DTLZ3, DTLZ4 en las cuales se necesitaron $150,000$ evaluaciones de la función objetivo.

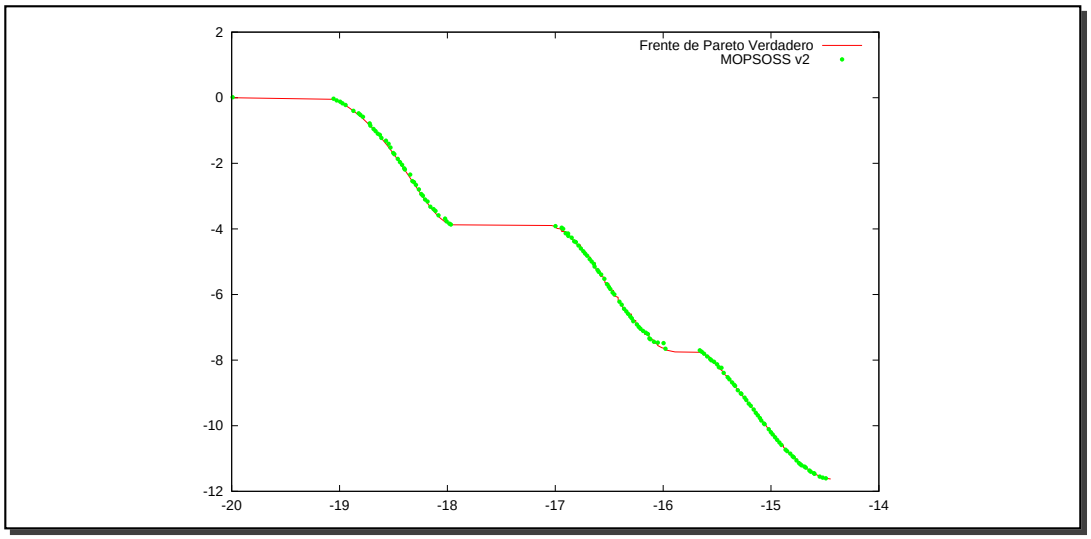


Figura B.1: Gráfica de convergencia del algoritmo MOPSOSS v2 en la función de Kursawe (realizando 7,000 evaluaciones de la función objetivo).

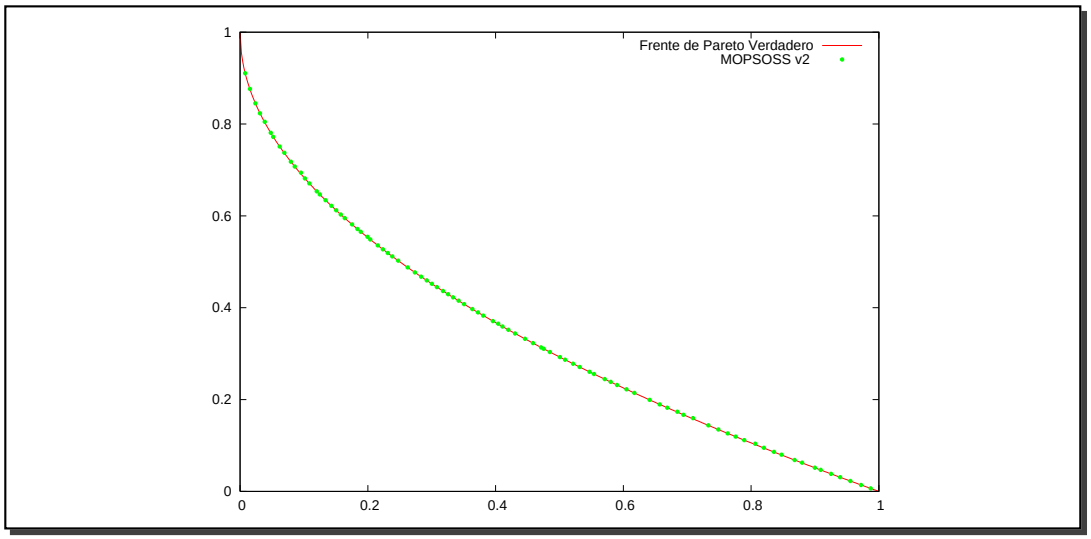


Figura B.2: Gráfica de convergencia del algoritmo MOPSOSS v2 en la función ZDT1 (realizando 7,000 evaluaciones de la función objetivo).

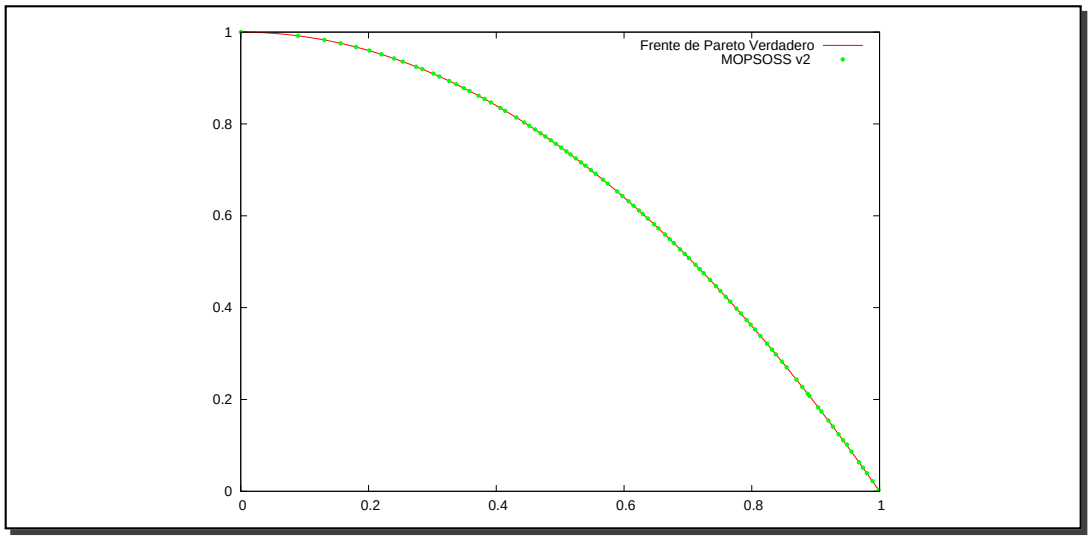


Figura B.3: Gráfica de convergencia del algoritmo MOPSOSS v2 en la función ZDT2 (realizando 7,000 evaluaciones de la función objetivo).

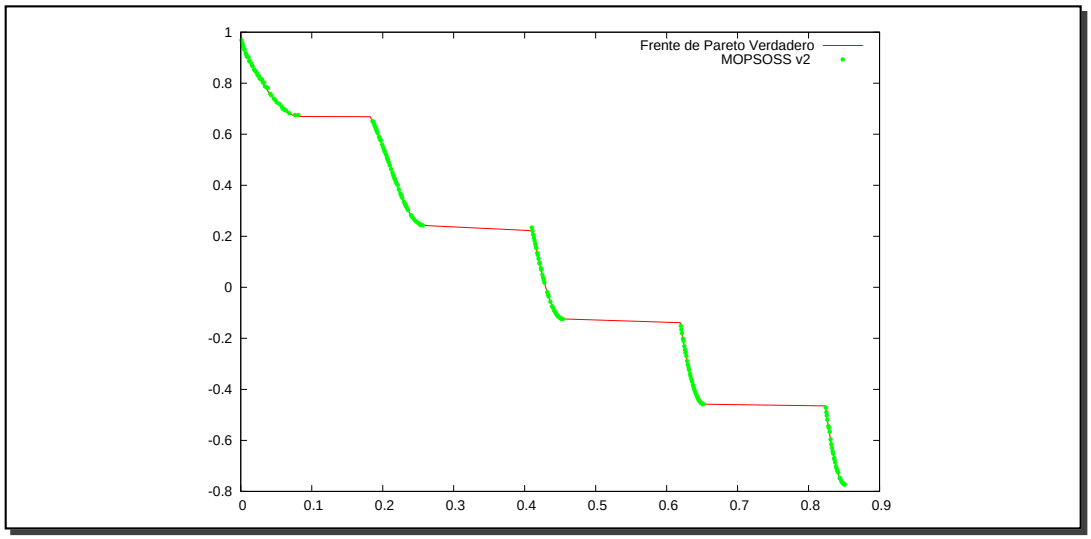


Figura B.4: Gráfica de convergencia del algoritmo MOPSOSS v2 en la función ZDT3 (realizando 7,000 evaluaciones de la función objetivo).

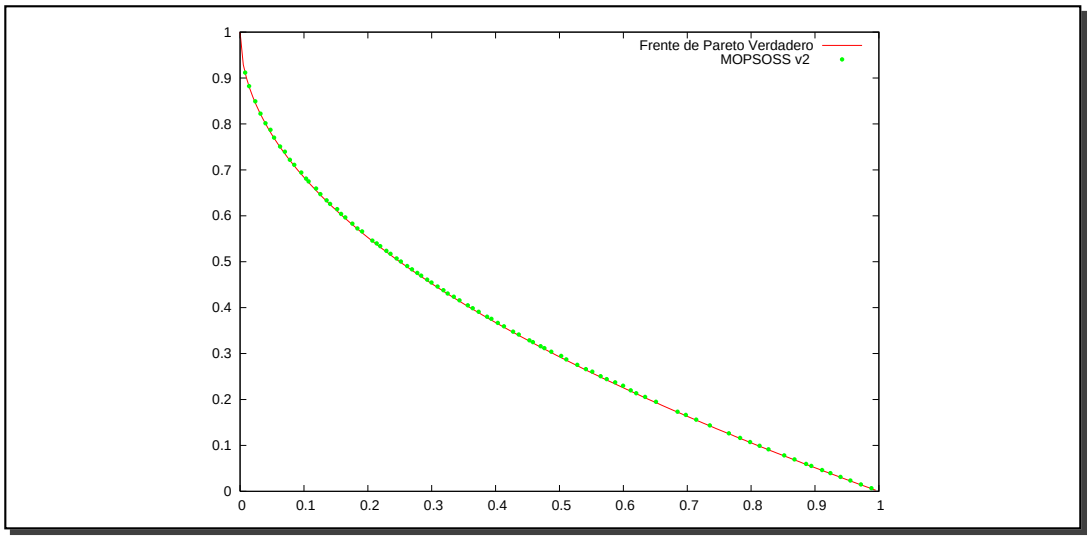


Figura B.5: Gráfica de convergencia del algoritmo MOPSOSS v2 en la función ZDT4 (realizando 30,000 evaluaciones de la función objetivo).

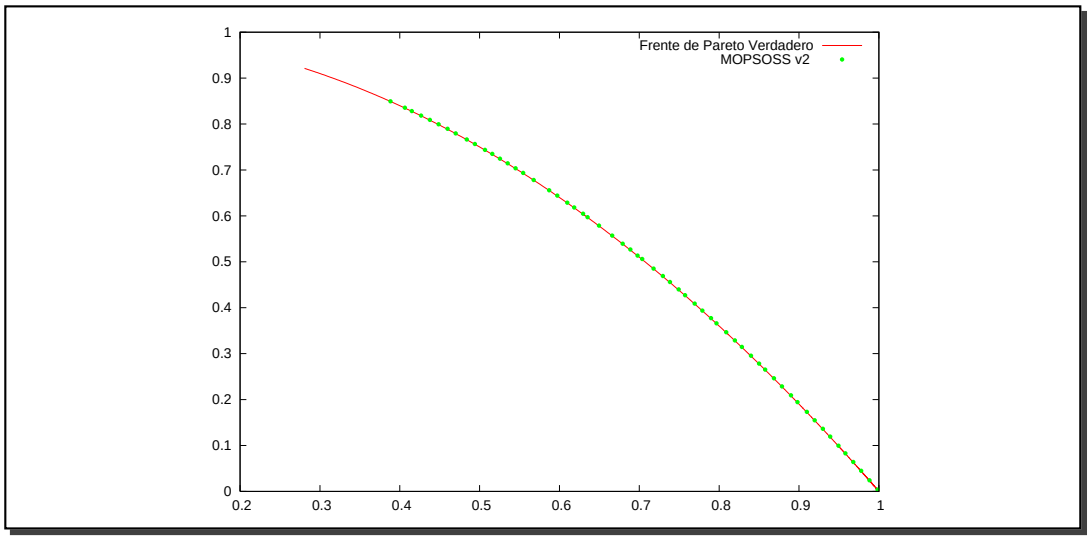


Figura B.6: Gráfica de convergencia del algoritmo MOPSOSS v2 en la función ZDT6 (realizando 7,000 evaluaciones de la función objetivo).

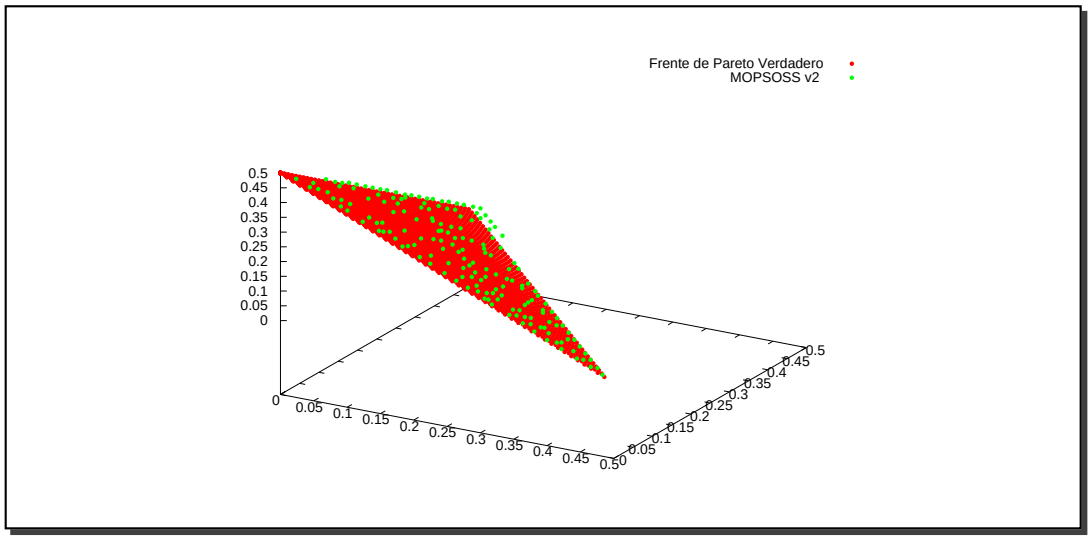


Figura B.7: Gráfica de convergencia del algoritmo MOPSOSS v2 en la función DTLZ1 (realizando 150,000 evaluaciones de la función objetivo).

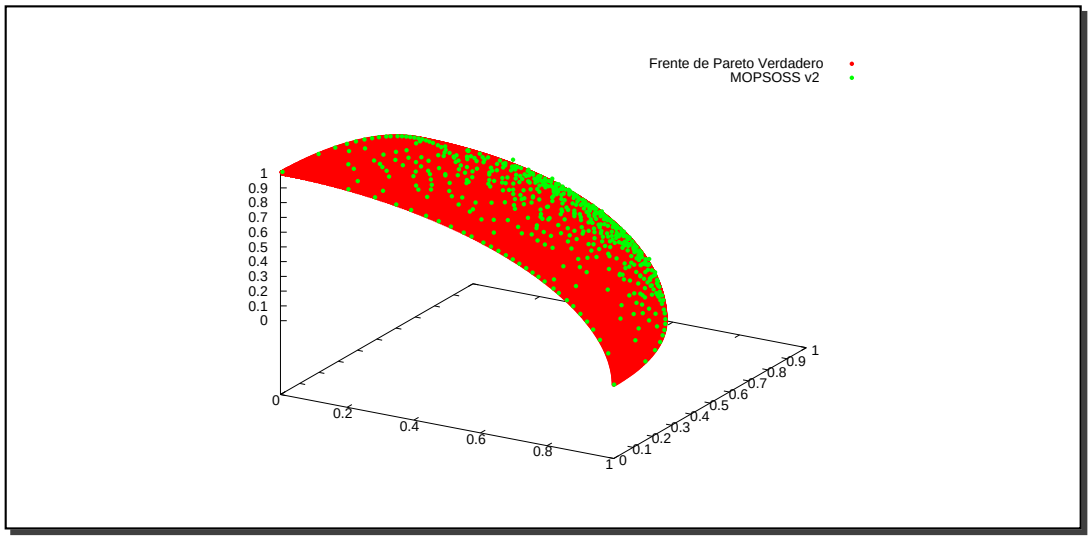


Figura B.8: Gráfica de convergencia del algoritmo MOPSOSS v2 en la función DTLZ2 (realizando 7,000 evaluaciones de la función objetivo).

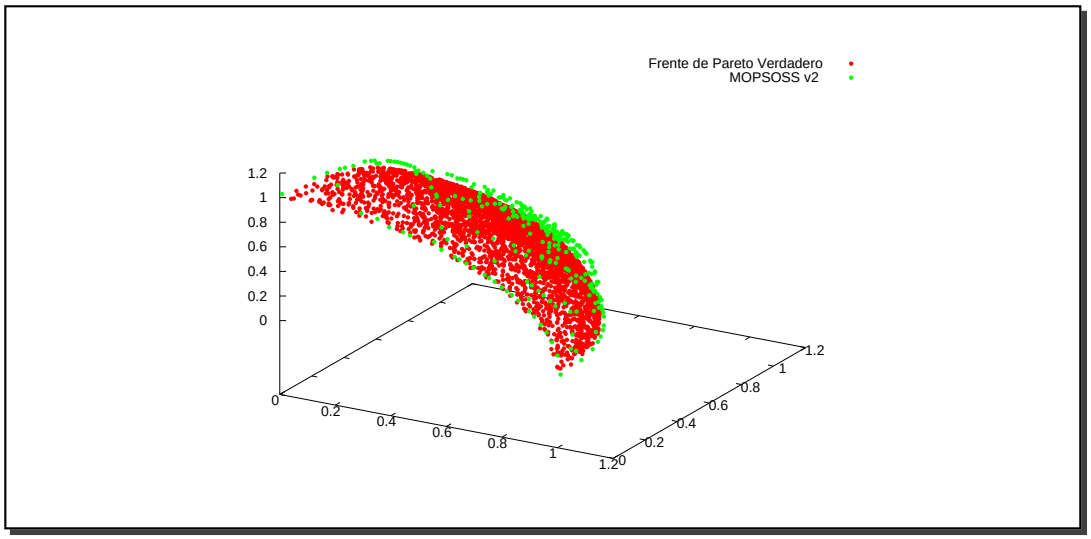


Figura B.9: Gráfica de convergencia del algoritmo MOPSOSS v2 en la función DTLZ3 (realizando 150,000 evaluaciones de la función objetivo).

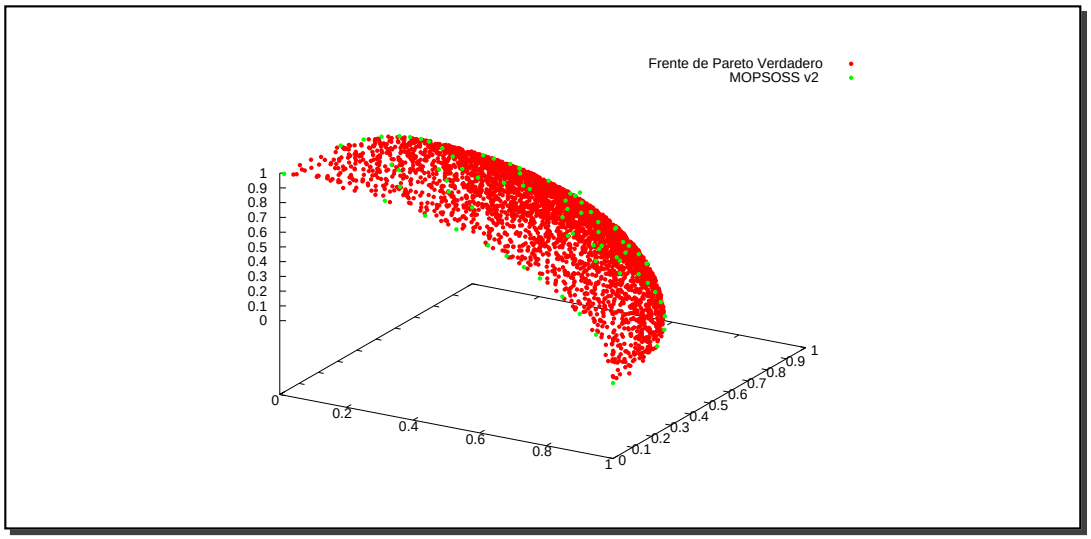


Figura B.10: Gráfica de convergencia del algoritmo MOPSOSS v2 en la función DTLZ4 (realizando 150,000 evaluaciones de la función objetivo).

Bibliografía

[1] Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors. *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*. Springer, USA, 2005. ISBN 1-85233-787-7.

[2] Julio E. Alvarez-Benitez, Richard M. Everson, and Jonathan E. Fieldsend. A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 459–473, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.

[3] Robert Axelrod. The dissemination of culture: A model with local convergence and global polarization. *Journal of Conflict Resolution*, 41:203–226, 1997.

[4] Albert Bandura. *Social Foundations of Thought and Action: A Social Cognitive Theory*. Prentice Hall, Englewood Cliffs, NJ, 1986.

[5] Thomas Bartz-Beielstein, Philipp Limbourg, Konstantinos E. Parsopoulos, Michael N. Vrahatis, Jörn Mehnen, and Karlheinz Schmitt. Particle Swarm Optimizers for Pareto Optimization with Enhanced Archiving Techniques. In *Proceedings of the 2003 Congress on Evolutionary*

Computation (CEC'2003), volume 3, pages 1780–1787, Canberra, Australia, December 2003. IEEE Press.

[6] Ulrike Baumgartner, Christian Magele, and Werner Renhart. Pareto Optimality and Particle Swarm Optimization. *IEEE Transactions on Magnetics*, 40(2):1172–1175, March 2004.

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

[8] Ricardo P. Beausoleil Delgado. MOSS: Multiobjective scatter search applied to non-linear multiple criteria optimization. *European Journal of Operational Research*, 169(2):426–449, March 2006.

[9] Arthur W. Burks. Computation, behavior and structure in fixed and growing automata. In M. C. Yovits and S. Cameron, editors, *Self-Organizing Systems*, pages 282–309, New York, 1960. Pergamon Press.

[10] Chan-Jin Chung and Robert G. Reynolds. Caep: An evolution-based tool for real-valued function optimization using cultural algorithms. *International Journal on Artificial Intelligence Tools*, 7(2):239–291, 1998.

[11] Carlos A. Coello Coello. A Short Tutorial on Evolutionary Multiobjective Optimization. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 21–40. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[12] Carlos A. Coello Coello, Alan D. Christiansen, and Arturo Hernández Aguirre. Using a New GA-Based Multiobjective Optimization Technique for the Design of Robot Arms. *Robotica*, 16(4):401–414, July–August 1998.

[13] Carlos A. Coello Coello and Ricardo Landa Becerra. Adding Knowledge and Efficient Data Structures to Evolutionary Programming: A Cultural Algorithm for Constrained Optimization. In W.B. Langdon, E.Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A.C. Schultz, J. F. Miller, E. Burke, and N.Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 201–209, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[14] Carlos A. Coello Coello and Maximino Salazar Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1051–1056, Piscataway, New Jersey, May 2002. IEEE Service Center.

[15] Carlos A. Coello Coello and Gregorio Toscano Pulido. Multiobjective Optimization using a Micro-Genetic Algorithm. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 274–282, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[16] Carlos A. Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga. Handling Multiple Objectives With Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, June 2004.

[17] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.

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

[19] Charles Robert Darwin. *On the Origin of Species by Means of Natural Selection Or the Preservation of Favoured Races in the Struggle for Life*. Cambridge University Press, Cambridge, UK, 1964. Originally published in 1859.

[20] Indraneel Das and John Dennis. A closer look at drabacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1):63–69, 1995.

[21] Dipankar Dasgupta, editor. *Immune Systems and Their Applications*. Springer-Verlag, Berlin, 1999.

[22] Kalyanmoy Deb. GeneAS: A Robust Optimal Design Technique for Mechanical Component Design. In Dipankar Dasgupta and Zbigniew Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*, pages 497–514, Springer-Verlag, Berlin, 1997.

[23] Kalyanmoy Deb. Solving Goal Programming Problems Using Multi-Objective Genetic Algorithms. In *1999 Congress on Evolutionary Computation*, pages 77–84, Washington, D.C., July 1999. IEEE Service Center.

[24] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.

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

[26] Kalyanmoy Deb and Amarendra Kumar. Real-coded Genetic Algorithms with Simulated Binary Crossover: Studies on Multimodal and Multiobjective Problems. *Complex Systems*, 9:431–454, 1995.

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

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

[29] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Test Problems for Evolutionary Multi-Objective Optimization. Technical Report 112, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2001.

[30] Kenneth A. DeJong. *Evolutionary Computation*. The MIT Press, Cambridge, Massachusetts, 2006.

[31] Marco Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.

[32] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. The MIT Press, July 2004.

[33] Lucien Duckstein. Multiobjective optimization in structural design: The model choice problem. In E. Atrek, R. H. Gallagher, K. M. Ragsdell, and O. C. Zienkiewicz, editors, *In New Directions in Optimum Structural Design*, pages 459–481, New York, NY, 1984. John Wiley and Sons, Inc.

[34] Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[35] Mark Erickson, Alex Mayer, and Jeffrey Horn. The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 681–695. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[36] Leon Festinger. A theory of social comparison processes. *Human Relations*, 7:117–140, 1954.

[37] David B. Fogel, editor. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*, New York, 1995. The Institute of Electrical and Electronic Engineers.

[38] David B. Fogel. An introduction to evolutionary computation. In David B. Fogel, editor, *Evolutionary Computation: the fossil record*, pages 1–2. IEEE Press, Piscataway, NJ, 1998.

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

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

[41] Fred Glover. Heuristic for integer programming using surrogate constraints. *Decision Sciences 8*, pages 156–166, 1977.

[42] Fred Glover. Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Applied Mathematics*, 49(1-3):231–255, 1994.

[43] Fred Glover. A template for scatter search and path relinking. In *AE '97: Selected Papers from the Third European Conference on Artificial Evolution*, pages 13–54, London, UK, 1998. Springer-Verlag.

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

[45] Prabhat Hajela and Chu-Yung Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4:99–107, 1992.

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

[47] John H. Holland. Concerning efficient adaptive systems. *Self-Organizing Systems–1962*, pages 215–230, 1962.

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

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

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

[51] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, June 1994. IEEE Service Center.

[52] James Kennedy. Thinking is social: Experiments with the adaptive culture model. *Journal of Conflict Resolution*, 42:56–76, 1998.

[53] James Kennedy and Russell C. Eberhart. Paticle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume IV, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.

[54] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, California, USA, 2001.

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

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

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

[58] Konrad Lorenz. *Behind the Mirror: A Search for a Natural History of Human Knowledge*. Harcourt Brace Jovanovich, New York, 1973.

[59] Mahdi Mahfouf, Min-You Chen, and Derek Arturh Linkens. Adaptive Weighted Particle Swarm Optimisation for Multi-objective Optimal Design of Alloy Steels. In *Parallel Problem Solving from Nature - PPSN VIII*, pages 762–771, Birmingham, UK, September 2004. Springer-Verlag. Lecture Notes in Computer Science Vol. 3242.

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

[61] Julián Molina, Manuel Laguna, Rafael Martí, and Rafael Caballero. SSPMO: A Scatter Search Procedure for Nonlinear Multiobjective Optimization. volume 17, pages 111–122. Informs Journal of Computing, 2005.

[62] Antonio J. Nebro, Francisco Luna, and Enrique Alba. New ideas in applying scatter search to multiobjective optimization. In in Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler (editors), editors, *Third International Conference, EMO 2005*, volume 3410, pages 443–458, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science.

[63] Antonio J. Nebro, Francisco Luna, Enrique Alba, Andreas Beham, and Bernabé Dorronsoro. AbYSS: Adapting Scatter Search for Multiobjective Optimization. Technical Report ITI-2006-2, Dept. Lenguajes y Ciencias de la Computación, University of Málaga, Malaga, Spain, 2006.

[64] Masatoshi Nei. *Molecular Evolutionary Genetics*. Columbia University Press, New York, Guildford, 1987.

[65] John Nelder and Roger Mead. A Simplex Method for Function Minimization. *Computer Journal*, 7:308, 1965.

[66] Tatsuya Nomura and Katsunori Shimohara. An analysis of two-parent recombinations for real-valued chromosomes in an infinite population. *Evol. Comput.*, 9(3):283–308, 2001.

[67] Jacques Périaux, Mourad Sefrioui, and Bertrand Mantel. Ga multiple objective optimization strategies for electromagnetic backscattering. In D. Quagliarella, J. Périaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science. Recent Advences and Industrial Applications*, pages 225–243, West Sussex,England, 1997. John Wiley and Sons.

[68] Singiresu S. Rao. Game theory approach for multiobjective structural optimization. *Computer and Structures*, 25(1):119–127, 1987.

[69] Singiresu S. Rao. *Engineering Optimization: Theory and Practice*. John Wiley and Sons, New York, USA, 1996.

[70] Margarita Reyes Sierra and Carlos A. Coello Coello. Fitness Inheritance in Multi-Objective Particle Swarm Optimization. In *2005 IEEE Swarm Intelligence Symposium (SIS'05)*, pages 116–123, Pasadena, California, USA, June 2005. IEEE Press.

[71] Margarita Reyes Sierra and Carlos A. Coello Coello. Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 505–519, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.

[72] Robert G. Reynolds, Zbigniew Michalewicz, and M. Cavaretta. Using cultural algorithms for constraint handling in GENOCOP. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 298–305. MIT Press, Cambridge, Massachusetts, 1995.

[73] Günter Rudolph. Convergence analysis of canonical genetic algorithms. In *IEEE Transaction on Neural Networks*, volume 5, pages 96–101, January 1994.

[74] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[75] Luis Vicente Santana-Quintero and Carlos A. Coello Coello. An Algorithm Based on Differential Evolution for Multiobjective Problems. In Cihan H. Dagli, Anna L. Buczak, David L. Enke, Mark J. Embrechts, and Okan Ersoy, editors, *Smart Engineering System Design: Neural Networks, Evolutionary Programming and Artificial Life*, volume 15, pages 211–220, St. Louis, Missouri, USA, November 2005. ASME Press.

[76] J. David Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.

[77] O. G. Selfridge. Pandemonium: A paradigm for learning. In *Proceeedings of the Symposium on Mechanization of Thought Processes*, pages 511–529, Teddington,England, 1958.

[78] Hans-Paul Shwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhäuser, Basel, Alemania, 1977.

[79] Hans-Paul Shwefel. *Numerical Optimization of Computer Models*. Wiley, Chichester, UK, 1981.

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

[81] Dipti Srinivasan and Tian Hou Seow. Particle Swarm Inspired Evolutionary Algorithm (PS-EA) for Multi-Criteria Optimization Problems. In Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective Optimization: Theoretical Advances And Applications*, pages 147–165. Springer-Verlag, London, 2005. ISBN 1-85233-787-7.

[82] Michael Tomaselo. *The Cultural Origins of Human Cognition*. Harvard University Press, Cambridge, MA, 1999.

[83] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective Evolutionary Algorithm Research: A History and Analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.

[84] David A. Van Veldhuizen and Gary B. Lamont. On Measuring Multiobjective Evolutionary Algorithm Performance. In *2000 Congress on Evolutionary Computation*, volume 1, pages 204–211, Piscataway, New Jersey, July 2000. IEEE Service Center.

[85] August Weismann. *The Germ Plasm: A Theory of Heredity*. Scott, London, UK, 1893.

[86] Dietrich Wienke, Carlos Lucasius, and Gerrit Kateman. Multicriteria target optimization of analytical procedures using a genetic algorithm. *Analytica Chimica Acta*, 265(2):211–225, 1992.

[87] P. B. Wilson and M. D. Macleod. Low implementation cost IIR digital filter design using genetic algorithms. In *IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, pages 4/1–4/8, Chelmsford, U.K., 1993.

[88] Ricardo Salem Zebulum, Marco Aurlio Pacheco, and Marley Vellasco. A multi-objective optimisation methodology applied to the synthesis of low-power operational amplifiers. In Ivan Jorge Cheuri and Carlos Alberto dos Reis Filho, editors, *Proceedings of the XIII International Conference in Microelectronics and Packaging*, volume 1, pages 264–271, Curitiba, Brazil, August 1998.

[89] Milan Zeleny. Compromise programming. In J.L. Cochrane and M. Zeleny, editors, *Multiple Criteria Decision Making*, pages 262–301. University of South Carolina Press, Columbia, SC, 1973.

[90] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

[91] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.

[92] Eckart Zitzler, Marco Laumanns, Lothar Thiele, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Why Quality Assessment of Multiobjective Optimizers Is Difficult. In W.B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 666–673, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

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