INSTITUTO TECNOLÓGICO
DE CIUDAD MADERO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

DOCTORADO EN CIENCIAS DE LA INGENIERÍA

TESIS

FRAMEWORK PARA HIPERHEURÍSTICAS QUE INTEGREN PREFERENCIAS EN LA SOLUCIÓN DE PROBLEMAS A GRAN ESCALA, CON MUCHOS OBJETIVOS, E IMPRECISIÓN

Que para obtener el grado de:

DOCTOR EN CIENCIAS DE LA INGENIERÍA

Presenta:

M.C. José Manuel Vargas Martínez

D00070864

CVU 210275

Bajo la dirección de:

Director:

Dr. Nelson Rangel Valdez

CVU 48135

Co-Director:

Dr. Eduardo R. Fernández González

Ciudad Madero, Tamaulipas marzo de 2025.

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Ciudad Madero
Subdirección Académica
División de Estudios de Posgrado e Investigación

Ciudad Madero, Tamaulipas, 16/diciembre/2024

Oficio No. U.170/2024
ASUNTO: Autorización de impresión de tesis

C. JOSÉ MANUEL VARGAS MARTÍNEZ
No. DE CONTROL D00070864
P R E S E N T E

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su Examen de Grado de Doctorado en Ciencias de la Ingeniería, se acordó autorizar la impresión de su tesis titulada:

"FRAMEWORK PARA HIPERHEURÍSTICAS QUE INTEGREN PREFERENCIAS EN LA SOLUCIÓN DE PROBLEMAS A GRAN ESCALA, CON MUCHOS OBJETIVOS, E IMPRECISIÓN"

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTE:	DR.	NELSON RANGEL VALDEZ
SECRETARIA:	DRA.	MARÍA LUCILA MORALES RODRÍGUEZ
VOCAL 1:	DRA.	CLAUDIA GUADALUPE GOMÉZ SANTILLÁN
VOCAL 2:	DRA.	NOHRA VIOLETA GALLARDO RIVAS
VOCAL 3:	DR.	JUAN JAVIER GONZÁLEZ BARBOSA
SUPLENTE:	DR.	EDUARDO RENE FERNÁNDEZ GONZÁLEZ
DIRECTOR DE TESIS:	DR.	NELSON RANGEL VALDEZ
CO-DIRECTOR:	DR.	EDUARDO RENE FERNÁNDEZ GONZÁLEZ

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

A T E N T A M E N T E
Excelencia en Educación Tecnológica •
"Por Mi Patria Y Por Mi Bien" •

EDUCACIÓN SECRETARÍA DE EDUCACIÓN PÚBLICA | TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MARCO ANTONIO CORONEL GARCÍA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

ccp. Archivo
MACG/NRV

"POR MI PATRIA Y POR MI BIEN" · Norma Mexicana en Igualdad Laboral y No Discriminación · ESPACIO 100% LIBRE DE PLÁSTICO DE UN SOLO USO

DEDICATORIA

A mi esposa Sara Kandy por todo su apoyo, paciencia y amor, a mi hijo Manuel Eliseo que lo llevo en mi corazón todos los días.

AGRADECIMIENTOS

A dios que cada día me demuestra que está a mi lado y escucha mis oraciones.

A mi director de tesis el Dr. Nelson Rangel Valdez por su guía, consejos y confianza en estos 4 años de esta emocionante aventura llamada doctorado.

Al Dr. Eduardo R. Fernández por su apoyo en mis actividades académicas.

A mis compañeros Ana, Paco y Nelson que hicieron mis días muy agradables.

Al Consejo Nacional de Humanidades, Ciencias y Tecnologías por el apoyo brindado a través de la beca 788668, que alcanzó para cubrir mis alimentos y gastos adicionales y

al Tecnológico Nacional de México/Instituto Tecnológico de Ciudad Madero mi Alma Mater que siempre está muy presente en mí.

Framework para Hiperheurísticas que Integren Preferencias en la Solución de Problemas a Gran Escala, con Muchos Objetivos, e Imprecisión

José Manuel Vargas Martínez

Resumen

La optimización multi-objetivo sigue siendo un tema de investigación clave, especialmente en el ámbito de la optimización evolutiva multi-objetivo (EMO) y la toma de decisiones multicriterio (MCDM). Sin embargo, muchos problemas del mundo real presentan desafíos adicionales, como la presencia de múltiples objetivos (MaOPs), un gran número de variables de decisión y la incertidumbre en los datos. Esta investigación propone un framework para la construcción de hiperheurísticas que integren preferencias del decisor, con el objetivo de mejorar la eficiencia en la solución de problemas de optimización de gran escala y muchos objetivos bajo condiciones de imprecisión. El framework permite la creación modular y flexible de hiperheurísticas mediante la combinación de estrategias de selección basadas en preferencias, métodos de descomposición y heurísticas para manejar incertidumbre. Se construyen algoritmos de optimización robustos utilizando un conjunto de heurísticas de bajo nivel (LLHs) y operadores evolutivos, permitiendo una selección adaptativa basada en el desempeño de las soluciones. La validación del framework se realiza en el problema de selección de proyectos (PSP) y en problemas estándar como DTLZ y WFG. Los resultados experimentales muestran que la integración de preferencias mejora la convergencia hacia la región de interés (ROI), optimizando la satisfacción del decisor y reduciendo la complejidad computacional. Además, los resultados muestran que los algoritmos son capaces de manejar desafíos de escalabilidad como lo son la presencia de muchos objetivos y la gran escala. Este trabajo contribuye al desarrollo de hiperheurísticas avanzadas con un enfoque adaptable y extensible, con aplicaciones potenciales en áreas como la optimización energética y la planificación industrial.

Framework for Hyper-heuristics with Integration of Preferences Solving Large-Scale Problems with Many Objectives and Imprecision

Jose Manuel Vargas Martinez

Abstract

Multi-objective optimization remains a crucial research topic, particularly in evolutionary multi-objective optimization (EMO) and multi-criteria decision-making (MCDM). However, real-world problems often present additional challenges, such as handling many objectives (MaOPs), large-scale decision variables, and uncertainty. This research proposes a novel framework for hyper-heuristics that integrates decision-maker preferences to improve the efficiency of optimization processes in large-scale, many-objective, and imprecise environments. The framework enables the flexible and modular construction of hyper-heuristics, incorporating preference-based selection mechanisms, decomposition strategies, and heuristics for handling uncertainty. It facilitates the generation of robust optimization algorithms by leveraging a portfolio of low-level heuristics (LLHs) and evolutionary operators, allowing adaptive selection based on solution performance. The proposed framework is validated through the project selection problem (PSP) and benchmark problems such as DTLZ and WFG. The experimental results demonstrate that integrating preferences enhances the convergence toward the region of interest (ROI), improving decision-maker satisfaction while reducing computational complexity. In addition, the results show that the algorithms can handle scalability challenges such as the presence of many objectives and large-scale. This research contributes to hyper-heuristics by introducing a comprehensive, extensible, and adaptable framework capable of tackling complex real-world optimization problems. Future work includes extending the framework to other domains, such as energy optimization and large-scale industrial planning, where preference-based decision-making is crucial.

Índice General

Resumen V

Abstract VI

Índice Tablas XII

Índice de Figuras XV

1 Introducción 1

1.1 Antecedentes 4

1.2 Definición del Problema 4

1.3 Planteamiento del Problema 6

1.4 Objetivo General 7

1.5 Objetivos Específicos 7

1.6 Hipótesis 7

1.7 Justificación 8

2 Marco Teórico 11

2.1 Problemas de Optimización 11

2.1.1 Problemas de Optimización Multi-objetivo (MOPs) 11

2.1.2 Problemas de Optimización de Muchos Objetivos (MaOPs) 12

2.1.3 Problemas de Optimización Global de Gran Escala (LSGO) 12

2.2 Problema de Selección de Proyectos (PSP) 12

2.3 Estrategias de Búsqueda 14

2.3.1 Heurísticas 14

2.3.2 Metaheurísticas 15

2.3.3 Hiperheurísticas 16

2.4 Algoritmos Metaheurísticos 18
2.4.1 Recocido Simulado (Simulated Annealing) 18
2.4.2 Algoritmo de Evolución Diferencial 19
2.4.3 El Algoritmo Evolutivo Multi-objetivo Basado en Descomposición (MOEA/D). 22
2.4.4 Coevolución Cooperativa (CC) 23
2.5 Toma de Decisiones 26
2.5.1 Modelos de Preferencia 26
2.5.2 Conceptos Básicos de Outranking 27
2.5.3 Índice de Credibilidad: Determinando la Solidez de $a'Sa$ 30
2.5.4 Índice de Concordancia 30
2.5.5 Relaciones de Outranking 31
2.6 Conceptos de Imprecisión e Incertidumbre 32
2.6.1 Intervalos 34
2.7 Estado del Arte de los Frameworks de Hiperheurísticas 35
2.7.1 Descripción de los Frameworks de Hiperheurísticas 35
3 Metodología 39
3.1 Descripción de la Metodología 39
3.2 Framework de Software para Hiperheurísticas (Hyper-Heuristic Framework) 42
3.2.1 Arquitectura del Hyper-Heuristic Framework 44
3.3 Indicadores de Desempeño. 48
3.3.1 PCArea: Parallel Coordinate Area. 48

3.4 Heurísticas de Bajo Nivel (LLHs) desarrolladas. 50

3.4.1 MOSA/D - Recocido Simulado Multi-objetivo Basado en Descomposición 50

3.4.2 MOSA/D-O: Recocido Simulado Multi-objetivo Basado en Descomposición y Outranking 52

3.4.3 MOSA/D-O-II: Recocido Simulado Multi-objetivo Basado en Descomposición y Outranking II 53

3.4.4 CO-MOSA/D: Recocido Simulado Multi-objetivo Coevolutivo Basado en Descomposición 55

3.4.5 Conversión de Variables Continuas a Discretas para PSP 59

3.4.6 MOSA/D-PSP - Recocido Simulado Multi-objetivo Basado en Descomposición para PSP 61

3.4.7 MOSA/D-O-PSP y MOSA/D-O-II-PSP: Recocido Simulado Multi-objetivo Basado en Descomposición y Outranking para PSP 61

3.4.8 MOSA/D-O-II-PSP: Recocido Simulado Multi-objetivo Basado en Descomposición y Outranking II para PSP 62

3.4.9 CO-MOSA/D-PSP: Recocido Simulado Multi-objetivo Coevolutivo Basado en Descomposición para PSP. 63

3.5 Hiperheurísticas Desarrolladas 64

3.5.1 MaOLS-HH-UP: Many Objective Large-scale Hyper-heuristic with Uncertainty and Preferences 65

3.5.2 Hiperheurísticas Derivadas del Modelo MaOLS-HH-UP 67

4 Análisis y Resultados 69

4.1 Contexto de la Investigación 69

4.2 Diseño Experimental Propuesto 71

4.2.1 Configuración de los Algoritmos Hiperheurísticos (HHs) 73

4.2.2 Configuración de las Heurísticas de Bajo Nivel (LLHs) 75
4.2.3 Configuraciones de las Instancias de Prueba 77
4.3 Análisis de Resultados 78
4.3.1 Análisis de la Capacidad de los Algoritmos de Alcanzar el Frente de Pareto. 80
4.3.2 Análisis de la Capacidad de los Algoritmos para Aproximarse a la Hacia la Región de Interés (ROI) de un Decisor 89
5 Conclusiones y Recomendaciones 101
5.1 Conclusiones Finales 102
5.2 Trabajos Futuros 104
5.3 Publicaciones Derivadas del Presente Trabajo de Investigación 106
Glosario 107
Bibliografía 109
Anexos 118
A.1 Experimento MOSA/D: MOSA/D-CGO y MOSA/D-DE 119
A.1.1 Resumen de los Resultados 119
A.2 Experimento del Algoritmo Hiperheurístico MaOLS-HH-UP 124
5.3.1 Resumen de los Resultados 125
A.3 Experimentación de MOSA/D-O y MOSA/D-O-II 130
A.3.1 Análisis de Resultados: Cinco Objetivos 132
A.3.2 Análisis de Resultados: Diez Objetivos 132

B.1 Tablas y Datos de Prueba MOSA/D-DE y MOSA/D-CGO 134

B.2 Puntos de Referencia Utilizados en el Trabajo de Falcon-Cardona y Coello..... 135

B.3 Lista de Variantes de los Algoritmos: HH-DTLZ, HH-WFG y HH-PSP 135

Índice Tablas

Tabla 2.1. Ejemplo de la valoración de opciones de automóviles..... 33

Tabla 2.2. Se establece el valor de kj con un formato de intervalo. 34

Tabla 2.3. Tabla comparativa del estado del arte. 38

Tabla 3.1. LLHs enfocados en los problemas estándar DTLZ y WFG. 50

Tabla 3.2. LLHs enfocados en el problema de selección de proyectos (PSP). 50

Tabla 4.1. Diseño experimental para los problemas DTLZ 71

Tabla 4.2. Diseño experimental para los problemas WFG..... 72

Tabla 4.3. Diseño experimental para los problemas PSP..... 72

Tabla 4.4. Variables configurables de los hiperheurísticos. 73

Tabla 4.5. Métricas utilizadas en el proceso de selección de LLHs en los algoritmos HH-DTLZ, HH-WFG y HH-PSP. 74

Tabla 4.6. Variables configurables de MOSA/D, MOSA/D-PSP, CO-MOSA/D y CO-MOSA/D-PSP..... 75

Tabla 4.7. Variables configurables de MOSA/D-O, MOSA/D-O-PSP, MOSA/D-O-II y MOSA/D-O-II-PSP. 76

Tabla 4.8. Configuraciones de las instancias DTLZ (Instancias de muchos objetivos). 77

Tabla 4.9. Configuraciones de las instancias WFG (Instancias de muchos objetivos). 77

Tabla 4.10. Configuraciones de las instancias PSP (Instancias de gran escala)..... 78

Tabla 4.11. Métricas utilizadas para observar si los algoritmos se aproximan al PF. 79

Tabla 4.12. Métricas utilizadas por el análisis Borda para observar si los algoritmos se aproximan a la ROI representada por el mejor compromiso (BC). 79

Tabla 4.13. Resultados términos de HV de 9 hiperheurísticas para las 14 instancias de prueba DTLZ. 80

Tabla 4.14. Resumen de los mejores algoritmos por problema..... 82

Tabla 4.15. Resultados términos de HV de 9 hiperheurísticas para las 18 instancias de prueba WFG. 83

Tabla 4.16. Resumen de los mejores algoritmos por problema WFG. 85

Tabla 4.17. Resultados términos de HV de 10 hiperheurísticas para las 12 instancias de prueba de gran escala PSP. 86

Tabla 4.18. Resumen de los mejores algoritmos de la Tabla 4.17 para el problema PSP.... 87

Tabla 4.19. Resumen de los mejores algoritmos de la Tabla 4.17 para el problema PSP.... 87

Tabla 4.20. Mejores algoritmos por métrica de desempeño para el problema DTLZ con preferencia de un decisor para 5 objetivos. 90

Tabla 4.21. Mejores algoritmos por métrica de desempeño para el problema DTLZ con preferencia de un decisor para 10 objetivos. 92

Tabla 4.22. Mejores algoritmos por métrica de desempeño para el problema WFG con preferencia de un decisor para 5 objetivos. 94

Tabla 4.23. Mejores algoritmos por métrica de desempeño para el problema WFG con preferencia de un decisor para 10 objetivos. 96

Tabla 4.24. Mejores algoritmos por métrica de desempeño para el problema PSP con preferencia de un decisor para 3 objetivos. 98

Tabla A.1. Condiciones experimentales para la validación de MOSA/D-CGO versus MOSA/D-DE. 119

Tabla A.2. Comparaciones de MOSA/D-CGO y MOSA/D-DE dada la media y desviación estándar (entre paréntesis) del Hipervolumen. 120

Tabla A.3. Comparaciones de MOSA/D-CGO y MOSA/D-DE dada la media y desviación estándar (entre paréntesis) del IGD. 121

Tabla A.3. Condiciones experimentales para la validación de MaOLS-HH-UP 124

Tabla A.4. Comparaciones de MaOLS-HH-UP con sus LLHs (MOSA/D-CGO y MOSA/D-DE) dada la media y desviación estándar (entre paréntesis) del Hipervolumen. 125

Tabla A.5. Comparaciones de MaOLS-HH-UP con sus LLHs (MOSA/D-CGO y MOSA/D-DE) dada la media y desviación estándar (entre paréntesis) de IGD. 126

Tabla A.6. Diseño experimental para MOSA/D-O y MOSA/D-O-II..... 130

Tabla A.7. Resultados de las pruebas con DTLZ and WFG 131

Tabla A.8. Resumen de los resultados de cinco objetivos..... 132

Tabla A.9. Resumen de los resultados de diez objetivos..... 133

Tabla B.1. Puntos de referencia para el cálculo de Hipervolumen..... 134

Tabla B.2. Puntos de referencia para el cálculo de Hipervolumen usados en el trabajo de Falcón-Cardona y Coello [72] 135

Tabla B.3. Se muestran las nueve variantes del hiperheurístico HH_DTLZ. 135

Tabla B.4. Se muestran las nueve variantes del hiperheurístico HH_WFG..... 136

Tabla B.5. Se muestran las diez variantes del hiperheurístico HH_PSP. 136

Índice de Figuras

Figura 2.1. Clasificación de hiperheurísticas según Drake et al. [30]. 17

Figura 2.2. Mapeo de una solución continua a binaria. Fuente [34]. 22

Figura 2.3. Funcionamiento de MOEAD. a) Se generan los sub-problemas λ. b) Se generan soluciones aleatorias. c) Cada sub-problema se optimiza en forma simultánea. 23

Figura 2.4. En coevolución cooperativa dos o más especies interactúan para beneficio mutuo. 24

Figura 2.5. Generación de *S* subgrupos (especies) y su población. Fuente: [41] 25

Figura 3.1. Diagrama a bloques de la metodología de investigación, etapa inicial (verde), etapa intermedia (azul) etapa final (marron). 40

Figura 3.2. Diagrama de clases de la arquitectura general propuesta del HH-Framework .. 45

Figura 3.3. Clase de alto nivel Hyperheuristic 45

Figura 3.4. Clase LLHSelectionOperator 46

Figura 3.5. Clase de alto nivel HHIndicatorSelection 46

Figura 3.6. Clase de bajo nivel LowLevelHeuristic 47

Figura 3.7. La clase LowLevelHeuristic y sus clases derivadas. 47

Figura 3.8. Archivo configuración de la Hiperheurística. 48

Figura 3.9. Parallel coordinate plot de una población de soluciones obtenida mediante MOSA/D-DE para el problema DTLZ4 con 10 objetivos. 48

Figura 3.10. Parallel coordinate plot de dos poblaciones del problema DTLZ4 con 10 objetivos obtenidas por a) MaOLS-HH-UP y b) MOSA/D-DE. 49

Figura 3.11. Ejemplo de colaboración entre tres especies. 57

Figura 3.12. Conversión de $indcand$ a $indbinary$. 58

Figura 3.13. Diagrama de clases de los problemas que maneja el Framework. 60

Figura 3.14. Estructura organizacional de MaOLS-HH-UP 65

Figura 3.15. Diagrama de flujo del marco general de hiperheurísticas. 66

Figura A.1. Gráficas de las poblaciones aproximadas contra el Frente de Pareto de los problemas DTLZ1, DTLZ2, DTLZ3 y DTLZ4 con 3 objetivos. 122

Figura A.2. Gráficas de las poblaciones aproximadas contra el Frente de Pareto de los problemas DTLZ5, DTLZ6 y DTLZ7 con 3 objetivos. 123

Capítulo 1

1 Introducción

La optimización multi-objetivo sigue siendo en la actualidad un tema muy estudiado desde los varios frentes como lo es la optimización evolutiva multi-objetivo (EMO), y la toma de decisiones multi-criterio (MCDM) ambas desde distintos puntos de vista han tratado de resolver este problema. El convencional MCDM refiere a métodos enfocados en programación matemática y, en contraste, los Algoritmos Evolutivos Multi-objetivo (MOEAs) son algoritmos designados a encontrar un conjunto de soluciones [1]. Un concepto clave en la optimización multi-objetivo es la relación de dominancia de Pareto, que define un conjunto de soluciones óptimas donde ninguno de los valores objetivos puede ser mejorado sin empeorar al menos uno de los restantes. A este conjunto de soluciones se le denomina conjunto óptimo de Pareto y a su imagen en el espacio de objetivos frente óptimo

Si bien este tipo de optimización ha tenido muy buen desempeño, muchos de los problemas del mundo real son tan masivos que representan un verdadero reto para los investigadores. Como por ejemplo existen dos conceptos que han sido tratados en forma separada, por un lado, están los problemas de optimización de muchos objetivos (MaOPs) y por el otro los Problemas de Gran Escala (Large-Scale). En ambos casos los investigadores se enfrentan a la maldición de la dimensionalidad. Los problemas más estudiados en la optimización evolutiva han sido los problemas de muchos objetivos, pero a pequeña escala [2], mientras que los problemas de optimización de gran escala lo son, pero con un solo objetivo [3].

Cuando los Problemas de Optimización Multi-objetivo manejan más de tres objetivos se les conoce como Problemas de Optimización de Muchos Objetivos [4]. Una vez que la dimensión del espacio de objetivos se incrementa, la proporción de soluciones no dominadas se incrementa deteriorando la habilidad del proceso de búsqueda para converger [5]. Por lo cual varios autores han creado diversas técnicas para afrontar estos problemas y se pueden clasificar según [6] en: Basados en dominancia relajada, basados en diversidad, basados en agregación, basados en indicador, basados en conjunto referencia, basados en preferencias y basados en reducción de la dimensionalidad.

Recientemente, distintos trabajos han demostrado que integrar preferencias mejora el desempeño de los algoritmos de optimización multi-objetivo debido a que se enfoca en forma más precisa en una región de interés (Region of Interest, ROI) [7] y no en todo el frente de Pareto, existen una gran variedad de métodos para integrar estas preferencias creando a su vez una amplia gama de aproximaciones. Como se puede observar en el trabajo de [8], esta variedad abarca algoritmos genéticos (GA), colonia de hormigas (ACO), optimización mediante nube de partículas (PSO), búsqueda tabú, entre otros. La integración de preferencias en algoritmos es una forma de atacar también algoritmos que manejan muchos objetivos. Como menciona [9], como ejemplo, el Algoritmo GALE basado en preferencias es superior a NSGA-II y SPEA2 en una variedad de problemas.

Por otro lado, los problemas que tienen una gran cantidad de variables de decisión se les conoce como problemas de optimización de gran escala, donde gran escala significa que tiene una dimensión de cientos de variables de decisión [10]. En este tipo de problemas los más estudiados son los problemas de optimización global de gran escala (LSGO). La principal característica de LSGO es que contemplan un único objetivo [11]. Los investigadores han atacado este tipo de problemas mediante el esquema "divide y vencerás"; dividiendo un problema de gran escala en varios problemas de pequeña escala. También existe otro enfoque en el que se agrupan los diversos tipos de variables de decisión mediante una técnica automática como, por ejemplo: agrupamiento aleatorio, agrupamiento dinámico, agrupamiento diferencial, agrupamiento diferencial global, etc. [11].

Uno de los problemas que ha entrado recientemente al campo de la investigación es la combinación de los dos anteriores, es decir, los problemas de optimización de muchos objetivos de gran escala [12]. Hacer la combinación de dos grandes problemas también heredan los mismos retos de sus predecesores, pero también abren nuevos caminos a la investigación.

Dado este panorama, el presente trabajo propone la creación de un framework de optimización basado en hiperheurísticas que permita la integración de preferencias, así como la consideración de imprecisión e incertidumbre en problemas de optimización de muchos objetivos de gran escala. Este framework será configurable y fácil de utilizar para la creación de hiperheurísticas, permitiendo la representación de problemas con variables de decisión tanto continuas como binarias. Su implementación se centrará en la resolución del problema de selección de proyectos (PSP) y en problemas estándar de la literatura, como DTLZ y WFG. Además, el framework será extensible, permitiendo su aplicación en futuros desarrollos de hiperheurísticas.

Esta investigación tiene potenciales aplicaciones futuras en otras áreas de conocimiento donde existan problemas de optimización afines. Un ejemplo es el área de energía, específicamente en la producción de energía eólica, donde se tiene el problema de diseño de parques eólicos (WFLOP). Este tipo de problema multi-objetivo busca en su forma básica maximizar la producción de energía, minimizando el efecto estela que disminuye el rendimiento de los aerogeneradores. Pero puede tener más objetivos como: Minimizar el costo, Minimizar el ruido, Minimizar el área ocupada, entre otros ([13], [14]). Lo que convierte a este problema en un problema de muchos objetivos que puede ser abordado en un futuro usando una solución hiperheurística como las que se abordan en esta investigación.

1.1 Antecedentes

El presente proyecto de investigación se encuentra enmarcado en las actividades del Proyecto de Cátedras CONAHCyT 3058-"Optimización de Problemas Complejos", y las actividades propuestas están orientadas a enriquecer tanto a dicho proyecto como a la investigación realizada dentro del Programa de Doctorado en Ciencias de la Ingeniería del TecNM/ITCM. La propuesta fortalece el framework de Optimización MS-DOSS desarrollado por el grupo al integrar técnicas para el manejo de incertidumbre e imprecisión, la gran escala, y muchos objetivos, en problemas de optimización a través de la implementación e hiperheurísticas. El interés en este proyecto de investigación nace de sus potenciales aplicaciones futuras para la solución de problemas complejos presentes en la generación y manejo de energía eficiente como puede ser la electromovilidad, un área de interés actualmente presente en retos nacionales. El trabajo que se propone busca extender las capacidades de solución del framework MS-DOSS.

1.2 Definición del Problema

Un problema que se presenta en muchos algoritmos es el manejo de la escalabilidad, es decir, cuando el número de objetivos se incrementa (Muchos Objetivos), los métodos de selección basados en la dominancia de Pareto dejan de funcionar debido a la dimensionalidad. Al incrementarse el número de objetivos, el número de soluciones para muestrear el Frente de Pareto (PF) crece exponencialmente [12]. Por lo cual utilizar estrategias de manejo de muchos objetivos resulta conveniente. Un método utilizado para atacar el problema de la escalabilidad en problemas de muchos objetivos es la incorporación de preferencias en el algoritmo de búsqueda, dado que permite aproximarse al PF [9].

Algunos de los modelos de manejo de preferencias más utilizados son: Funciones de utilidad, metas, relaciones de superación, relaciones de preferencia, entre otras ([15]; [16]). Los modelos de manejo de preferencias también poseen sus propios retos, como lo es la representación adecuada de las preferencias de un decisor (DM). Uno de ellos es la imprecisión, dado que muchas veces las preferencias expresadas son imprecisas o difusas, es necesario contemplar el manejo de imprecisión en preferencias. Para el manejo de

imprecisión se han desarrollado diversas técnicas como, por ejemplo: la lógica difusa y el uso de intervalos ([17]; [18]). Al igual que con el número de objetivos, cuando el número de variables de decisión se incrementa (Gran Escala), las metaheurísticas estándar sufren de una degradación en su desempeño por dos razones: La primera es que la superficie del problema se vuelve compleja y la segunda es que el espacio de búsqueda se incrementa exponencialmente [19]. Por lo cual, técnicas de búsqueda local [19] y de descomposición [20] parecen ser lo más adecuado para manejar la escalabilidad en variables de decisión.

Algoritmos como Recocido Simulado y algoritmos basados en poblaciones han reaccionado bien en problemas de optimización global de gran escala (LSGO) [19], por lo que explotar sus cualidades parece conveniente. En este orden de ideas, en la actualidad se está buscando poder aplicar también hiperheurísticas a problemas de gran escala, dado su probado uso en problemas mono-objetivo y más recientemente con el estudio teórico de [21] que muestra su gran potencialidad en problemas multi-objetivo. En el caso de hiperheurísticas para problemas de gran escala, Del Ser et al. [12] señala que aún se debe trabajar para mejorar el desempeño de estas técnicas en dos problemas principales: la codificación de los problemas de estudio y el método de selección del portafolio de metaheurísticas.

Una gran cantidad de los MOEAs desarrollados por los investigadores en la literatura han resuelto los problemas de optimización como los mencionados anteriormente de forma muy particular, los cuales poseen fortalezas y debilidades. Cada heurística y metaheurística tienen ventajas que, si fuera posible explotar, se podrían crear algoritmos más robustos. Esta problemática de poder utilizar las ventajas de cada algoritmo particular puede ser resuelta mediante un framework que los unifique como un portafolio de heurísticas y metaheurísticas y que permita su combinación o hibridación para resolver problemas específicos. Del Ser et al. [12] menciona que una idea interesante es combinar componentes de diferentes técnicas multi-objetivo en un solo Framework que permita aprovechar sus ventajas. Extendiendo lo dicho por Del Ser et al. [12], se puede crear un Framework de hiperheurísticas que permita combinar diversas metaheurísticas y sus componentes para el manejo de Muchos Objetivos y la Gran escala. La combinación estaría representada por una hiperheurística que aprovecharía las ventajas que cada algoritmo y/o técnica puede ofrecer.

Como resumen, se pueden identificar las siguientes problemáticas:

a) El número de soluciones no dominadas tiende a crecer exponencialmente debido al incremento de los objetivos, haciendo difícil obtener una muestra representativa del frente de Pareto.
b) La forma de expresar las preferencias por parte del DM debe ser definida;
c) Las preferencias del DM regularmente no son precisas, por lo cual se debe manejar la imprecisión y la incertidumbre;
d) La superficie de los problemas con muchas variables de decisión se vuelve más compleja.
e) El espacio de búsqueda crece exponencialmente al incrementarse el número de variables de decisión y el número de objetivos.
f) El unificar diferentes estrategias para crear algoritmos robustos (Manejo de Muchos Objetivos, Gran Escala) representa un reto de ingeniería.

Por lo cual, en el presente trabajo contempla atacar las problemáticas anteriores mediante la creación de un Framework de optimización que genere algoritmos robustos para su solución.

1.3 Planteamiento del Problema

Las preguntas de investigación que pretende abordar con la presente propuesta de tesis son:

a) ¿Es posible combinar metaheurísticas en hiperheurísticas dentro de un Framework para mejorar la calidad de las soluciones del problema de Selección de Proyectos?
b) ¿Es posible aproximar la región de interés de un tomador de decisiones a través de una hiperheurística a través de la integración de adecuados modelos para manejo de preferencia, incertidumbre, e imprecisión, y muchos objetivos?
c) ¿Es posible desarrollar modelos para manejo de la gran escala en hiperheurísticas para dar solución a instancias de problemas de optimización específicos que involucren grandes números de variables de decisión?

d) ¿Es posible desarrollar un framework que mejore la versatilidad y dinamismo en las implementaciones de hiperheurísticas para dar solución al problema de selección de proyectos?

1.4 Objetivo General

Desarrollar hiperheurísticas a través de un framework de optimización que integre preferencias en el proceso de solución del problema de selección de proyectos que involucra imprecisión e incertidumbre, gran escala y muchos objetivos.

1.5 Objetivos Específicos

- Desarrollar o utilizar modelos para el manejo de incertidumbre, preferencias, muchos objetivos, y gran escala en el problema de selección de proyectos;
- Desarrollar el framework de optimización que permita implementar hiperheurísticas para la solución del problema de selección de proyectos a partir de la integración de modelos de imprecisión e incertidumbre, preferencias, muchos objetivos, y gran escala;
- Analizar el desempeño de framework de optimización en indicadores que midan la calidad de las soluciones producidas y la facilidad de creación de nuevas hiperheurísticas; y
- Validar el framework de optimización.

1.6 Hipótesis

H_0. La integración conjunta de relaciones de superación, intervalos, y descomposición dentro de un framework de optimización no permite el desarrollo rápido y versátil de hiperheurísticas, con manejo de preferencias en el proceso de búsqueda, que aproximen eficientemente el frente óptimo de Pareto del Problema de Selección de Proyectos y afines en presencia de imprecisión e incertidumbre, gran escala y muchos objetivos.

H_1. La integración conjunta de relaciones de superación, intervalos, y descomposición dentro de un framework de optimización permite el desarrollo rápido y versátil de hiperheurísticas, con manejo de preferencias en el proceso de búsqueda, que aproximen eficientemente el frente óptimo de Pareto del Problema de Selección de Proyectos y afines en presencia de imprecisión e incertidumbre, gran escala y muchos objetivos.

1.7 Justificación

Los problemas del mundo real involucran una serie de retos, como se ha mencionado anteriormente, con problemas de múltiples/muchos objetivos, de muchas variables de decisión, etc. Aunque existen algoritmos clásicos de muy buen funcionamiento, estos suelen tener problemas dado que fueron diseñados para otros propósitos. Pero muchas de sus fortalezas pueden ser aprovechadas. Por lo que los investigadores suelen innovar modificando estos algoritmos y agregándoles nuevos mecanismos que permiten manejar los nuevos desafíos. Del Ser et al. [12] dice que la comunidad también continuará produciendo nuevos sabores de los MOEA más populares en uso actual (es decir, MOEA/D y NSGA-III), pero seguramente hay espacio para mejoras más creativas y desarrollos algorítmicos sin precedentes que pueden allanar vías de investigación inexploradas en este campo [12].

Una forma de allanar estas vías de investigación es la creación de un Framework. Una interesante idea es combinar componentes de diferentes técnicas multi-objetivo en un Framework que permita explotar sus ventajas (Del Ser et al., 2019). Del Ser et al. [12] menciona técnicas multi-objetivo, pero no necesariamente se tendría que limitar a ese tipo de técnicas, puede también combinarse con técnicas y modelos LSGO, muchos objetivos y de gran escala. Este framework tendría las características de ser flexible, extensible y escalable. Una forma de conseguirlo la utilización de buenas prácticas de Programación Orientada a Objetos como lo es S.O.L.I.D. (Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion).

Una vez constituido, el Framework permitiría la creación de nuevos algoritmos mediante la combinación de diversos componentes de otras en heurísticas. A este tipo de estrategias se le llama optimización basada en ensamble. Una combinación de diferentes estrategias, operadores, valores de parámetros y métodos es referida como un ensamble, el cual puede proveer mejores resultados en un conjunto de problemas de optimización que un conjunto simple de estrategias, operadores, valores de parámetros y métodos.

El tener un Framework con diversas heurísticas y sus componentes en forma de piezas que pueden ser ensambladas para crear nuevas heurísticas pueden ser aprovechadas para la creación de Hiperheurísticas. Las hiperheurísticas han sido definidas como métodos de búsqueda o estrategias de aprendizaje que para seleccionar o generar heurísticas para un problema de optimización determinado [22]. De esta manera sería más rápido y eficiente crear hiperheurísticas basados en un framework. Por ejemplo, se podría diseñar una Hiperheurística que cambie de heurística con el tiempo dependiendo de cómo se va comportando históricamente el algoritmo. Por lo mencionado anteriormente, un framework de optimización para la integración de preferencias en hiperheurísticas para la solución de problemas de muchos objetivos de gran escala tiene las siguientes ventajas generales:

Evitar tareas repetitivas. Los algoritmos, en especial los evolutivos, tienen componentes comunes, por lo que se evitaría hacer la misma tarea para diferentes algoritmos o, por lo menos, reducir el trabajo al crear componentes basados en ya existentes;

Aumentar la productividad. Al tener componentes ya preexistentes, ensamblar un nuevo algoritmo sería más rápido y eficiente;

Favorecer el trabajo en equipo. Partiendo de un mismo marco de trabajo el trabajo en equipo puede mejorar el desempeño, al manejar un mismo estándar el equipo puede integrar más fácilmente las distintas partes de una misma solución;

Asegurar que el proyecto se realiza de la mejor forma conocida. Al utilizar una metodología para integrar preferencias, manejar muchos objetivos y gran escala, se asegura que es algo que ya se ha utilizado y aplicado antes con buenos resultados;

Una curva de aprendizaje más corta. Se acorta la curva de aprendizaje si ya se utiliza una metodología y se puede invertir tiempo en problemas más puntuales;

Hace más fácil responder al cambio. Si de pronto se tiene que hacer un cambio, se sabe puntualmente donde puede ser realizado o también qué nueva estrategia aplicar;

Y las siguientes ventajas particulares:

Manejo de incertidumbre e imprecisión: El framework tendría la fortaleza de poder modelar preferencias de una forma más robusta, pues si el decisor tiene imprecisión es lo que prefiere, esto puede ser modelado;

Integración de preferencias en el proceso de búsqueda: Dado la alta escalabilidad de los problemas que se resolverán, el manejo de preferencias sería una ventaja del Framework, pues esto acortaría el proceso de búsqueda dada su capacidad de aproximarse al PF;

Manejo de muchos objetivos. Integraría componentes de algoritmos de manejo de muchos objetivos que servirían para crear nuevos algoritmos o innovar los existentes. En especial atención, una estrategia para atacar muchos objetivos es la integración de preferencias que el presente trabajo contempla;

Manejo de gran escala. Sería posible atacar problemas de gran escala debido a que el Framework contempla el manejo de componentes de que se han usado para este tipo de problemas;

Implementación fácil de hiperheurísticas. Es posible, al disponer de un Framework crear algoritmos de tipo hiperheurística que puedan combinar diversas estrategias de forma aplicada para resolver un determinado tipo de problema;

Aplicación a otros problemas de optimización. Es posible reutilizar las hiperheurísticas desarrolladas para aplicarse a otros problemas de optimización.

Capítulo 2

2 Marco Teórico

En este capítulo se describen los conceptos básicos y el fundamento teórico y matemático necesario para realizar este trabajo. Se comenzará hablando de los tipos de problemas existentes, los tipos de algoritmos que se utilizarán, las de decisiones, preferencias e intervalos, temas que son necesarios para este trabajo.

2.1 Problemas de Optimización

2.1.1 Problemas de Optimización Multi-objetivo (MOPs)

Un problema de optimización multi-objetivo (MOP) es un problema con más de un objetivo en conflicto que debe optimizarse simultáneamente. Formalmente, la ecuación 2.1 puede definir formalmente un MOP con *m* objetivos.

$$\max F(x) = (f_1(x), f_2(x), \ldots, f_m(x)) \quad (2.1)$$
$$\text{subject to: } x \in \Omega$$

Según Fleming et al. [23] los problemas de optimización multi-objetivo (MOPs) son aquellos con dos o tres objetivos.

2.1.2 Problemas de Optimización de Muchos Objetivos (MaOPs)

Aquellos problemas con más de tres objetivos son conocidos como problemas de optimización de muchos objetivos (MaOPs) ([24]; [25]). Según Ishibuchi et al. [24], cuando se utiliza algún algoritmo conocido basado en dominancia de Pareto para resolver problemas de muchos objetivos se presentan los siguientes problemas:

- Deterioro de la capacidad de búsqueda de algoritmos basados en dominancia de Pareto (Ejemplo: SPEA y NSGA-II).
- Aumento exponencial del número de soluciones necesarias para aproximar todo el frente de Pareto.
- Dificultad para visualizar las soluciones.

2.1.3 Problemas de Optimización Global de Gran Escala (LSGO)

En este tipo de problemas tiene un solo objetivo y muchas variables de decisión o gran escala. Usualmente, MOPs con cientos de variables de decisión son considerados de gran escala [10]. Como menciona Zhang et al., [26] muchos problemas del mundo real involucran muchos objetivos y variables de decisión a gran escala. Esta combinación de muchos objetivos y gran escala da origen a los problemas de optimización de muchos de objetivos de gran escala (LSMaOPs). Los cuales heredan las características de los problemas de muchos objetivos y los problemas de optimización global de gran escala mencionados anteriormente.

2.2 Problema de Selección de Proyectos (PSP)

Es un proceso que es de vital importancia para en el sector público y privado. Cuando existen factores de riesgo, el decisor (DM) tiene la responsabilidad de seleccionar la mejor cartera. A la solución de este problema se le define como cartera. Una cartera se encuentra definida por un conjunto de proyectos, los cuales se encuentran limitados a un presupuesto, donde cada proyecto impacta en más de un objetivo en diferente proporción. De acuerdo con (Carazo et al., 2008), un proyecto es un proceso temporal, único e irrepetible que persigue un conjunto específico de objetivos. Por otra parte, una cartera es un conjunto de proyectos que pueden ser realizados en el mismo periodo de tiempo [27]. Por esta razón, los proyectos que pertenecen a una misma cartera comparten los recursos disponibles de la organización.

Consideremos una situación de toma de decisión en la cual el DM es el encargado de seleccionar un grupo de proyectos (cartera) que una institución llevará a cabo. El objetivo de este problema de decisión es elegir la "mejor" cartera satisfaciendo algunas restricciones presupuestales.

Formalizando estos conceptos, consideremos un conjunto de N proyectos, donde el i-ésimo proyecto es representado por un vector p-dimensional $F(x_i) = (f_1(x_i), f_2(x_i), f_3(x_i), \dots, f_p(x_i))$, donde cada $f_j(x_i)$, indica la contribución del proyecto i al j-ésimo objetivo. Cada objetivo indica el beneficio, el cual es, el número de personas que pertenecen a una categoría social (pobreza extrema, pobreza, clase media) y quienes reciben un nivel de beneficio (alto impacto, impacto medio, impacto bajo) del i-ésimo proyecto. El i-ésimo proyecto corresponde a un área (salud, educación, etc.) señalada por a_i. Cada área tiene un límite presupuestal definido por el DM o por cualquier otra autoridad competente. Consideremos para cada área k, un límite inferior y superior, L_k y U_k respectivamente. Basado en esto, la restricción de cada área k de acuerdo a la Ecuación 1 es:

$$L_k \leq \sum_{i=1}^{N} x_i g_i(k) c_i \leq U_k \qquad (2.2)$$

Donde $g(k)$ se define en la Ecuación 2 como:

$$g_i(k) = \begin{cases} 1 & si\ a_i = k, \\ 0 & en\ otro\ caso \end{cases} \qquad (2.3)$$

Además, cada proyecto corresponde a una región geográfica a la cual beneficiará. De la misma forma que las áreas, cada región tiene un límite inferior y superior como otra restricción que debe ser cumplida por una cartera factible. La calidad de la cartera *x* es determinada por la unión de los beneficios de cada uno de los proyectos que la componen. Esto puede ser expresado en la Ecuación 2.4 como:

$$z(x) = (z_1(x), z_2(x), z_3(x), \dots, z_p(x)) \qquad (2.4)$$

Donde $z_j(x)$ de la ecuación 2.4 en su forma más simple se define en la Ecuación 2.5 como:

$$z_j(x) = \sum_{i=1}^{N} x_i f_i\ (i) \qquad (2.5)$$

Si expresamos por F_R la región de carteras factibles, el problema de cartera de proyectos es identificar una o más carteras que resuelvan:

$$max_{x \in R_F}\{z(x)\} \qquad (2.6)$$

Por otro lado, una cartera denominada x es un subconjunto de esos proyectos el cual es modelado generalmente como un vector binario $x = (x_1, x_2, \dots, x_n)$. En este vector, x_i es una variable binaria donde $x_i = 1$ si el i-ésimo proyecto es apoyado y $x_i = 0$ en otro caso. Existe un presupuesto total, especificado en la ecuación 2.7, que la organización está dispuesta a invertir, el cual es denotado como *B*, y cada proyecto x_i tiene un costo asociado C_i, por lo que la sumatoria de los costos asociados a cada proyecto debe ser menor o igual al presupuesto total. Por lo tanto, las carteras están sujetas a la restricción presupuestal:

$$\left(\sum_{i=1}^{N} x_i * c_i\right) \leq B \qquad (2.7)$$

2.3 Estrategias de Búsqueda

2.3.1 Heurísticas

El término "heurística" proviene de una palabra griega con un significado relacionado con el concepto de encontrar y se vincula a la supuesta exclamación eureka de Arquímedes al descubrir su famoso principio. Se califica de heurístico a un procedimiento para el que se tiene un alto grado de confianza en que encuentra soluciones de alta calidad con un coste computacional razonable, aunque no se garantice su estatus de óptimo o su factibilidad, e incluso, en algunos casos, no se llegue a establecer lo cerca que se está de dicha situación.

Se usa el calificativo heurístico en contraposición a exacto, que se aplica a los procedimientos a los que se les exige que la solución aportada sea óptima o factible. Unas heurísticas para resolver un problema de optimización pueden ser más generales o específicas que otras. Los métodos heurísticos específicos deben ser diseñados a propósito para cada problema, utilizando toda la información disponible y el análisis teórico del modelo. Los procedimientos específicos bien diseñados suelen tener un rendimiento significativamente más alto que las heurísticas generales [28].

2.3.2 Metaheurísticas

El término metaheurísticas se obtiene de anteponer a heurística el sufijo meta que significa "más allá" o "a un nivel superior". Las metaheurísticas son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos muy generales con un alto rendimiento. El término metaheurística apareció por primera vez en el artículo seminal sobre búsqueda tabú de Glover [29]. A partir de entonces han surgido multitud de propuestas de pautas para diseñar buenos procedimientos para resolver ciertos problemas que, al ampliar su campo de aplicación, han adoptado la denominación de metaheurísticas. Las metaheurísticas son estrategias para diseñar procedimientos heurísticos. Por tanto, los tipos de metaheurísticas se establecen, en primer lugar, en función del tipo de procedimientos a los que se refiere. Algunos de los tipos fundamentales son las metaheurísticas para los métodos de relajación, las metaheurísticas para los procesos constructivos, las metaheurísticas para las búsquedas por entornos y las metaheurísticas para los procedimientos evolutivos [28].

- Las metaheurísticas de relajación se refieren a procedimientos de resolución de problemas que utilizan relajaciones del modelo original (es decir, modificaciones del modelo que hacen al problema más fácil de resolver), cuya solución facilita la solución del problema original.

- Las metaheurísticas constructivas se orientan a los procedimientos que tratan de la obtención de una solución a partir del análisis y selección paulatina de las componentes que la forman.
- Las metaheurísticas de búsqueda guían los procedimientos que usan transformaciones o movimientos para recorrer el espacio de soluciones alternativas y explotar las estructuras de entornos asociadas.
- Las metaheurísticas evolutivas están enfocadas a los procedimientos basados en conjuntos de soluciones que evolucionan sobre el espacio de soluciones.

2.3.3 Hiperheurísticas

La característica principal de las hiperheurísticas es que buscan un espacio de heurísticas en lugar de un espacio de soluciones directamente. En este sentido, difieren de la mayoría de las aplicaciones de las metaheurísticas, aunque, por supuesto, las metaheurísticas pueden (y han sido) ser utilizadas como hiperheurísticas. La motivación detrás de las hiperheurísticas es elevar el nivel de generalidad en el que operan las metodologías de búsqueda [22].

Una hiperheurística es un método de búsqueda o mecanismo de aprendizaje para seleccionar o generar heurísticas para resolver problemas de búsqueda computacional. Burke et al. [22] establecen dos categorías principales para clasificar hiperheurísticas:

La clasificación propuesta de enfoques hiperheurísticos se puede resumir de la siguiente manera (Figura 2.1):

- Con respecto a la naturaleza del espacio de búsqueda heurística
    - Metodologías de selección heurística: Producen combinaciones de preexistentes:
        - Heurística de construcción
        - Heurísticas de perturbación
    - Metodologías de generación heurística: Genere nuevos métodos heurísticos utilizando componentes básicos (bloques de construcción) de:
        - Heurística de construcción
        - Heurísticas de perturbación

- Con respecto a la fuente de retroalimentación durante el aprendizaje
    - Hiperheurística de aprendizaje en línea: aprenda mientras resuelve una instancia determinada de un problema.
    - Hiperheurística de aprendizaje fuera de línea: aprende, a partir de un conjunto de instancias de capacitación, un método que se generalizaría a instancias desconocidas.
    - Hiperheurística sin aprendizaje: no utiliza la retroalimentación del proceso de búsqueda.

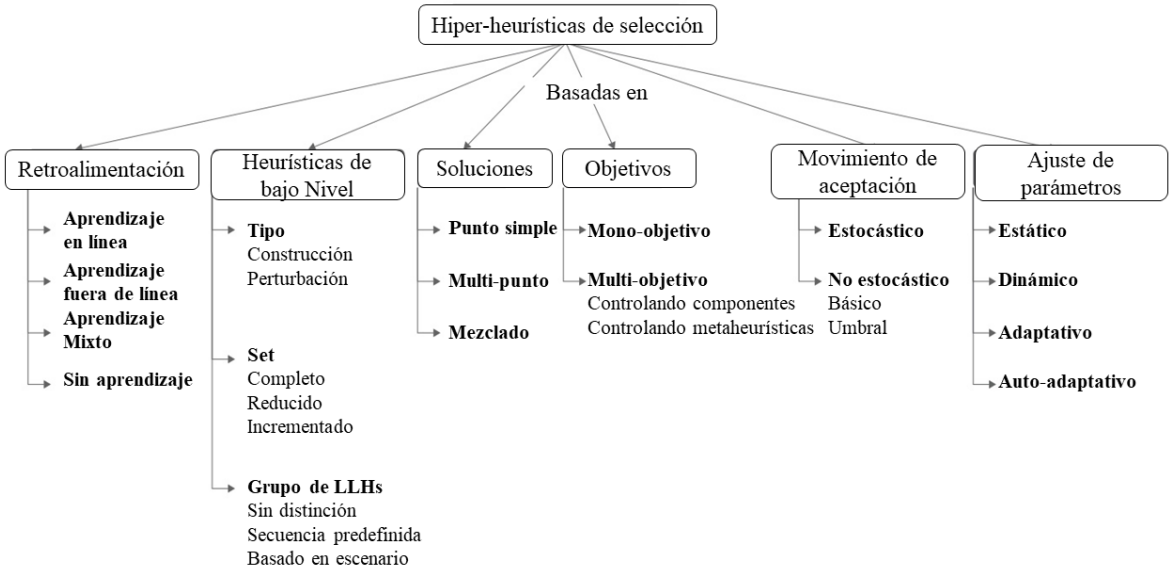


Figura 2.1. Clasificación de hiperheurísticas según Drake et al. [30].

2.4 Algoritmos Metaheurísticos

2.4.1 Recocido Simulado (Simulated Annealing)

En 1953, Metropolis et al. propuso un algoritmo para la simulación eficiente de la evolución de un sólido hacia el equilibrio térmico. Treinta años después, Kirkpatrick et al. [31] y Cerny [32] establecieron una analogía entre minimizar la función de costo de un problema de optimización combinatoria y enfriar lentamente un sólido, utilizando el proceso de optimización presentado por Metropolis. Al ejecutar el algoritmo de Metropolis en una secuencia de valores de temperatura que disminuyen lentamente, Kirkpatrick y sus colaboradores obtuvieron un algoritmo de optimización combinatoria, al que llamaron "Recocido Simulado" (SA). El algoritmo de recocido simulado imita la evolución de un sistema físico inestable hacia un equilibrio termodinámico a una temperatura fija.

En cada iteración de dicho procedimiento, una nueva solución (x') es seleccionada de forma aleatoria de las vecindades de la solución actual (x). El algoritmo de recocido simulado acepta nuevas soluciones de acuerdo con dos criterios: i) si el valor de la función objetivo de la nueva solución es mejor, ii) si el valor de la función objetivo de la nueva solución es peor, y un valor aleatorio generado entre cero y uno es menor que la diferencia entre la solución actual y la nueva solución divido por la temperatura del sistema *T*. Al inicio del algoritmo, la temperatura del sistema es ajustada en un valor *T0*. Este valor disminuye cada N_{cool} iteraciones de manera proporcional a un factor de enfriamiento $\propto$, donde $\propto < 1$. El Algoritmo a continuación resume el procedimiento de recocido simulado [33].

Algorithm 2.1: Recocido Simulado

Input: Temperatura inicial Ti, Temperatura final Tf, factor de temperatura α, ciclos L
Output: Solucion S_{act}

```
1  S_act = initializeSolution()
2  while Ti ≥ Tf do
3      for i ← 1 to L do
4          S_cand = perturbation(S_act)
5          Δ = f(S_cand) − f(S_act)
6          p = e^(−Δ/T)
7          if Δ < 0 or U(0,1) < p then
8              S_act = S_cand
9          end if
10     end for
11     Ti = Ti * α
12 end while
```

2.4.2 Algoritmo de Evolución Diferencial

Un algoritmo de evolución diferencial (DE) es una técnica de optimización global que se utiliza para encontrar soluciones aproximadas a problemas difíciles de optimización de tipo continuo. Fue propuesto por Storn y Price en 1997 [34] y se ha convertido en una herramienta ampliamente utilizada para abordar problemas de optimización en diversas disciplinas. Su funcionamiento se describe en el algoritmo 2.2: Comienza con la generación de la población inicial de tamaño NP en la línea 2. Después comienza el ciclo principal por un número máximo de G_{max} generaciones. Dentro del ciclo principal realiza las siguientes operaciones:

- Por cada uno de los individuos objetivo de la población $P_G = \{X_{1,G}, X_{2,G}, \dots, X_{NP,G}\}$ hará las siguientes operaciones:
    - Mutación. Usando al menos tres individuos aleatorios ($X_{r1,G}$, $X_{r2,G}$, $X_{r2,G}$) se genera un individuo mutante denotado por $V_{i,G}$ (línea 6).
    - Cruza. La cruza se realiza entre el individuo objetivo $X_{i,G}$ y $V_{i,G}$ para generar el individuo de prueba $U_{i,G}$ (línea 8 y 9).

- Selección. Se compara la función de aptitud para los individuos objetivo $X_{i,G}$ y de prueba $U_{i,G}$. Si $f(U_{i,G}) \leq f(X_{i,G})$, entonces la solución en el i-ésimo individuo objetivo de la siguiente generación es $X_{i,G+1} = U_{i,G}$ ó en caso contrario $X_{i,G+1} = X_{i,G}$.

Algorithm 2.2: Differential Evolution Algorithm

Input: Scale Factor F, Crossover rate Cr, population size NP, generations G_{MAX}

Output: Population $P_{G_{MAX}}$

```
1  G = 0
2  P_G = initializePopulation()
3  while G ≤ G_MAX do
4      for i ← 1 to NP do
5          //mutation
6          V_{i,G} = X_{r1,G} + F * (X_{r2,G} − X_{r3,G})
7          //Crossover
8          U_{i,G} = {u^1_{i,G}, u^2_{i,G}, ..., u^D_{i,G}}
9          u^j_{i,G} = { v^j_{i,G} if rand_j(0,1) ≤ Cr or j = j_rand
                       { x^j_{i,G} otherwise
           , for j = 1, 2, ..., D
10         //Selection
11         if f(U_{i,G}) ≤ f(X_{i,G}) then
12             X_{i,G+1} = U_{i,G}
13         else
14             X_{i,G+1} = X_{i,G}
15         end if
16     end for
17     G = G + 1
18 end while
19 return P_{G_MAX}
```

Al alcanzar el número máximo de generaciones G_{max} se retorna la población de la última generación $P_{G_{max}}$ (línea 19).

2.4.2.1 Operadores de Evolución Diferencial

Los operadores se utilizan en un orden diferente al de los operadores tradicionales de algoritmos genéticos. Primero, se debe calcular la mutación; entonces, se debe ejecutar la cruza.

Mutación Rand/1. Rand/1 es una mutación básica de las estrategias de mutación más populares en Evolución Diferencial (DE) [35]. Funciona de la siguiente manera:

$$V_{i,G} = X_{r1,G} + F * (X_{r2,G} - X_{r1,G}) \tag{2.8}$$

En la ecuación (2.8), $X_{r1,G}$, $X_{r2,G}$, $X_{r2,G}$ representan tres soluciones aleatorias de la población en la generación G, y $V_{i,G}$ representa una nueva solución mutante. *F* es un parámetro llamado factor de escala.

Cruce binomial. Después de la mutación, se realiza una operación de cruce entre el vector objetivo $X_{i,G}$ y el vector mutante $V_{i,G}$ para producir un vector de prueba $U_{i,G} = \{u_{i,G}^1, u_{i,G}^2, \dots, u_{i,G}^D\}$. Se utiliza principalmente el operador de cruce binomial, que se define como:

$$u_{i,G}^j = \begin{cases} v_{i,G}^j & si\ rand_j(0,1) \le Cr\ or\ j = j_{rand} \\ x_{i,G}^j & de\ otro\ modo \end{cases} \tag{2.9}$$

En la ecuación (2.9), $Cr \in [0,1]$ representa la tasa de cruce y j_{rand} representa la diferencia mínima entre el vector objetivo $X_{i,G}$ y el vector de prueba $U_{i,G}$.

2.4.2.2 Conversión de Variables Continuas a Binarias

Conversión de soluciones continuas a binarias. En un algoritmo DE tradicional, se genera aleatoriamente un conjunto inicial de soluciones candidatas, cada una representada por un vector continuo de una longitud igual al número total de elementos (*N*). En el trabajo de Ali et al. [36] para calcular los valores de aptitud de estos individuos, sus valores continuos (con_pop) se asignan a valores binarios (bin_pop) utilizando el valor continuo promedio de cada solución (Figura 2.2), como denota la ecuación 2.11.

$$bin\_pop_j^i = \begin{cases} 0 & con\_pop_j^i < Avg_i \\ 1 & con\_pop_j^i \geq Avg_i \end{cases} \tag{2.11}$$

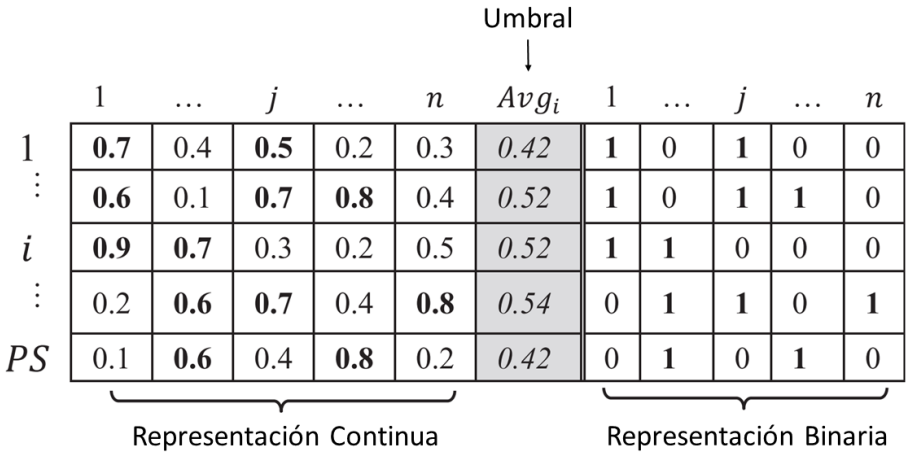


Figura 2.2. Mapeo de una solución continua a binaria. Fuente [34].

2.4.3 El Algoritmo Evolutivo Multi-objetivo Basado en Descomposición (MOEA/D).

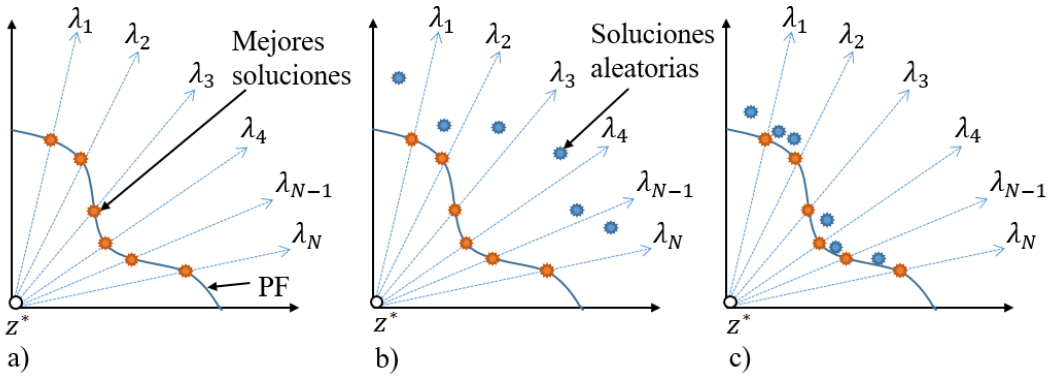
El MOEA/D fue propuesto por Zhang y Li [37] este se basa en descomponer un problema Multi-objetivo en un conjunto de sub-problemas escalares que MOEA/D optimiza en forma simultánea. Sean $\lambda_1, \dots, \lambda_N$ un conjunto de vectores de pesos y z^* un punto de referencia. El problema de optimización multi-objetivo se puede descomponer en N subproblemas de optimizacion escalar, utilizando el enfoque de Tchebycheff y la función objetivo del j-ésimo subproblema, de la siguiente forma:

$$g^{te}\left(x\middle|\lambda^j, z^*\right) = \max_{1<i<m} \{\lambda_i^j | f_i(x) - z_i^*\} \tag{2.12}$$

donde $\lambda^j = (\lambda_1^j, \dots, \lambda_m^j)$. MOEA/D minimiza estas N funciones objetivo en una sola ejecución. Un vecindario de vectores de peso λ_i está definido como un conjunto de sus vectores de pesos más cercanos en $(\lambda_1, \dots, \lambda_N)$. Así el vecindario del i-ésimo sub-problema consiste en todos los sub-problemas con los vectores de pesos del vecindario de λ_i. La población está compuesta de las mejores soluciones encontradas hasta el momento para cada problema. z^* representa el punto ideal, pero en el proceso de MOEA/D está conformado por el mejor valor conocido para cada objetivo [38].

MOEA/D funciona (Figura 2.3) de la siguiente manera:

1. Se generan N subproblemas λ basados en C1,
2. Se genera una población inicial con Nindividuos, un individuo por cada subproblema,
3. Se determina la dominancia de cada individuo mediante un ordenamiento no dominado,
4. Se genera una nueva población mediante operadores genéticos,
5. Se elige una nueva población actual seleccionado individuos de las dos poblaciones,
6. Se elimina a los individuos dominados que exceden la población de tamaño N ,
7. Si se cumple condición de paro se termina el algoritmo, si no se pasa al paso 4,
8. Las mejores soluciones están representadas por la población actual.



9.

Figura 2.3. Funcionamiento de MOEAD. a) Se generan los sub-problemas λ. b) Se generan soluciones aleatorias. c) Cada sub-problema se optimiza en forma simultánea.

2.4.4 Coevolución Cooperativa (CC)

La coevolución cooperativa es un proceso en el cual dos especies interactúan entre sí de tal forma que ambas resultan beneficiadas en la interacción. Por ejemplo, regularmente las vacas tienen ciertos problemas con parásitos como las garrapatas, pulgas y piojos, y esto representa un problema. Pero entonces entran en acción las garzas que consumen estos parásitos, dándose una interacción llamada mutualismo donde ambos se benefician (Figura 2.4).

La idea principal de esta técnica es descomponer un problema de alta dimensión en subcomponentes de baja dimensión y evolucionar estos cooperativamente por un número predefinido de ciclos. Un ciclo consiste en una evolución completa de cada una de las subpoblaciones por un número de generaciones establecido. La cooperación ocurre cuando se evalúan a los individuos, ya que es aquí en donde se lleva a cabo la interacción entre los

individuos de cada una de las especies. Al evaluar cada individuo o solución, esta estará formada por las distintas especies que coevolucionan.

Figura 2.4. En coevolución cooperativa dos o más especies interactúan para beneficio mutuo.

La idea principal de un algoritmo con coevolución cooperativa es la siguiente:

- Una especie representa un subcomponente de una solución potencial.
- Las soluciones completas se obtienen de la unión de miembros representativos de cada especie.
- La asignación de crédito a nivel especie se define en términos del resultado de la colaboración entre especies.
- Cuando se requiere, el número de especies evoluciona por sí misma.
- La evolución de cada especie es manejada por un algoritmo evolutivo.

El primer marco de trabajo de coevolución cooperativa (CC) fue propuesto por Potter y De Jong [39], y en el trabajo de Yang et al. [40] se propone un resumen del marco de trabajo de CC:

1. Descomponer el vector objetivo en m subcomponentes de menor dimensión.
2. Iniciar un índice i=1 para comenzar un nuevo ciclo.
3. Optimizar el i-ésimo subcomponente con un cierto algoritmo evolutivo por un número predefinido de evaluaciones de fitness (FE).
4. Si i < m entonces volver al paso 3.
5. Parar si el criterio de parada es satisfecho, en caso contrario volver al paso 2 para comenzar otro ciclo.

En el trabajo de Antonio y Coello [41] trasladan lo utilizado por Potter y De Jong [39] en algoritmos LSGO al área de los algoritmos de optimización multi-objetivo de gran escala (LSMOPs). Recordando que los problemas que presentan cientos de variables de decisión se les considera de gran escala, los algoritmos basados en CC tienen buenos resultados, pero fallan cuando el conjunto de variables es no separable. Por tal motivo, Antonio y Coello [41]

utilizan una estrategia de crear subgrupos llamados especies, cada subgrupo estará asociado a un número *m* de variables de decisión seleccionadas de manera aleatoria. De forma que se tienen *D* variables de decisión, dividido en S subgrupos de tamaño m por lo cual podemos decir que $D=m\times S$. Cada subgrupo *S* tiene un número NP de individuos como se puede observar en la figura 2.5.

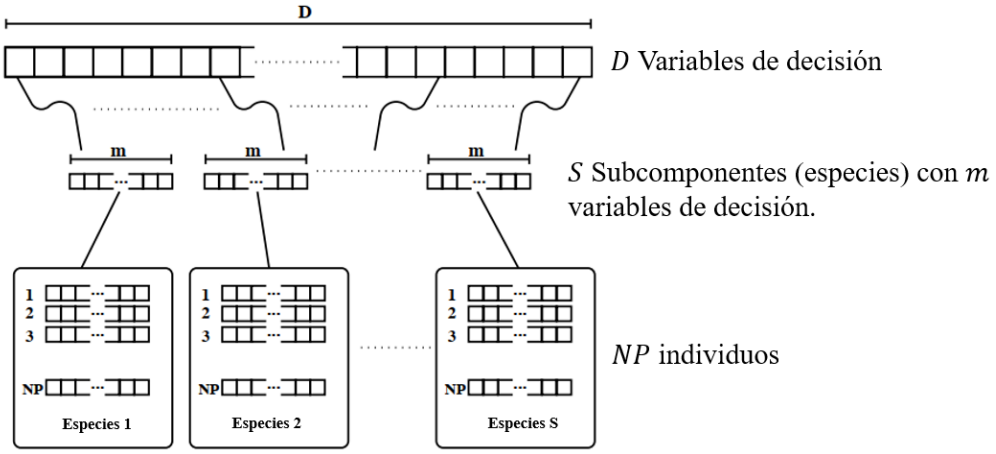


Figura 2.5. Generación de *S* subgrupos (especies) y su población. Fuente: [41]

Una vez que las subpoblaciones han sido creadas para cada especie, se inicia un número de ciclos donde cada especie evoluciona de forma independiente. La colaboración se dará en cada generación de la siguiente forma: En la primera generación se evaluarán las soluciones que estarán compuestas por un representante de cada especie elegido de forma aleatoria. Esta evaluación se hará usando las funciones objetivo y dicho resultado será devuelto a cada representante, para así obtener su aptitud y saber qué tan buena fue su colaboración.

A partir de la segunda generación se crearán soluciones mediante la combinación de un representante de cada especie. Este será seleccionado de forma aleatoria del mejor nivel de individuos no dominados de cada especie. Al evaluar cada solución, el resultado es devuelto a cada individuo que lo compone para saber su aptitud respecto a su colaboración. Esto seguirá por varias generaciones hasta que se cumpla una condición de paro, que suele ser un número determinado de ciclos. Al final, se aplica un ordenamiento basado en dominancia a los mejores niveles de cada subpoblación para obtener el conjunto final de soluciones.

2.5 Toma de Decisiones

Es un proceso de aprendizaje natural o estructurado mediante el cual se tiene que elegir entre dos o más alternativas de solución que ayuden a resolver una problemática específica que se le presenta a una o varias personas, que es o son conocidas comúnmente como tomadores de decisiones. Tenemos que para poder tomar una buena decisión es necesario tener la información necesaria con la finalidad de poder visualizar los posibles escenarios y contemplar así el impacto positivo, el riesgo y las consecuencias que trae consigo esta decisión [42]. Para tomar una decisión, no importa su naturaleza, es necesario conocer, comprender, analizar un problema, para así poder darle solución; en algunos casos por ser tan simples y cotidianos, este proceso se realiza de forma implícita y se soluciona muy rápidamente, pero existen otros casos en los cuales es necesario realizar un proceso más estructurado que puede dar más seguridad e información para resolver el problema.

Para hacer el análisis y la evaluación de las carteras generadas se realiza de manera subjetiva, es decir, que se pueden emitir opiniones, juicios, prioridades, etc. del DM, los cuales representan las necesidades específicas de una organización, lo cual se conoce como preferencias. Por lo que podemos definir como proceso de toma de decisiones a la situación en la cual el tomador de decisiones tiene que decidir cuál o cuáles son las carteras que se apegan fielmente a sus preferencias. Lo que implica que para tomar una decisión se necesita tener al menos dos soluciones para poder discernir cuál es la mejor alternativa. [42].

2.5.1 Modelos de Preferencia

Son modelos que ayudan a determinar cuál es la alternativa que se adapte a las, metas, objetivos, deseos del DM tanto como sea posible [42]. Por esta razón, el DM es el actor central que existe en todo problema de decisión, ya que el modelo preferencial expresa las necesidades. El DM cuenta con un sistema de preferencias, el cual representa con valores las prioridades, metas, objetivos, deseos, etc. Las cuales impactan de manera directa en la selección de atributos, ponderándolos de manera subjetiva, esto tiene impacto en el proceso de solución. Según Coello [43] y Wang et al. [44] algunos de los modelos de preferencias utilizados en el campo de la optimización son:

- Metas.
- Pesos.
- Funciones de utilidad.
- Relaciones de preferencia.
- Relaciones de superación (outranking).
- Vectores de referencia.
- Lógica difusa

2.5.2 Conceptos Básicos de Outranking

El concepto de outranking está ligado al campo de la Toma de Decisiones Multicriterio (MCDA). Esta es alternativa a opciones como las Funciones de Utilidad de uso frecuente en MCDA. El outranking representa un modelo de preferencias integral basado en relaciones de outranking. Donde se busca comparar dos opciones a' y a basado claramente en las preferencias de un decisor (DM) con la siguiente afirmación "a' es al menos tan buena como a" ($a'Sa$). La confirmación de la afirmación $a'Sa$ puede estar respaldada por $a'\Delta_F a$ (a'domina a a) pero otros argumentos válidos pueden usarse. La construcción de una relación de outranking significa enriquecer la dominancia de manera que se facilite la solución de un problema de decisión [45].

2.5.2.1 El Concepto de Outranking

Outranking es un modelo de preferencias integrales con las siguientes consideraciones:

i. Un conjunto A de potenciales acciones,
ii. Una familia F de n criterios gj,
iii. La imprecisión, incertidumbre o inexacta determinación de las funciones puede llevar a juzgar prematuramente: indiferencia o preferencia estricta de a' sobre a y
iv. Se debe considerar un nivel integral en la comparación de a' y a basado en $gj(a')$ y $gj(a)$. El modelo integral no pretende ser un modelo detallado de preferencias, pero si uno que refleje las preferencias de un decisor [46].

2.5.2.1.1 El Concepto de Outranking: Restringido al j-ésimo Criterio

Cada criterio $g_j \in F$ es posible asociarlo a una relación outranking restringida S_j. Por definición S_j es una relación binaria: $a' S_j a$ es valido si los valores de las funciones $g_j(a')$ y $g_j(a)$ dan un argumento lo suficientemente sólido para considerar verdadera la siguiente declaración: a' con respecto al j-ésimo criterio solamente, es al menos tan buena como a.

2.5.2.1.2 El concepto de Outranking: Nivel de Preferencias Integrales

Tomando en cuenta en cuenta toda la familia de criterios, es posible definir una relación de outranking integral S. Por definición S es una relación binaria: $a' S a$ es válida si los valores de las funciones $g(a')$ y $g(a)$ dan un argumento suficientemente solido para considerar verdadera la siguiente declaración: a' con respecto a los n criterios, es al menos tan buena como a.

2.5.2.2 Conceptos para Construir Relaciones de Outranking

La expresión formal y aun la naturaleza de las condiciones que deben ser satisfechas para validar la afirmación $a' S a$ puede estar influenciada por muchos factores. Los más importantes son:

- El grado de significancia de los criterios tomados en cuenta en F,
- La naturaleza de los conceptos básicos usados: concordancia, discordancia, tasa de sustitución, intensidad de preferencia, etc.,
- La información entre criterios requerida,
- y (iv) la solidez de los argumentos requeridos: El argumento más sólido que podemos imaginar para validar $a' S a$ es $a' \Delta_F a$ (a'domina a a), pero argumentos débiles pueden ser suficientes. Por lo cual S es mas versátil que Δ_F.

La variedad de opciones que pueden existir para cada uno de los criterios expuestos indica que no existe una forma mejor y única de formular condiciones para validar la afirmación $a' S a$. Por lo cual los métodos ELECTRE son una forma particular de formular condiciones para validar la afirmación $a' S a$, con el fin de construir relaciones de outranking en las que estos están basados.

Por lo que los métodos ELECTRE se basan primordialmente en las siguientes condiciones para los factores antes mencionados:

- El grado de significancia de cada criterio g_j se refleja mediante dos umbrales indiferencia $\boldsymbol{q_j}$ y preferencia $\boldsymbol{p_j}$,
- Los conceptos básicos utilizados son los de **concordancia** y **discordancia**,
- La información entre criterios es sintetizada por dos tipos de datos: para cada criterio g_j, su **coeficiente de importancia** $\boldsymbol{k_j}$ y su **umbral de veto** $\boldsymbol{v_j}$,
- Con respecto a la solidez de los argumentos, se han elegido dos modelos para expresar la solidez de las relaciones de outranking: El primero de forma nítida por medio del conjunto $\boldsymbol{r}$ $(1,\ 2, \ldots,\ r)$. El segundo de forma difusa denotado por el par ordenado $\boldsymbol{\sigma(a', a)}$ $(0 \leq \sigma(a', a) \leq 1)$ llamado ***índice de credibilidad***.

2.5.2.3 Criterio de Concordancia.

Por definición el j-esimo criterio esta en concordancia con la afirmación $a'Sa$ si y solo si $a'S_j a$.

$$a'S_j a \text{ si y solo si } \ g_j(\text{a}') \geq g_j(a) - \ q_j \qquad (2.16)$$

Al subconjunto de todos los criterios de F que están en concordancia con la afirmación $a'Sa$ se le llama ***coalición de concordancia*** y es denotada por $C(a'Sa)$

2.5.2.4 Criterio de Discordancia.

Por definición el j-esimo criterio esta en discordancia con la afirmación $a'Sa$ si y solo si a$P_j a'$.

$$aP_j a' \text{ si y solo si } \ g_j(a) \geq g_j(a') + p_j \qquad (2.17)$$

Al subconjunto de todos los criterios de F que están en discordancia con la afirmación $a'Sa$ se le llama ***coalición de discordancia*** y es denotada por $C(aPa')$

Derivado de las definiciones anteriores tenemos que:

$$C(a'Sa) \cap C(aPa') = \emptyset \text{ y } C(a'Sa) \cup C(aPa') \subset F \qquad (2.18)$$

Y enfatiza que: $C(a'Sa) \cup C(aPa') \neq F$

2.5.2.5 Criterio de Preferencia Débil.

La unión de las dos coaliciones no es igual a F y solo existe un criterio que no es ni concordante ni discordante con la afirmación $a'Sa$ y es el caso donde:

Si y solo si $g_j(a) - p_j \leq g_j(a') < g_j(a) - q_j,\ (p_j > q_j)$

El subconjunto de F que satisface la última condición esta denotada por $C(aQa')$. El j-esimo criterio pertenece a $C(aQa')$ si y solo si aQ_ja', la relación binaria Q_j modela la preferencia débil restringida al j-esimo criterio. Con lo mencionado anteriormente cada par ordenado (a, a') se asocia a una partición de F en tres subconjuntos [46].

$$C(a'Sa), C(aPa'), C(aQa') \qquad (2.19)$$

2.5.3 Índice de Credibilidad: Determinando la Solidez de $a'Sa$

Con lo mostrado en la sección anterior sabemos que al afirmar que la opción a' es al menos tan buena como a existen criterios que están en concordancia, discordancia y en ninguno de los dos anteriores. Por lo cual es necesario poder calcular el índice de credibilidad $\sigma(a', a)$. El índice de credibilidad mide la solidez de la afirmación "a' es al menos tan buena como a" [47]. El valor de $\sigma(a', a)$ es fundamental para determinar las relaciones de outranking. En los métodos ELECTRE, la confirmación de la afirmación $a'Sa$ depende de las siguientes dos condiciones: (i). si la unión de los criterios a favor de tal aseveración es lo suficientemente fuerte para apoyarla; este valor se conoce como índice de concordancia y se representa como $c(a', a)$; y (ii) si al aprobar tal aserción no se genera un desacuerdo bastante fuerte por parte de los criterios que no están de acuerdo con dicha afirmación; este valor se conoce como índice de discordancia y se expresa como $d_j(a', a)$.

2.5.4 Índice de Concordancia

Para calcular el índice de concordancia tenemos:

$$c(a', a) = \frac{\sum k_j c_j(a', a)}{\sum k_j} \tag{2.20}$$

donde

$$c_j(a', a) = \begin{cases} 1, & \text{si } a' S_j a \\ 0, & \text{si } a P_j a' \\ \frac{p_j + g_j(a') - g_j(a)}{p_j - q_j}, & \text{en otro caso} \end{cases} \tag{2.21}$$

2.5.4.1 Índice de Discordancia

El índice de discordancia para cada criterio g_j se define como:

$$d_j(a', a) = \begin{cases} 0, & \text{,si } \nabla_j(a', a) \leq p_j \\ \frac{\nabla_j(a', a) - p_j}{v_j - p_j} & \text{,si } p_j < \nabla_j(a', a) < v_j \\ 1 & \text{,si } \nabla_j(a', a) > v_j \end{cases} \tag{2.22}$$

donde

$$\nabla_j(a', a) = g_j(a) - g_j(a') \tag{2.23}$$

Con los índices de concordancia (2.21) y discordancia (2.22) podemos ahora calcular el índice de credibilidad (2.20):

$$\sigma(a', a) = \begin{cases} c(a', a), \ si\ d_j(a', a) \leq c(a', a) \\ c(a', a) \cdot \prod_{d_j(a', a) > c(a', a)} \frac{1 - d_j(a', a)}{1 - c(a', a)} \end{cases} \tag{2.24}$$

2.5.5 Relaciones de Outranking

El modelo de relaciones de outranking propuesto por Fernández et al. [48] está basado en las relaciones de preferencia planteadas por Roy [49]. El modelo determina la calidad de las soluciones utilizando la dominancia de Pareto y las relaciones de outranking, las cuales proporcionan una mayor capacidad discriminatoria. La relación de superación entre un par de soluciones establece el grado de credibilidad de la afirmación "*x* es al menos tan buena como *y*", denotado por *σ(x,y).* Este valor puede calcularse mediante algunos métodos reportados en la literatura, por ejemplo, ELECTRE ([49], [50]) y PROMETHEE [51]. También con las relaciones de outranking se generan relaciones más complejas como las de

Fernández et al. [48]. El modelo establece para cada par de soluciones (x, y) alguna de las relaciones de preferencia listadas:

1. **Preferencia estricta (*xPy*).** Representa la situación donde el DM tiene claras y bien definidas razones para justificar la elección de x sobre y. Para que se establezca una relación de preferencia estricta se debe cumplir al menos una de sus condiciones;
2. **Indiferencia (*xIy*).** Esta relación representa la situación en la cual el DM tiene claras y positivas razones que justifican una equivalencia entre x e y. La indiferencia se cumple cuando las dos condiciones que la representan se satisfacen;
3. **Preferencia débil (*xQy*).** Modela la situación donde el DM duda entre xPy y xIy. Esta relación es consecuencia de la conjunción de las condiciones que la representan;
4. **Incomparabilidad (*xRy*).** Simula la situación en la cual el DM percibe un alto grado de heterogeneidad entre x e y, pero no puede expresar una preferencia a favor de alguna de ellas;
5. **k-preferencia (*xKy*).** Representa la situación donde el DM duda entre xPy y xRy. Esta relación se cumple si las tres condiciones que la identifican se satisfacen.

2.6 Conceptos de Imprecisión e Incertidumbre

Según Smets [52] la incertidumbre es una propiedad que resulta de la falta de información acerca del mundo para decidir si una afirmación es verdadera o falsa. La incertidumbre es el conocimiento parcial del verdadero valor de los datos. La incertidumbre conduce a la ignorancia. Esto es esencialmente, si no es que siempre, una propiedad epistémica causada por la falta de información. La mayor causa de incertidumbre es la imprecisión. Según Smets [52] la imprecisión es una propiedad de la información misma. La imprecisión puede ser caracterizada por una presencia o ausencia de un error. Algunos aspectos de la imprecisión sin error son información: ambigua, aproximada, difusa, perdida e incompleta. Algunos aspectos de la imprecisión con error son la información: incorrecta, inexacta, inválida, distorsionada, sesgada, sin significado y sin sentido. Smets [52] menciona que la imprecisión es un componente de la información, mientras que la incertidumbre resulta de no poder determinar si algo es verdadero o falso. A menudo la imprecisión causa incertidumbre. Para

ilustrar la diferencia entre incertidumbre e imprecisión, consideremos las siguientes dos situaciones:

1. Juan tiene al menos dos hijos, de eso estoy seguro.
2. Juan tiene tres hijos, pero de eso no estoy seguro.

En el primer caso, el número de hijos es impreciso, pero cierto. En el segundo caso, el número de hijos es preciso, pero es incierto. Ambos aspectos pueden coexistir, pero son diferentes. A menudo, información menos precisa puede ser establecida con más certeza e información más precisa puede ser establecida con menos certeza. Aunque los métodos de superación modelan de una forma integral las preferencias de un decisor, aún hay detalles que considerar. Uno de ellos es la incertidumbre provocada por la imprecisión de los datos. El decisor quizás no pueda determinar con precisión un valor determinado para criterios, indiferencias, incomparabilidad, umbrales, etc., lo cual le provoca incertidumbre. Pero, por otro lado, puede ser capaz de determinar intervalos que le generen más confianza y certeza. Tomando el ejemplo de los automóviles y su matriz de preferencias, el decisor tiene problemas para establecer un valor preciso de su valoración para cada atributo, los valores puntuales que ha proporcionado son para el menos certeros y que le generan incertidumbre (Tabla 2.1).

Tabla 2.1. Ejemplo de la valoración de opciones de automóviles

Opción/Criterio	**Precio**	**Estética**	**Seguridad**
Auto1	230000	7	7
Auto2	240000	9	8
Auto3	260000	8	9

La tabla 2.2 podemos observar los pesos que ha designado para el coeficiente de importancia en formato de intervalo. Esto con el fin de establecer pesos que reflejen la imprecisión del DM, pero que le dan más certeza de que el proceso llegara a un mejor resultado.

Tabla 2.2. Se establece el valor de k_j con un formato de intervalo.

Opción/Criterio	**Precio**	**Estética**	**Seguridad**
$\boldsymbol{k_j}$	[0.6, 0.8]	[0.85, 0.9]	[1, 1]

En la sección 2.6.1 del presente documento se describe la forma en que se maneja el modelo de intervalos el cual tiene una serie de operaciones aritméticas y lógicas que permite tener un manejo sencillo de la imprecisión. Esto permite al decisor establecer los intervalos en las cuestiones donde tengo incertidumbre. Por ejemplo, los parámetros que utiliza la técnica de outranking ELECTRE. Pero esto no está limitado solo a esto, puede utilizarse también para describir el problema de optimización y parámetros imprecisos de algoritmos de optimización ya establecidos.

2.6.1 Intervalos

Un número de intervalo es un rango de valores que se encuentran entre dos límites. Los números de intervalo son una manera fácil de modelar la imprecisión derivada de mediciones inexactas o la variabilidad de los juicios y creencias del DM. Los números de intervalo son una extensión de los números reales y un subconjunto $\mathbb{R}$ [53]. La representación de los números de intervalo son en negrita y cursiva, por ejemplo, $E = [\underline{E}, \overline{E}]$ donde $\underline{E}$ y $\overline{E}$ corresponden a los límites superior e inferior. Entre varias operaciones para manejar números de intervalo, solo subrayamos la operación de suma (aritmética) y las relaciones de orden, definidas a continuación. Sean $D = [\underline{D}, \overline{D}]$ y $E = [\underline{E}, \overline{E}]$ dos números de intervalo. Las operaciones aritméticas básicas se definen como:

$$D + E = [\underline{D} + \underline{E}, \overline{D} + \overline{E}] \tag{2.25}$$

$$D - E = [\underline{D} + \underline{E}, \overline{D} - \overline{E}] \tag{2.26}$$

$$D \cdot E = [\min\{\underline{D} \cdot \underline{E}, \underline{D} \cdot \overline{E}, \overline{D} \cdot \underline{E}, \overline{D} \cdot \overline{E}\}, \max\{\underline{D} \cdot \underline{E}, \underline{D} \cdot \overline{E}, \overline{D} \cdot \underline{E}, \overline{D} \cdot \overline{E}\}] \tag{2.27}$$

Si D es un número real:

$$D \cdot E = [D \cdot \underline{E}, D \cdot \overline{E}] \tag{2.28}$$

2.7 Estado del Arte de los Frameworks de Hiperheurísticas

Los frameworks de hiperheurísticas son herramientas fundamentales para el diseño y la comparación de estrategias de optimización que combinan diferentes heurísticas. Estos frameworks proporcionan una plataforma estandarizada que facilita el desarrollo de algoritmos complejos, permitiendo la implementación y comparación de heurísticas de bajo nivel sin preocuparse por los aspectos específicos del problema. En este estado del arte, se presentan los frameworks más relevantes utilizados en el desarrollo de hiperheurísticas, junto con una tabla comparativa que resume sus principales características.

2.7.1 Descripción de los Frameworks de Hiperheurísticas

Hyperion

Hyperion es un framework orientado a la implementación de hiperheurísticas recursivas que permite la selección y generación adaptativa de heurísticas de bajo nivel (LLHs). Su principal característica es el uso de mecanismos recursivos que adaptan la selección de heurísticas en función del rendimiento obtenido en iteraciones previas [54]. Este enfoque le permite refinar sus decisiones heurísticas y adaptarse dinámicamente a cambios en el problema, lo cual es beneficioso en entornos donde la naturaleza del problema es cambiante.

HyFlex

HyFlex fue desarrollado como una plataforma flexible para implementar y evaluar hiperheurísticas a través de diversos dominios de problemas de optimización combinatoria [55]. Diseñado inicialmente para el Cross-domain Heuristic Search Challenge (CHeSC) en 2011, HyFlex ofrece una interfaz común que facilita la comparación y evaluación de estrategias de hiperheurísticas sin necesidad de conocimientos profundos sobre los problemas específicos [56]. HyFlex ha sido aplicado a problemas como MAX-SAT, Bin Packing, TSP y programación de horarios de personal.

ParHyFlex

ParHyFlex es una extensión de HyFlex que incorpora capacidades de procesamiento paralelo para mejorar la eficiencia en la selección y aplicación de heurísticas [57]. Al permitir la evaluación simultánea de diferentes estrategias heurísticas.

EvoHyp

EvoHyp se centra en la aplicación de algoritmos evolutivos para la generación y selección de heurísticas. Este framework permite automatizar la composición de hiperheurísticas adaptativas, mejorando el proceso de búsqueda mediante la evolución continua de heurísticas en función de su rendimiento [58] EvoHyp es ideal para problemas de optimización combinatoria y permite explorar un espacio de búsqueda amplio con principios evolutivos.

hMod

hMod enfatiza la modularidad y flexibilidad para integrar componentes heurísticos. Este framework permite ensamblar algoritmos de hiperheurísticas de forma detallada y utiliza un enfoque orientado a objetos que facilita la experimentación con diferentes combinaciones de heurísticas [59]. La modularidad del framework permite una adaptación rápida a las características del problema y facilita el diseño detallado de hiperheurísticas.

HH-DSL

HH-DSL utiliza un lenguaje específico de dominio (DSL) que permite definir y aplicar hiperheurísticas de manera intuitiva, simplificando el proceso de programación y reduciendo la complejidad del desarrollo de hiperheurísticas [60]. Este framework facilita el prototipado rápido y la experimentación, siendo ideal para investigadores que desean probar diferentes estrategias sin necesidad de profundizar en los aspectos técnicos de bajo nivel [61].

MatHH

MatHH es un framework que integra técnicas matemáticas para optimizar la selección y aplicación de heurísticas [62]. Utiliza modelado matemático para mejorar el rendimiento de los algoritmos de hiperheurísticas, facilitando la selección de heurísticas más adecuadas para problemas específicos. La arquitectura modular de MatHH permite una fácil integración de nuevos operadores matemáticos, lo que lo hace flexible para diferentes tipos de problemas.

Presente investigación

EL HH-Framework de la presente investigación se centra en la creación y aplicación de hiperheurísticas adaptativas para problemas continuos y binarios. Utiliza un proceso de selección basado en métricas de la población, como la métrica PCArea desarrollada en esta investigación, que ha demostrado buen rendimiento frente a otras métricas como IGD y HV. Este framework permite la integración de preferencias del decisor. Además, el framework aborda problemas de gran escala mediante coevolución y descomposición en subproblemas, haciéndolo altamente adaptable y versátil para diferentes tipos de optimización.

Ventajas de HH-Framework frente a otros frameworks:

- Muchos objetivos: A diferencia de la mayoría de los otros frameworks, HH-Framework maneja eficientemente problemas con muchos objetivos mediante la descomposición en subproblemas y el esquema de Tchebycheff.
- Gran escala: HH-Framework utiliza técnicas de coevolución para dividir los problemas en subproblemas más manejables, lo que lo hace efectivo para problemas de gran escala, a diferencia de frameworks como HyFlex y MatHH que no se centran en esta característica.
- Preferencias del decisor: Ofrece una integración directa de las preferencias del decisor, utilizando operadores de selección basados en preferencias, lo cual no está presente en otros frameworks, destacando su utilidad en aplicaciones personalizadas.
- Manejo de imprecisión: A diferencia de otros frameworks que requieren valores exactos, HH-Framework puede manejar la imprecisión usando intervalos, lo cual es útil para problemas donde los datos son inciertos o imprecisos.

La Tabla 2.3 muestra una comparación entre los distintos frameworks de hiperheurísticas encontrados en el estado del arte y el presente trabajo. Esta tabla destaca las características del HH-Framework frente a los demás en características como muchos objetivos, gran escala, preferencias e imprecisión.

Tabla 2.3. Tabla comparativa del estado del arte.

Framework	Descripción General	Aplicaciones y Casos de Uso	Característica Destacada	Arquitectura Modular	Muchos Objetivos	Gran Escala	Preferencias	Imprecisión	Referencias
Hyperion	Herramienta para desarrollar hiperheurísticas recursivas.	Problemas de optimización combinatoria.	Mecanismo recursivo para refinamiento adaptativo.	Sí	No	No	No	No	Swan et al. (2011) [54]
HyFlex	Plataforma flexible para implementar y evaluar hiperheurísticas en múltiples dominios.	MAX-SAT, Bin Packing, TSP, VRP, etc.	Soporte multi-dominio para benchmarking.	Sí	No	No	No	No	Ochoa et al. 2012 [55]
HH-DSL	Framework que usa un lenguaje específico de dominio (DSL) para facilitar la creación de hiperheurísticas.	Problemas de programación de horarios, ruteo y asignación.	DSL para simplificar la programación y prototipado rápido de heurísticas.	Sí	No	No	No	No	Cora et al. (2013) [60]
ParHyFlex	Extensión de HyFlex con capacidades de procesamiento paralelo.	Programación de horarios, problemas de asignación.	Exploración simultánea de estrategias heurísticas.	Sí	No	No	No	No	Onsem &Demoen (2013) [57]
EvoHyp	Framework especializado en el uso de algoritmos evolutivos.	Problemas de optimización combinatoria (COPs).	Integración de principios evolutivos.	Sí	No	No	No	No	Pillay & Beckedahl (2017) [58]
hMod	Framework orientado a la modularidad para integración flexible de heurísticas.	Problemas de programación de horarios, ruteo y asignación.	Implementación orientada a objetos para facilitar la construcción de algoritmos detallados.	Sí	No	No	No	No	Urra et al. (2019) [59]
MatHH	Framework basado en técnicas matemáticas para resolver problemas de optimización.	Programación de horarios, problemas de ruteo.	Integración de técnicas matemáticas en la selección de heurísticas.	Sí	No	No	No	No	Cruz-Duarte et al. 2022 [62]
HH-Framework	Herramienta para la construcción y aplicación de hiperheurísticas adaptativas para optimización continua y binaria.	Problemas continuos (DTLZ, WFG) y problemas binarios (PSP).	Selección adaptativa de LLHs mediante métrica innovadora (PCArea), integración de DE.	Sí	Sí	Sí	Sí	Sí	Presente investigación

Capítulo 3

3 Metodología

En este capítulo se describe la metodología utilizada en la presente investigación para construir un framework de software para hiperheurísticas (HH-Framework). El capítulo está organizado de la siguiente forma: Descripción de la metodología del proyecto de investigación (sección 3.1), el framework de software para hiperheurísticas y su arquitectura (sección 3.2). A partir de la sección 3.3 se describen algoritmos principales que se encuentran en el HH-Framework, por ejemplo: Indicadores (sección 3.4), heurísticas de bajo nivel (sección 3.5) y finalmente los algoritmos hiperheurísticos (sección 3.6).

3.1 Descripción de la Metodología

En el ámbito de la optimización y la resolución de problemas complejos, las hiperheurísticas han surgido como una poderosa herramienta para abordar problemas de diversa naturaleza. Estas técnicas, que operan a un nivel superior a las heurísticas convencionales, ofrecen la capacidad de adaptarse dinámicamente a diferentes contextos y problemáticas, brindando soluciones eficientes y flexibles. Sin embargo, el diseño y la implementación de hiperheurísticas efectivas requieren un enfoque estructurado y sistemático. Es en este contexto que surge la necesidad de desarrollar frameworks específicos que guíen el proceso de creación y evaluación de estas técnicas. Un framework de hiperheurísticas proporciona un marco de trabajo que facilita la conceptualización, el desarrollo y la evaluación de nuevas estrategias de búsqueda.

En la figura 3.1 se observa el diagrama a bloques que describe de manera detallada los pasos fundamentales para la creación del framework de hiperheurísticas HH-Framework. Cada etapa del proceso juega un papel fundamental en la construcción de un marco sólido y efectivo para el desarrollo de nuevas técnicas de optimización basadas en hiperheurísticas.

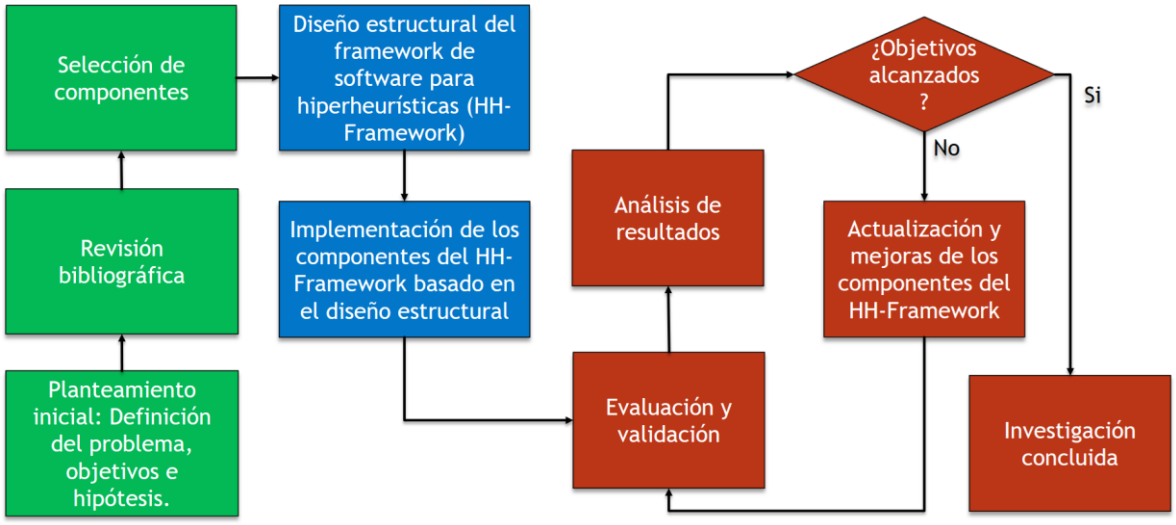


Figura 3.1. Diagrama a bloques de la metodología de investigación, etapa inicial (verde), etapa intermedia (azul) etapa final (marron).

A continuación, se presenta una breve descripción de los pasos incluidos en el diagrama:

Etapa inicial. En esta etapa inicial, se identificó y especificó claramente el problema de investigación, los objetivos que se buscaban en la investigación y la hipótesis de investigación:

- **Revisión bibliográfica.** Se llevó a cabo una exhaustiva búsqueda de la literatura relacionada con hiperheurísticas y técnicas de optimización similares. De esta revisión surgieron componentes básicos que poseen las hiperheurísticas y que sirvieron para el diseño de clases y estructuras de datos con el fin de aumentar la productividad en el momento de generación de nuevas hiperheurísticas.
- **Selección de componentes.** En esta fase, se eligen los componentes básicos que conformarán el framework de hiperheurísticas, como operadores de búsqueda, funciones de evaluación y métodos de selección.

Etapa intermedia. En esta etapa basado en análisis previo se propuso el diseño estructural del framework y la estrategia de implementación:

- **Diseño estructural del framework de software para hiperheurísticas.** Se definió la arquitectura y la estructura general del framework, especificando cómo interactúan los diferentes componentes y cómo se integran en un sistema coherente basado en principios de la programación orientada a objetos (P.O.O).
- **Implementación.** En esta etapa se procede a la implementación del framework, desarrollando los algoritmos (heurísticas de bajo nivel e hiperheurísticos) y funcionalidades necesarias. Posteriormente, se realizan pruebas de software necesarias para identificar posibles errores de código y funcionamiento.

Etapa final. Una vez que se tuvo el framework implementado se utilizó una metodología de investigación y desarrollo iterativa, entregando en cada iteración una versión funcional del framework en forma incremental:

- **Evaluación y validación.** En esta etapa se evalúa los algoritmos hiperheurísticos generados en el framework utilizando conjuntos de datos de prueba y comparándolos con sus heurísticas de bajo nivel y otros algoritmos. Esta evaluación proporciona información valiosa sobre la eficacia y la generalidad del framework en diferentes dominios de aplicación.
- **Análisis de resultados.** En esta etapa se analizan los resultados obtenidos y se evalúa si han aportado al cumplimiento de los objetivos planteados en la investigación. Si estos resultados cubren todos los objetivos se puede dar por concluida la investigación, en caso contrario se debe trabajar en los objetivos restantes, y en las mejoras e innovaciones encontradas en el proceso de investigación y actualizar el framework.
- **Actualización y mejoras.** Si aún no se alcanzan los objetivos, se deben generar nuevos componentes para el framework que puede incluir nuevos algoritmos de hiperheurísticos LLH, además de componentes necesarios que permitan la interacción entre las distintas clases. Una vez hechas las mejoras y las actualizaciones en el framework se pueden volver a evaluar y validar con diseños experimentales regresando a la evaluación y validación del framework.

3.2 Framework de Software para Hiperheurísticas (Hyper-Heuristic Framework)

Los algoritmos con un enfoque hiperheurístico consideran la búsqueda de las mejores heurísticas que puedan resolver un problema. Según Burke et al. [64] una forma simple de entenderlo es la mostrada en el Algoritmo 3.1:

Algorithm 3.1: Algoritmo Hiperheurístico simple

```
1 if problemType(P)==p1 then
2 |  apply(heuristic1,P)
3 else
4 |  if problemType(P)==p2 then
5 |  |  apply(heuristic2,P)
6 |  else
7 |  |  ...
8 |  end if
9 end if
```

Como lo plantea [64] si un algoritmo es capaz de saber a qué tipo de problema se enfrenta, lo más adecuado sería utilizar una estrategia (algoritmo, heurística, etc.) que pueda tratar con él, que tenga buenos resultados o fortalezas ante este. Es decir, no busca la mejor solución para el problema, sino el mejor algoritmo. Mientras que el enfoque metaheurístico se lleva a cabo en el dominio de las soluciones, el hiperheurístico lo hace en el dominio de las (meta) heurísticas. Al momento de crear algoritmos, el poder tener un marco referencia o reutilizar componentes de software que ya hayan sido probados y utilizados permite la creación de nuevos algoritmos de una forma más ágil. Por lo cual, contar con Framework de Software para la programación de hiperheurísticas permite al investigador concentrarse en lo más importante al momento de proponer y llevar a cabo este tipo de algoritmos. Con eso en mente se ha diseñado una arquitectura de software para la creación de un Framework de Software para Hiperheurísticas llamado Hyper-Heuristic Framework (HH-Framework). La propuesta tiene las siguientes características técnicas:

Lenguaje de programación. El lenguaje de programación para codificar el HH-Framework es el lenguaje C++.

Paradigma de programación. El paradigma utilizado es la programación orientada a objetos (POO) en conjunto con los conceptos de S.O.L.I.D.

Problemas. HH-F considera los siguientes problemas estándar y específicos como parte del Framework:

- DTLZ Benchmark. Este benchmark que contiene 9 problemas de tipo continuo (DTLZ1, DTLZ2, …, DTLZ9) que tienen la cualidad de ser escalables tanto en variables de decisión como en objetivos. Siete de ellos no manejan restricciones, mientras que dos de ellos si las manejan. HH-F solo considerará los problemas sin restricciones DTLZ1 hasta DTLZ7 [63].
- WFG Benchmark. El conjunto de problemas WFG (de Walking Fish Group) es un conjunto de benchmarks diseñado para evaluar algoritmos de optimización multi-objetivo. Estos problemas son utilizados comúnmente en la investigación para evaluar y comparar el rendimiento de algoritmos de optimización multi-objetivo en términos de convergencia y diversidad de soluciones. HH-F considera como problemas objeto de estudio los problemas de tipo continuo WFG1 hasta WFG9 [65].
- Problema de selección de proyectos (PSP). Problema de optimización de tipo combinatorio que está sujeto a una serie de restricciones y puede ser mono y multi-objetivo, y de gran escala [26].

Población. Representan las soluciones de la hiperheurística.

Método de selección. La forma en que se elige la LLH que se aplicará.

Heurísticas de bajo nivel (LLH). Algoritmos que se encuentran en el nivel del problema y que son utilizados por la hiperheurística. HH-Framework considera las siguientes heurísticas de bajo nivel (LLHs) como parte del framework las cuales tienen cualidades para enfrentar algunas características presentes en los problemas (preferencias, incertidumbre, muchos objetivos y gran escala):

- MOSA/D
- MOSA/D-O
- MOSA/D-O-II
- CO-MOSA/D
- MOSA/D-PSP
- MOSA/D-O-PSP

- MOSA/D-O-II-PSP
- CO-MOSA/D-PSP
- FDEA-I
- FDEA-II

Conjunto de LLHs. Todas las heurísticas contempladas para resolver el problema.

Indicador de desempeño de las poblaciones. El uso de indicadores permite medir el trabajo realizado por cada LLH. HH-F considera para análisis del desempeño de las poblaciones de LLHs:

- IGD – Distancia generacional invertida
- AverageDominance – Dominancia promedio de una población sobre otra.
- PCArea – Parallel cordinate area
- PR – Región de preferencia.
- HV – Hipervolumen.
- Distancia euclidiana (Minima, Mediana y Promedio)
- Distancia Tchebicheff (Minima, Mediana y Promedio)

Interacción entre clases. Las distintas clases tienen métodos estandarizados para interactuar: initialize(), execute(), get(), set() son algunos ejemplos de métodos que permitirán la interacción entre las instancias creadas a partir de las clases del Framework.

3.2.1 Arquitectura del Hyper-Heuristic Framework

La arquitectura (figura 3.1) propuesta del Framework de Software para Hiperheurísticas (HH-Framework) está formada por una serie de clases abstractas que deben ser implementadas para generación de nuevas Hiperheurísticas. Maashi et al. [66] mencionan que: por lo general, en un marco hiperheurístico de selección, existe una clara separación entre el enfoque hiperheurístico de alto nivel, también denominado estrategia, y el conjunto de heurísticas de bajo nivel o componentes heurísticos. Esta característica es respetada en la arquitectura dado que las clases derivadas de Hyperheuristic no tienen conocimiento de las operaciones que llevan a cabo las clases derivadas de Algorithm. Por ejemplo, la hiperheurística HH-DTLZ (Figura 3.3) solo puede acceder a las heurísticas de bajo nivel como por ejemplo MOSA/D a través de sus funciones públicas como lo es execute(). A

continuación, se describe en forma breve las clases principales que conforman la arquitectura de HH-Framework.

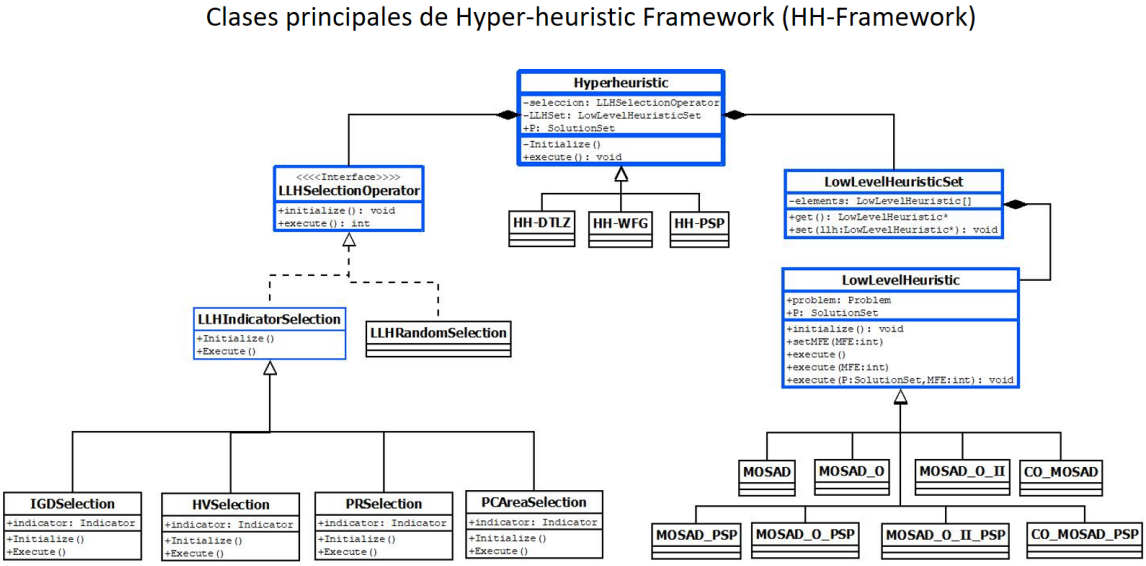


Figura 3.2. Diagrama de clases de la arquitectura general propuesta del HH-Framework

La clase `Hyperheuristic`. Esta es una clase (figura 3.3) de la cual se pueden derivar nuevas clases de Hiperheurísticas con las características de poseer: i) un operador de selección, ii) un conjunto de LLHs, y iii) una población. Es la clase principal del HH-Framework y tiene interacción directa con la clase `LLHSelectionOperator` y `LowLevelHeuristicSet`, además de interacción indirecta con la clase `LowLevelHeuristic`.

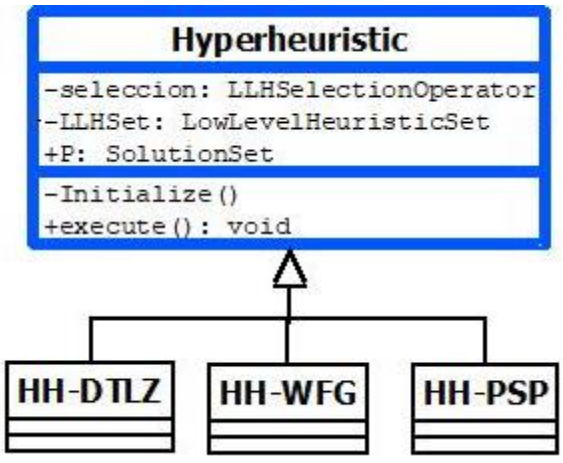


Figura 3.3. Clase de alto nivel Hyperheuristic

La clase `LLHSelectionOperator`. Esta es una clase (figura 3.4) abstracta que permite la implementación de un método para seleccionar entre distintas LLHs. Al ser derivable, cada desarrollador puede implementar el método de selección que sea más apropiado para el algoritmo que está desarrollando.

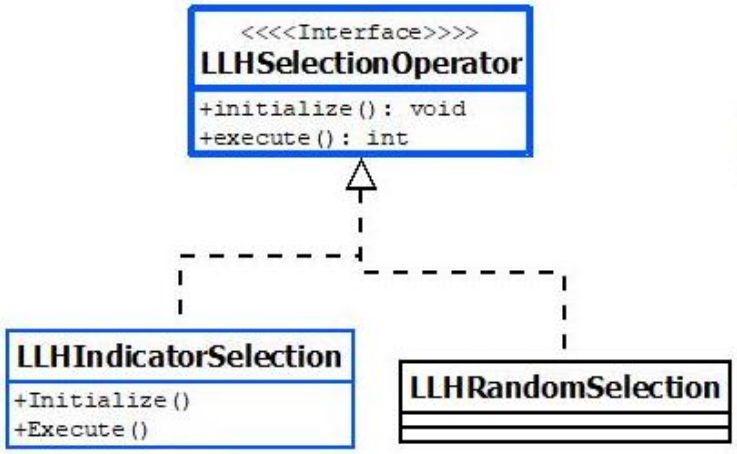


Figura 3.4. Clase LLHSelectionOperator

La clase `HHIndicatorSelection`. Esta es una clase (figura 3.5) abstracta que permite derivar métodos de selección basados en indicadores de desempeño de una población de soluciones dadas. Por ejemplo, indicadores como lo es la distancia generacional invertida (IGD) y el Hipervolumen (HV) pueden ser implementados.

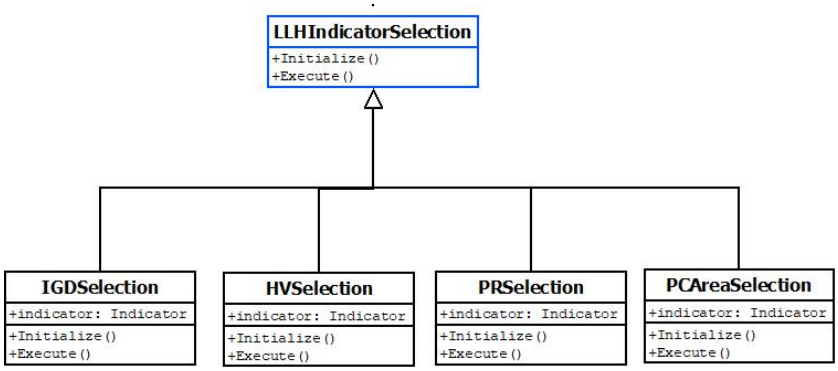


Figura 3.5. Clase de alto nivel HHIndicatorSelection

La clase `LowLevelHeuristicSet`. Esta clase (figura 3.6) sirve como almacenamiento para a una o más LLHs que serán utilizadas por las hiperheurísticas creadas en el framework. Esta facilita para la hiperheurística la gestión de todas las LLHs disponibles en tareas como: carga a memoria, ejecución, y acceso a la población de soluciones.

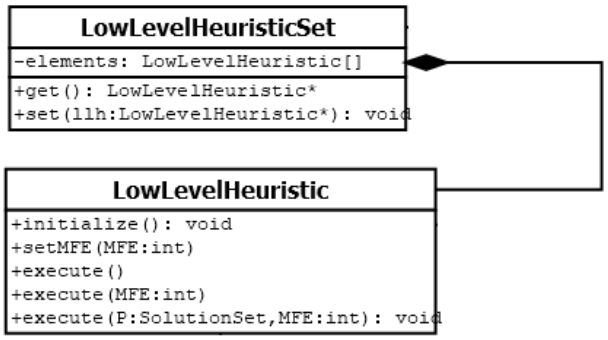


Figura 3.6. Clase de bajo nivel LowLevelHeuristic

Los algoritmos metaheurísticos en el contexto de una hiperheurística son conocidos como heurísticas de bajo nivel. La clase LowLevelHeuristic (figura 3.7) permite poder desarrollar las metaheurísticas (LLHs) que pueden ser utilizadas por el Framework de Hiperheurísticas. La clase LowLevelHeuristic deriva de la clase Algorithm que originalmente pertenece al Framework MS-DOSS (desarrollado por el cuerpo de investigación). Por tal motivo, toda metaheurística desarrollada en MS-DOSS es altamente reutilizable por el HH-Framework. Ejemplos de estas metaheurísticas son MOSA/D, MOSA/D-O, MOSA/D-O-II y CO-MOSA/D entre otras.

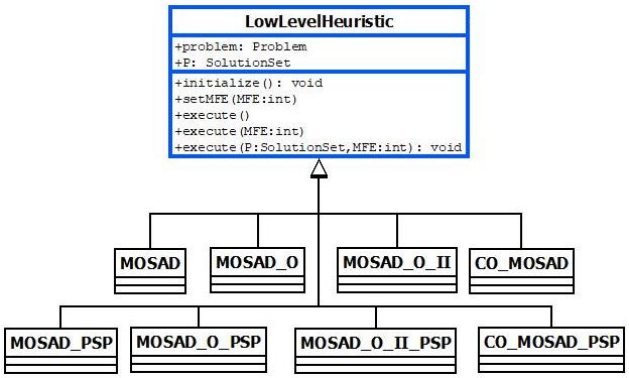


Figura 3.7. La clase LowLevelHeuristic y sus clases derivadas.

Una vez creados los algoritmos necesarios y los mecanismos del HH-Framework es necesario configurar la ejecución mediante archivos de configuración. Por ejemplo, para el algoritmo HH-DTLZ se consideran los siguientes archivos de configuración. En este ejemplo HH-DTLZ (ver figura 3.8) está configurado para evaluarse 100000 con el fin de resolver el problema PSP1 con 3 objetivos. LLHSize indica el número de LLHs a utilizar y LLHSet cuáles son los archivos de configuración de las LLHs seleccionadas.

```
Archivo    Editar    Ver

#Hyperheuristic HH_PSP
#maxEvaluations 100100
#N 100
#alpha 1000
#beta 6000

#Problem-Instance _INPUT/PSP/N03-I01-PSP1.txt

#LLHSize 4
#LLHSet
Heuristics/Config-MOSAD_DE_PSP.txt
Heuristics/Config-MOSAD_O_PSP.txt
Heuristics/Config-MOSAD_O_II_PSP.txt
Heuristics/Config-CO_MOSAD_PSP.txt

#LLHSelectionOperator LLHIndicatorSelectionIGD
```

Figura 3.8. Archivo configuración de la Hiperheurística.

3.3 Indicadores de Desempeño.

Como parte de la investigación, se propuso la creación de indicadores de desempeño que midan el desempeño de los algoritmos en las hiperheurísticas.

3.3.1 PCArea: Parallel Coordinate Area.

El parallel coordinate plot (figura 3.9) es una herramienta utilizada para visualizar poblaciones de soluciones en problemas de múltiples y muchos objetivos. Aunque su enfoque es meramente gráfico y descriptivo.

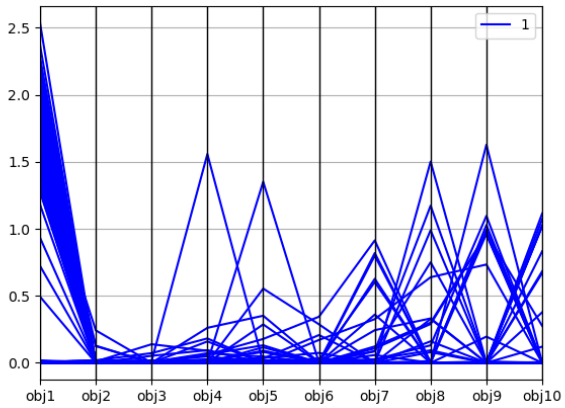


Figura 3.9. Parallel coordinate plot de una población de soluciones obtenida mediante MOSA/D-DE para el problema DTLZ4 con 10 objetivos.

Los problemas de muchos objetivos tienen el problema de poseer una gran cantidad de soluciones no dominadas y comparar soluciones se vuelve una tarea complicada [67]. Una

estrategia utilizada para enfrentar este tipo de problemas es usar algunos tipos de dominancia relajada como por ejemplo la dominancia fraccionada [68].

Alternativamente, un posible uso del parallel cordinate plot es considerarlo una fuente de información. Si se toma el polígono que describe cada solución graficada como un área y esta es calculada, obtenemos información interesante. Si consideramos un problema de minimización y contamos con dos poblaciones que se desean comparar como las que se muestran en la figura 3.10, a simple vista podemos observar que la población a) tiende a tener un mayor número de soluciones que minimizan el valor de sus objetivos que la población b).

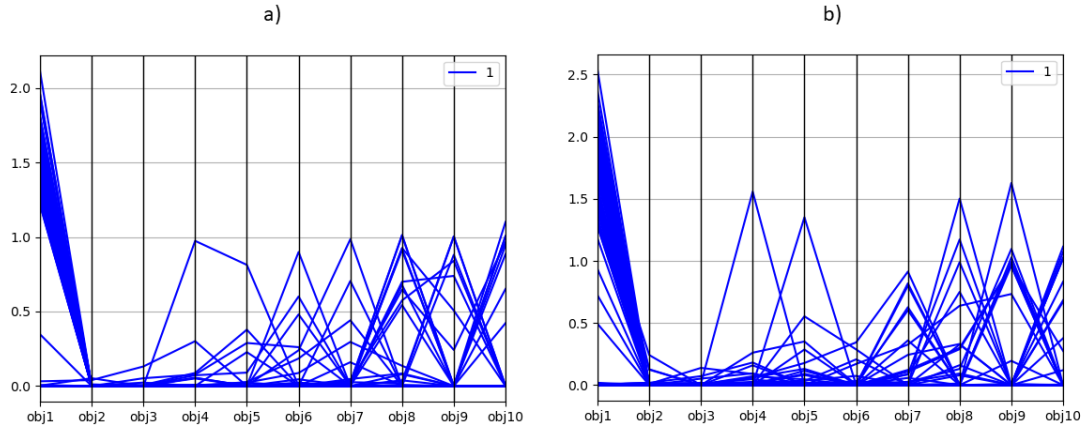


Figura 3.10. Parallel coordinate plot de dos poblaciones del problema DTLZ4 con 10 objetivos obtenidas por a) MaOLS-HH-UP y b) MOSA/D-DE.

Por lo cual se propone la métrica PCArea basada en el gráfico parallel cordinate plot, donde esta métrica obtiene el área de todas las soluciones de una población y calcula su promedio definiéndola como sigue:

$$PCArea = \frac{\sum_i^n area(s_i)}{n}$$

Donde s_i es una solución de la población S, $n = |S|$ y $i = 1,2,3, \dots, n$.

El resultado de calcular PCArea muestra la convergencia de las soluciones, pero no muestra que tan diversa es, por ejemplo, puede calcular el área de dos poblaciones donde el resultado es similar, pero no muestra cómo se distribuye el área en los distintos objetivos, lo que podría ser valioso para tomar una decisión.

3.4 Heurísticas de Bajo Nivel (LLHs) desarrolladas.

El presente trabajo utiliza como base el Framework MS-DOSS que ha servido de modelo para implementar las Heurísticas de Bajo Nivel (LLHs) (Tabla 3.1 y 3.2). La estrategia para generar nuevas LLHs es que integren al menos intervalos, descomposición, outranking y coevolución. Un ejemplo de LLH producto de la presente investigación es MOSA/D [69]. Esta LLH combina Recocido Simulado (SA), Descomposición (D) y Evolución Diferencial (DE). Las LLHs contempladas son las que se encuentran mencionadas en la Tabla 3.1 y 3.2. Cabe mencionar que estas LLHs fueron creadas en esta investigación y algunos de los resultados que se obtuvieron ya fueron publicados.

Tabla 3.1. LLHs enfocados en los problemas estándar DTLZ y WFG.

Basado en descomposición	Basado en descomposición y outranking	Basado en descomposición y coevolución cooperativa
MOSA/D	MOSA/D-O	CO-MOSA/D
	MOSA/D-O-II	

Tabla 3.2. LLHs enfocados en el problema de selección de proyectos (PSP).

Basado en descomposición	Basado en descomposición y outranking	Basado en descomposición y coevolución cooperativa
MOSA/D-PSP	MOSA/D-O-PSP	CO-MOSA/D-PSP
	MOSA/D-O-II-PSP	

3.4.1 MOSA/D - Recocido Simulado Multi-objetivo Basado en Descomposición

El algoritmo de Recocido Simulado Multi-objetivo Basado en Descomposición (MOSA/D) toma como base el trabajo de Engrand [70] y la estrategia de descomposición de MOEA/D (Zhang et al [35]). Básicamente MOSA/D toma un problema de optimización multi-objetivo (MOP) y lo descompone en un conjunto de N sub-problemas, representado por un conjunto de N vectores de peso. Cada sub-problema se recose mediante L ejecuciones a un nivel de temperatura T, obteniendo nuevas soluciones mediante {operaciones de cruza y muta de evolución diferencial y estas son evaluadas mediante comparación de la función de Tchebycheff de la solución candidata y la solución P_i asociada a i-ésimo sub-problema (v_i).

Algorithm 3.2: MOSA/D

Input: $MOP, Ti, \alpha, L, Tf, N, MFE$
Output: Population P

1 Initialize $FE, P, v, S_{current}, S_{cand}$;
2 **while** $Ti \geq Tf$ & $FE \leq FME$ **do**
3 **for** $i \leftarrow 1$ *to* N **do**
4 $S_{current} \leftarrow P_i$;
5 **for** $j \leftarrow 1$ *to* L **do**
6 $S_{cand} \leftarrow perturbation(S_{current})$;
7 $bp \leftarrow boltzmannProbability(S_{cand}, S_{current}, Ti)$;
8 **if** $g((S_{cand}, v_i, z) \leq g((P_i, v_i, z)$ **then**
9 $P_i \leftarrow S_{cand}$;
10 **end if**
11 **if** $g((S_{cand}, v_i, z) \leq g((S_{current}, v_i, z)$ *or* $bp < U(0,1)$ **then**
12 $S_{current} \leftarrow S_{cand}$;
13 **end if**
14 $z \leftarrow obtainIdealPoint(S_{cand})$;
15 **end for**
16 **end for**
17 $FE \leftarrow N * L$;
18 $Ti \leftarrow Ti * \alpha$;
19 **end while**

El pseudocódigo de MOSA/D es descrito el Algoritmo 3.2. El algoritmo recibe como entrada el problema multi-objetivo a resolver, La temperatura inicial Ti, la temperatura final Tf, el factor de tiempo α, el tamaño de la población N, y el número máximo de evaluaciones MFE que llevara a cabo. La salida está representada por la última generación de la población P. El algoritmo inicializa la población aleatoria P de tamaño N, los sub-problemas representados por vectores de peso v de tamaño N, la temperatura T = Ti, la solución $S_{current}$ que sirve de guía en la búsqueda, la solución S_{cand} la cual se genera a partir de perturbaciones sobre $S_{current}$, el cálculo del z (punto de referencia) y se actualiza FE = N esto debido a que es el tamaño de la población inicial (línea 1 y 2). Después el ciclo principal inicia (línea 3). Mientras la temperatura T sea mayor que Tf o no se alcance el número máximo de evaluaciones el ciclo principal hará lo siguiente:

1) Iniciará un segundo ciclo que recorrerá todos los sub-problemas (línea 4),
2) Actualizara FE = N × L como la multiplicación del número de sub-problemas por los ciclos de recocido (línea 16),

3) la temperatura T se actualiza al multiplicarse por el factor de descenso de temperatura α (línea 17).

El segundo ciclo (línea 4) tiene por tareas las siguientes:

1. La solución $S_{current}$ toma por valor la solución almacenada en P_i (línea 5),
2. Se recose el i-esimo sub-problema durante L ciclos (línea 6 - 14). El recocido del i-ésimo sub-problema (línea 7 -13) funciona de la siguiente forma:
    a. Se obtiene S_{cand} mediante la perturbación de $S_{current}$ (línea 7),
    b. Se calcula la probabilidad p de Boltzman de S_{cand} (línea 8),
    c. Se calcula para S_{cand} y P_i la función de Tchebycheff, si g (*Scand*, vi, z) es menor que g (Pi, vi, z) entonces S_{cand} es una mejor solución de v_i y P_i toma el valor de S_{cand} (línea 9-10),
    d. Se debe actualizar la solución guía $S_{current}$ para eso se compara la función de Tchebycheff de S_{cand} y $S_{current}$, si el resultado de la función de S_{cand} es menor que la de $S_{current}$ entonces $S_{current} = S_{cand}$, o si al generar un número aleatorio (U (0,1)) este es menor que p entonces $S_{current} = S_{cand}$.
    e. Se actualiza el punto de referencia z=obtainReferencePoint(S_{cand})
    f. Se actualiza el contador j = j + 1
1. Se actualiza el contador i = i + 1

Al terminar el ciclo principal se retorna la última generación de P y termina el algoritmo (línea 18).

3.4.2 MOSA/D-O: Recocido Simulado Multi-objetivo Basado en Descomposición y Outranking

Los algoritmos MOSA/D-O y MOSA/D-O-II utilizan el enfoque de outranking (superación) para explorar soluciones factibles para cada sub-problema producto de la descomposición. El algoritmo MOSA/D-O que se muestra en el Algoritmo 3.3 se deriva de la estructura principal MOSA/D (Algoritmo 3.2) utilizando la función de perturbación de evolución diferencial (MOSA/D-DE). MOSA/D-O integra el enfoque de superación en las líneas 8, 9, 10 y 13. En las líneas 8 y 9, se obtiene la relación xSy y xSz respectivamente. La condición de Tchebycheff se modifica en la línea 10. El objetivo de esta modificación es actualizar Pi

mediante dos condiciones: i) S_{cand} debe ser mejor que *Pi* en términos de Tchebycheff, y ii) S_{cand} debe ser al menos tan bueno como P_i en términos de outranking. En la línea 13, $S_{current}$ debe actualizarse mediante tres condiciones: i) S_{cand} debe ser mejor que $S_{current}$ en términos de Tchebycheff y ii) S_{cand} debe ser al menos tan bueno como $S_{current}$ o iii) alcanzar la condición de Boltzmann. Mientras que P mantiene las mejores soluciones de los N sub-problemas (v), $S_{currente}$ funge como solución guía de la cual surgen las nuevas soluciones y es mediante $S_{current}$ que también se agrega diversidad.

Algorithm 3.3: MOSA/D-O

Input: $MOP, Ti, \alpha, L, Tf, N, MFE$
Output: Population P

1 Initialize $FE, P, v, S_{current}, S_{cand}$;
2 **while** $Ti \geq Tf$ & $FE \leq FME$ **do**
3 **for** $i \leftarrow 1$ *to* N **do**
4 $S_{current} \leftarrow P_i$;
5 **for** $j \leftarrow 1$ *to* L **do**
6 $S_{cand} \leftarrow perturbation(S_{current})$;
7 $bp \leftarrow boltzmannProbability(S_{cand}, S_{current}, T_i)$;
8 **xSy** $\leftarrow$ **outranking**(S_{cand}, P_i);
9 **xSz** $\leftarrow$ **outranking**(S_{cand}, $S_{current}$);
10 **if** $g((S_{cand}, v_i, z) \leq g((P_i, v_i, z)$ *and* **xSy** $=$ **true then**
11 $P_i \leftarrow S_{cand}$;
12 **end if**
13 **if** $g((S_{cand}, v_i, z) \leq g((S_{current}, v_i, z)$ *and* **xSz** $=$ **true** *or* $bp < U(0,1)$ **then**
14 $S_{current} \leftarrow S_{cand}$;
15 **end if**
16 $z \leftarrow obtainIdealPoint(S_{cand})$;
17 **end for**
18 **end for**
19 $FE \leftarrow N * L$;
20 $Ti \leftarrow Ti * \alpha$;
21 **end while**

3.4.3 MOSA/D-O-II: Recocido Simulado Multi-objetivo Basado en Descomposición y Outranking II

MOSA/D-O-II (Algoritmo 3.4) se basa también MOSA/D, pero tiene algunos cambios en el mecanismo de búsqueda mostrados en el algoritmo 3.2. Por ejemplo, en las líneas 4 y 5, el algoritmo 3 toma una copia de la población P llamada *currentP*. Mientras que *P* servirá para almacenar las nuevas soluciones encontradas *currentP* servirá para obtener $S_{current}$ y los individuos aleatorios usados en la función de perturbación. MOSA/D-O-II tiene un cambio

primario en el mecanismo de búsqueda. En la línea 9, se calcula la relación de outranking xSy en seguida se revisa el criterio de Tchebycheff (línea 10). Si se cumple el criterio de Tchebycheff se ejecutan dos tareas: i) $S_{current}$ es actualizada con S_{cand} y ii) Si la solución candidata es al menos tan buena como Pi o cumple el criterio de Boltzmann (línea 12), P_i será igual a la solución candidata (Pi ← S_{cand}).

Algorithm 3.4: MOSA/D-O-II

Input: $MOP, Ti, \alpha, L, Tf, N, MFE$
Output: Population P

1 Initialize $FE, P, currentP, v, S_{current}, S_{cand}$;
2 **while** $Ti \geq Tf$ & $FE \leq FME$ **do**
3 **for** $i \leftarrow 1$ *to* N **do**
4 $currentP \leftarrow P$;
5 $S_{current} \leftarrow currentP_i$;
6 **for** $j \leftarrow 1$ *to* L **do**
7 $S_{cand} \leftarrow perturbation(S_{current})$;
8 $bp \leftarrow boltzmannProbability(S_{cand}, S_{current}, T_i)$;
9 **xSy** ← **outranking(**S_{cand}**,** P_i**)**;
10 **if** $g((S_{cand}, v_i, z) \leq g((P_i, v_i, z)$ **then**
11 $S_{current} \leftarrow S_{cand}$;
12 **if xSy** = **true** *or* **bp** < **U**(**0**, **1**) **then**
13 $P_i \leftarrow S_{cand}$;
14 **end if**
15 **end if**
16 $z \leftarrow obtainIdealPoint(S_{cand})$;
17 **end for**
18 **end for**
19 $FE \leftarrow N * L$;
20 $Ti \leftarrow Ti * \alpha$;
21 **end while**

En resumen, MOSA/D-O (Algoritmo 3.8) es estricto al aceptar una solución candidata en la nueva población, ya que debe satisfacer tanto la relación de Tchebycheff como la de superación (xSy). Mientras que MOSA/D-O-II (Algoritmo 3.9) relaja la condición al tratar de encontrar soluciones que satisfagan la relación de superación xSy, pero si esto no sucede, probablemente se seleccionará al menos una solución que satisfaga el criterio de Tchebycheff.

3.4.4 CO-MOSA/D: Recocido Simulado Multi-objetivo Coevolutivo Basado en Descomposición

CO-MOSA/D es un algoritmo coevolutivo cooperativo basado en MOSA/D. Básicamente, el algoritmo divide el vector de variables de decisión en varios subconjuntos, es decir, el fenotipo único se divide en *N* fenotipos más pequeños llamados especies. Cada especie debe evolucionar de forma independiente por un tiempo determinado. En ese proceso, cada vez que un nuevo individuo es creado, se evalúa mediante la función de Tchebycheff utilizando cooperadores de las demás especies. El algoritmo de CO-MOSA/D es descrito en el algoritmo 3.5 y tiene el siguiente funcionamiento:

El algoritmo CO-MOSA/D recibe como entrada las evaluaciones de la función máxima (*MFE*), número de evaluaciones por especie ($sEvals$), tamaño de la población y subproblemas (*N*), número de especies (*nSpecies*), tamaño de la población de especies (n) y número de variables de decisión ($nVars$). Además, retorna como salida la población externa (exP) que representa el frente de Pareto aproximado. Se generan las configuraciones de cada especie en forma aleatoria (línea 5). El algoritmo funciona de la siguiente manera: en la primera línea se generan N subproblemas. A continuación, se crean aleatoriamente la población externa (línea 2) y las poblaciones de especies (línea 3). En la línea 4 se inicializa la temperatura para cada especie. En la línea 5 se crea el fenotipo de cada especie mediante la asignación aleatoria de genes solución, donde se crean máscaras de fenotipo para cada especie, apuntando solo a unos pocos genes y bloqueando el resto. Las máscaras de fenotipo se almacenan en $sPhenotypes$ para su aplicación en applyMOSAD(), que se encarga de generar nuevos individuos para cada especie.

El bucle principal comienza en la línea 6, y mientras no se alcanza el FME, realiza las siguientes tareas:

1. En la línea 7, se crea el conjunto de índices (ϕ). Este conjunto representa los índices de subproblemas que se optimizarán donde $|\phi| = n$. Por ejemplo, dado *N*=6 y *n*=2, se seleccionan $\phi = \{1, 2\}$ subproblemas de índice para este ciclo.
2. En la línea 8, se selecciona un subconjunto de subproblemas (S_λ) para optimizar. El subconjunto $S_\lambda \subseteq \lambda$ se genera mediante la selección de *n* subproblemas donde $S_\lambda =$

$\{\lambda_{\phi_1}, \ldots, \lambda_{\phi_n}\}$. Siguiendo el último ejemplo, si tenemos $\phi = \{1,2\}$, entonces $S_\lambda = \{\lambda_{\phi_1}, \lambda_{\phi_2}\} = \{\lambda_1, \lambda_2\}$ para optimizar en esta generación.

3. En la línea 9 comienza el bucle para evolucionar cada especie desde $k \leftarrow 1$ hasta $nSpecies$:
    a. En la línea 10, el subconjunto S_λ se optimiza con la $species_k$ actual y en colaboración con sus cooperadores.
4. En la línea 11, se actualiza la población externa exP. Una vez que se optimizó S_λ en el paso anterior, el subconjunto $\{exP_{\phi_1}, \ldots, exP_{\phi_n}\}$ de soluciones se puede actualizar con las $species$. En el algoritmo 3, el proceso de actualización comienza con un bucle desde $i \leftarrow 1$ hasta $nSpecies$. A continuación, comienza un bucle interno desde $j \leftarrow 1$ hasta n. Si en términos de Tchebycheff, $species_{i,j}$ es al menos tan buena como exP_{ϕ_i}, entonces $exP_{\phi_i} \leftarrow species_{i,j}$.

Algorithm 3.5: CO-MOSA/D

Input: MFE, $sEval$, N, n, $nSpecies$, $nVars$
Output: exP

```
1  λ ← generateSubProblems(N)
2  exP ← initExternalPopulation(N)
3  species ← initSpeciesPopulations(nSpecies, n)
4  temps ← initTemperatures(nSpecies)
5  sPhenotypes ← randomAssign(nSpecies, nVars)
6  while FE ≤ FME do
7      ϕ ← nextSubProblems(N, n)
8      S_λ ← asignSubProblems(λ, ϕ)
9      for k ← 1 to nSpecies do
10         species_k ← applyMOSAD(k, sEval, temps_k, sPhenotypes_k, species, S_λ, n)
11     end for
12     exP ← updateExP(species, nSpecies, n, ϕ, S_λ, exP)
13     FE ← FE + (sEval * nSpecies)
14 end while
15 return exP
```

5. El *FE* se actualiza en la línea 12; si se alcanza la condición de detención, el bucle principal se detiene; de lo contrario, el proceso continúa en la línea 7.

Finalmente, la línea 13 devuelve la última generación de exP, finalizando el algoritmo principal.

3.4.4.1 El Algoritmo applyMOSAD

El núcleo del algoritmo CO-MOSA/D es la función applyMOSAD(); se utiliza entre las líneas 9 y 10 del algoritmo 1 para optimizar S_λ. El algoritmo recibe como entrada la especie actual (k), el número de evaluaciones ($sEvals$), la temperatura para k ($temps_k$), las máscaras de fenotipo para k ($sPhenotypes_k$), el conjunto de poblaciones de especies ($species$), el conjunto de subproblemas seleccionados (S_λ) y n. Como salida, el algoritmo devuelve la última generación de P. La función funciona de la siguiente manera: en la primera línea, se establece $sPhenotypes_k$ en los operadores genéticos de DE. A continuación, en la línea 2, se obtiene la población de especies actual (P) a partir del conjunto de especies dado por k. En la línea 3, se seleccionan los cooperadores de k especies y se inicializa la variable $evals$ en la línea 4.

En la línea 5, comienza el bucle principal; mientras la temperatura final (Tf) de no se alcance en $temps_k$ o el número de evaluaciones ($sEvals$) no se haya alcanzado en evals, se realizarán los siguientes pasos:

1. En la línea 6, el segundo ciclo comienza a cubrir n subproblemas, luego:
    a) El individuo actual (ind_{curr}) se obtiene de la posición i-ésima en P (línea 7).
    b) En la línea 8, el segundo bucle comienza el recocido simulado para el subproblema S_{λ_i} por L veces:
        i. En la línea 9, la solución colaborativa se crea combinando las DV de ind_{cand} y las DV de sus cooperadores. Un ejemplo ilustrativo se muestra en la Figura 3.11, donde se generó una solución por cooperación.

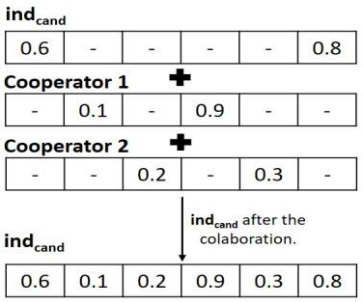


Figura 3.11. Ejemplo de colaboración entre tres especies.

ii. En la línea 11, sin la máscara de fenotipo, los objetivos se calculan a partir de ind_{cand}.

1. Se realiza una conversión de DV continuos a binarios generando la solución ind_{binary} (Figura 3.12).

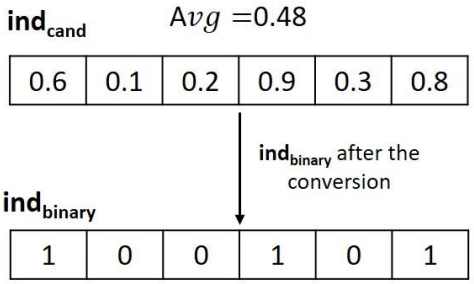


Figura 3.12. Conversión de ind_{cand} a ind_{binary}.

2. Los objetivos se calculan después de que ind_{binary} esté listo.

iii. En la línea 12, se evalúan las restricciones y el resultado se almacena en c.

iv. En la línea 13, ind_{curr} recibe colaboraciones de sus cooperadores.

v. En la línea 14, se evalúan los objetivos de ind_{curr}.

vi. En la línea 15, se calcula la probabilidad de Boltzmann.

vii. En la línea 16, se calcula Tchebycheff ponderado (g_{cand}) para ind_{cand}.

viii. En la línea 17, se calcula Tchebycheff ponderado (g_{P_i}) para P_i.

ix. En la línea 18, se calcula Tchebycheff ponderado (g_{curr}) para ind_{curr}.

x. Ahora es el momento de actualizar la solución P_i, si g_{cand} es al menos tan buena como g_{P_i}, entonces $P_i \leftarrow ind_{cand}$ (líneas 19 y 20).

xi. Para actualizar ind_{curr} (líneas 21 y 22); para esta tarea, $ind_{curr} \leftarrow ind_{cand}$ si 1) g_{cand} es al menos tan bueno como g_{curr}, o 2) $bp < U(0,1)$.

xii. El contador de evaluaciones ($evals$) se actualiza en la línea 23.

xiii. El punto ideal para $species_k$ y exP se calcula en las líneas 24 y 25.

xiv. Al finalizar el bucle, $temps_k$ debe estar enfriando; a continuación, el bucle principal continúa o se detiene según la condición de detención (línea 26).

2. Finalmente, en la línea 27, se devuelve la última generación de P de la especie k, lo que finaliza el procedimiento.

Algorithm 3.6: applyMOSAD

Input: k, $sEval$, $temps_k$, $sPhenotypes_k$, $species$, S_λ, n
Output: P

```
1  initSpeciesOperators(Phenotypes_k)
2  P ← species_k
3  coops ← cooperatorsFor(k, species)
4  evals ← 1
5  while temps_k ≥ Tf and evals ≤ specieEval do
6      for i ← 1 to n do
7          ind_curr ← P_i
8          for j ← 1 to L do
9              ind_cand ← pert(ind_curr, P)
10             ind_cand ← blend(ind_cand, coops)
11             evaluatedObjectives(ind_cand)
12             c ← evalConstraints(ind_cand)
13             ind_curr ← blend(ind_curr, coops)
14             evaluatedObjectives(ind_curr)
15             bp ← boltzmann(ind_cand, ind_curr, temps_k)
16             g_cand ← g(ind_cand, S_λi, z_k)
17             g_Pi ← g(P_i, S_λi, z_k)
18             g_curr ← g(ind_curr, S_λi, z_k)
19             if g_cand ≤ g_Pi and c = 0 then
20                 P_i ← ind_cand
21             end if
22             if (g_cand ≤ g_curr and c = 0) or bp < U(0,1) then
23                 ind_curr ← ind_cand
24             end if
25             evals ← evals + 1
26             z_k ← sidealPoint(ind_cand)
27             Z ← idealPoint(ind_cand)
28         end for
29     end for
30     temps_k ← temps_k * α
31 end while
32 return P
```

3.4.5 Conversión de Variables Continuas a Discretas para PSP

Los problemas estándar DTLZ y WFG utilizan variables de decisión de tipo continuo y la evolución diferencial (DE) tiene un desempeño adecuado. Pero el PSP utiliza variables de decisión binarias, por lo cual aplicar los operadores de DE en su estado natural no es posible. Es necesario entonces una adecuación de los algoritmos descritos anteriormente para

aplicarlos en PSP. En la sección 2.4.2.2 se mencionan dos trabajos que permiten utilizar DE en problemas de variables de decisión binarias. Para la presente investigación se implementó la estrategia de Ali et al. [36] donde se hace una conversión de las variables de decisión continuas a variables binarias. Esta conversión se hace en el ámbito del problema, específicamente cuando se calculan los objetivos. La figura 3.13 muestra la clase `Conversions` que tiene el método `ConToBin` que se ejecuta cuando el problema PSP calcula los objetivos del problema.

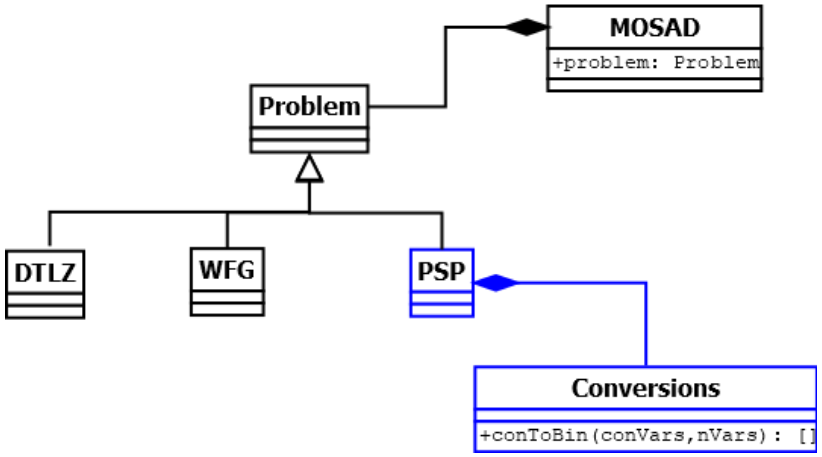


Figura 3.13. Diagrama de clases de los problemas que maneja el Framework.

El algoritmo 3.7 muestra el método `ConToBin` de la clase `Conversions` que convierte las variables de decisión continuas en binarias mediante el cálculo de un umbral representado por el promedio de todas las variables continuas (línea 2). Entonces en la línea 3 se analiza cada variable continua y si esta es igual o mayor al umbral su contraparte binaria adquiere el valor de 1 y en caso contrario el valor de 0.

Algorithm 3.7: ConToBin

Input: decision variables *conVars*, variables number *nVars*
Output: decision variables *binVars*

```
1  threshold ← 0
2  threshold ← mean(conVars)
3  for i ← 1 to nVars do
4  |   if conVars_i ≥ threshold then
5  |   |   binVars_i ← 1
6  |   else
7  |   |   binVars_i ← 0
8  |   end if
9  end for
10 return binVars
```

3.4.6 MOSA/D-PSP - Recocido Simulado Multi-objetivo Basado en Descomposición para PSP

MOSA/D-PSP (Algoritmo 3.13) se deriva de MOSA/D principalmente. El uso de la conversión de las variables de decisión recae en la implementación del problema en la clase PSP.

Algorithm 3.8: MOSA/D-PSP

Input: $MOP, Ti, \alpha, L, Tf, N, MFE$
Output: Population P

1 Initialize $N, P, v, S_{current}, S_{cand}$
2 **while** $T \geq Tf$ & $FE \leq FME$ **do**
3 **for** $i \leftarrow 1$ *to* N **do**
4 $S_{current} \leftarrow P_i$
5 **for** $j \leftarrow 1$ *to* L **do**
6 $S_{cand} \leftarrow perturbation(S_{current})$
7 $bp \leftarrow boltzmannProbability(S_{cand}, S_{current}, T)$
8 **vConstraints** $\leftarrow$ **obtainViolatedConstraints**(S_{cand})
9 **if** $g((S_{cand}, v_i, z) \leq g((P_i, v_i, z)$ ***and vConstraints=0*** **then**
10 $P_i \leftarrow S_{cand}$
11 **end if**
12 **if** $g((S_{cand}, v_i, z) \leq g((S_{current}, v_i, z)$ ***and vConstraints=0*** *or* $bp < U(0,1)$ **then**
13 $S_{current} \leftarrow S_{cand}$
14 **end if**
15 $z \leftarrow obtainIdealPoint(S_{cand})$
16 **end for**
17 **end for**
18 $FE \leftarrow N * L;$
19 $T \leftarrow T * \alpha;$
20 **end while**

Las actualizaciones hechas en el algoritmo principal de MOSA/D son mostradas en el algoritmo 3.13 en negritas. Estas actualizaciones son en el sentido de que PSP es un problema que utiliza restricciones. Por lo cual, cada vez que se genera una nueva solución, se calcula el número de restricciones no respetadas (línea 8) y este indicador se implementó como una condición en el mecanismo de búsqueda en las líneas 9 y 12.

3.4.7 MOSA/D-O-PSP y MOSA/D-O-II-PSP: Recocido Simulado Multi-objetivo Basado en Descomposición y Outranking para PSP

MOSA/D-O-PSP está enfocado en el manejo de preferencias al integrar relaciones de superación (Outranking) para el problema PSP. Al igual que al algoritmo 3.13 el algoritmo

MOSA/D-O (Algoritmo 3.9) tiene actualizaciones sobre las restricciones en las líneas 10, 11 y 14.

Algorithm 3.9: MOSA/D-O-PSP

Input: $MOP, Ti, \alpha, L, Tf, N, MFE$
Output: Population P

```
1  Initialize N, P, v, S_current, S_cand
2  while T ≥ Tf & FE ≤ FME do
3      for i ← 1 to N do
4          S_current ← P_i
5          for j ← 1 to L do
6              S_cand ← perturbation(S_current)
7              bp ← boltzmannProbability(S_cand, S_current, T)
8              xSy ← outranking(S_cand, P_i)
9              xSz ← outranking(S_cand, S_current)
10             vConstraints ← obtainViolatedConstraints(S_cand)
11             if g((S_cand, v_i, z) ≤ g((P_i, v_i, z) and xSy = true and vConstraints = 0 then
12                 P_i ← S_cand
13             end if
14             if g((S_cand, v_i, z) ≤ g((S_current, v_i, z) and xSz = true and vConstraints=0 or
                   bp < U(0,1) then
15                 S_current ← S_cand
16             end if
17             z ← obtainIdealPoint(S_cand)
18         end for
19     end for
20     FE ← N * L
21     T ← T * α
22 end while
```

3.4.8 MOSA/D-O-II-PSP: Recocido Simulado Multi-objetivo Basado en Descomposición y Outranking II para PSP

MOSA/D-O-II-PSP está enfocado en el manejo de preferencias al integrar relaciones de superación (Outranking) para el problema PSP. MOSA/D-O-II (Algoritmo 3.10) agrega cambios con respecto MOSA/D-O-II en las líneas 10 y 11 para verificar que las restricciones del PSP sean respetadas.

Algorithm 3.10: MOSA/D-O-II-PSP

Input: $MOP, Ti, \alpha, L, Tf, N, MFE$
Output: Population P

1 Initialize $N, P, v, S_{current}, S_{cand}$;
2 **while** $T \geq Tf$ & $FE \leq FME$ **do**
3 **for** $i \leftarrow 1$ *to* N **do**
4 $currentP \leftarrow P$
5 $S_{current} \leftarrow currentP_i$
6 **for** $j \leftarrow 1$ *to* L **do**
7 $S_{cand} \leftarrow perturbation(S_{current})$
8 $bp \leftarrow boltzmannProbability(S_{cand}, S_{current}, T)$
9 $xSy \leftarrow outranking(S_{cand}, P_i)$
10 **vConstraints** $\leftarrow$ **obtainViolatedConstraints**($\mathbf{S_{cand}}$)
11 **if** $g((S_{cand}, v_i, z) \leq g((P_i, v_i, z)$ *and* **vConstraints** $= 0$ **then**
12 $S_{current} \leftarrow S_{cand}$
13 **if** $xSy = true or bp < U(0,1)$ **then**
14 $P_i \leftarrow S_{cand}$
15 **end if**
16 **end if**
17 $z \leftarrow obtainIdealPoint(S_{cand})$;
18 **end for**
19 **end for**
20 $FE \leftarrow N * L$;
21 $T \leftarrow T * \alpha$;
22 **end while**

3.4.9 CO-MOSA/D-PSP: Recocido Simulado Multi-objetivo Coevolutivo Basado en Descomposición para PSP.

CO-MOSA/D-PSP es una derivación del algoritmo coevolutivo CO-MOSA/D enfocado en problemas de gran escala para el problema de PSP. En este algoritmo derivado CO-MOSA/D los cambios principales se encuentran en el algoritmo secundario applyMOSAD() para PSP (Algoritmo 3.11) en las líneas 10, 11 y 14 donde se verifican que se cumpla con las restricciones del problema PSP.

Algorithm 3.11: applyMOSAD

Input: k, $sEval$, $temps_k$, $sPhenotypes_k$, $species$, S_λ, n
Output: P

```
initSpeciesOperators(Phenotypes_k)
P ← species_k
coops ← cooperatorsFor(k, species)
evals ← 1
while temps_k ≥ Tf and evals ≤ specieEval do
    for i ← 1 to n do
        ind_curr ← P_i
        for j ← 1 to L do
            ind_cand ← pert(ind_curr, P)
            ind_cand ← blend(ind_cand, coops)
            evaluatedObjectives(ind_cand)
            c ← evalConstraints(ind_cand)
            ind_curr ← blend(ind_curr, coops)
            evaluatedObjectives(ind_curr)
            bp ← boltzmann(ind_cand, ind_curr, temps_k)
            g_cand ← g(ind_cand, S_λi, z_k)
            g_Pi ← g(P_i, S_λi, z_k)
            g_curr ← g(ind_curr, S_λi, z_k)
            if g_cand ≤ g_Pi and c = 0 then
                P_i ← ind_cand
            end if
            if (g_cand ≤ g_curr and c = 0) or bp < U(0,1) then
                ind_curr ← ind_cand
            end if
            evals ← evals + 1
            z_k ← sidealPoint(ind_cand)
            Z ← idealPoint(ind_cand)
        end for
    end for
    temps_k ← temps_k * α
end while
return P
```

3.5 Hiperheurísticas Desarrolladas

En esta sección se describen dos hiperheurísticas diseñadas para tratar con problemas de muchos objetivos, gran escala, preferencias e incertidumbre MaOLS-HH-UP y MaOLS-HH-UP-PSP.

3.5.1 MaOLS-HH-UP: Many Objective Large-scale Hyper-heuristic with Uncertainty and Preferences

En la sección 3.1 se describió en forma general el Framework de Software para Hiperheurísticas que tiene como fin que los investigadores puedan crear algoritmos para los trabajos que son de su pertenencia con enfoque hiperheurístico. La ventaja de manejar un Framework es que este cuenta con una serie de estructuras y algoritmos que son reutilizables y esto agiliza la construcción de nuevos algoritmos. MaOLS-HH-UP tendrá como fin poder resolver problemas que incluyan al menos una de las siguientes características: preferencias, incertidumbre, muchos objetivos y gran escala. La estructura organizacional del algoritmo se muestra en la figura 3.14 donde podemos observar los mecanismos de alto nivel que usa MaOLS-HH-UP como lo es la estrategia de selección y también las heurísticas de bajo nivel que puede utilizar como lo son MOSA/D, MOSA/D-O, MOSA/D-O-II y CO-MOSA/D.

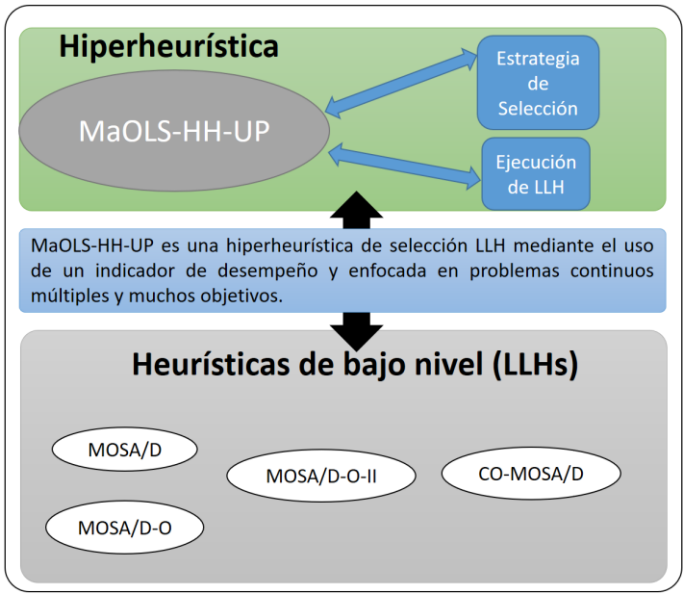


Figura 3.14. Estructura organizacional de MaOLS-HH-UP.

MaOLS-HH-UP se puede clasificar como una Hiperheurística de Aprendizaje en línea, con manejo de grupo de LLHs basada en escenarios, de tipo multi-objetivo que controla metaheurísticas, sin movimiento de aceptación y con un ajuste de parámetros estático. El diagrama de flujo del marco general de hiperheurísticas en la que se basa MaOLS-HH-UP se muestra en la figura 3.14.

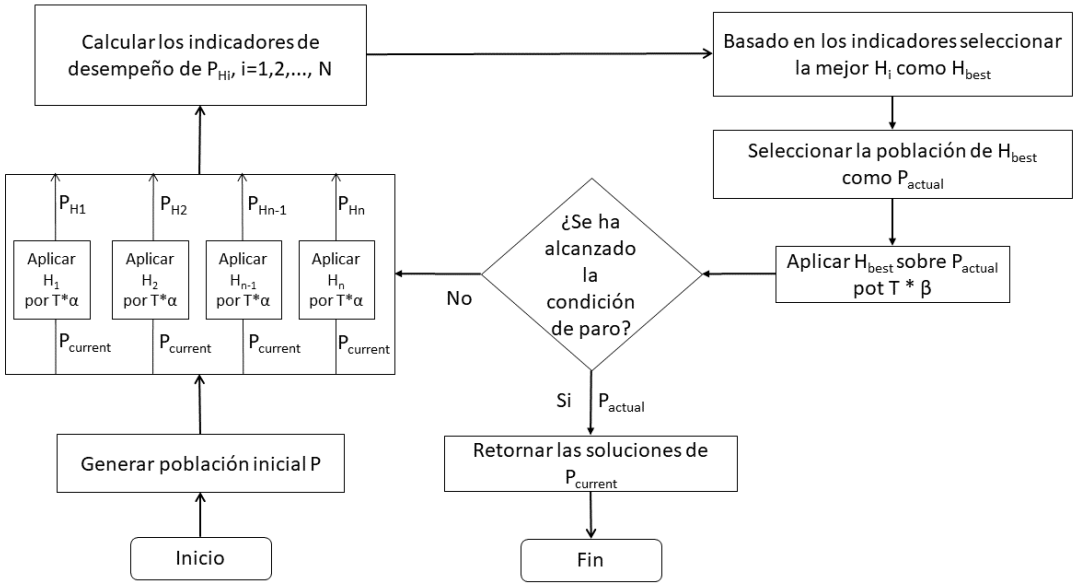


Figura 3.15. Diagrama de flujo del marco general de hiperheurísticas.

Podemos resumir que el algoritmo MaOLS-HH-UP en esta primera versión básica utiliza un conjunto de LLHs que son evaluadas durante un tiempo α. Una vez terminada la evaluación, se procede a determinar cuál fue el mejor algoritmo basado en un indicador, en este caso el promedio de dominancia. Entonces se encoge al algoritmo que tenga el mayor promedio de dominancia con respecto a la población de referencia (*P*). Una vez seleccionado se aplica el mejor algoritmo por un tiempo β a la población. Cuando el tiempo β se termina se evalúa si ha alcanzado la condición de paro y si no vuele a comenzar.

El pseudocódigo de MaOLS-HH-UP es descrito en el algoritmo 3.13, el algoritmo recibe como entrada el conjunto de LLHs y los tiempos de evaluación α (para la aplicación de cada LLH) y β (para la aplicación de H_{best}) donde $\alpha \leq \beta$. La salida esta representa por la última generación de *P*. El algoritmo comienza generando una población inicial de forma aleatoria relacionada con el problema de optimización (línea 1), también se inicializa cada LLH del conjunto *H* (línea 2). En la línea 3, se obtiene el número de LLHs disponibles. Inicia el ciclo principal (línea 4), en el ciclo principal ocurren las siguientes tareas:

a. Se inicia un ciclo que va de i=1 hasta $size$ para evaluar cada LLH (línea 5-8):
    i. Se aplica cada LLH a *P* durante un número α de ciclos (línea 6)
    ii. Se actualiza el número de evaluaciones *FE* (línea 7),

b. Se aplica un proceso de selección de una heurística (H_{best}) de entre el conjunto de heurísticas (LLHS) (línea 9).
c. Se aplica H_{best} durante un número β de evaluaciones (Línea 10)
d. Se actualiza $FE = FE + \beta$ y termina el ciclo (línea 11)
e. Finalmente, al terminar el ciclo principal (línea 12) se retorna P con la última generación de soluciones y termina el algoritmo (línea 13).

Algorithm 3.13: MaOLS-HH-UP

Input: Low Level Heuristics Set $LLHSet$, evaluations MFE, size of population N, evaluations α, evaluations β
Output: Last generation of P

```
1  P ← initializePopulation(N)
2  LLHSet ← LoadLowLevelHeuristics()
3  size ← |LLHSet|
4  while FE ≤ FME do
5      for i ← 1 to size do
6          P ← applyLLH(LLHSet_i, P, α)
7          FE ← FE + α
8      end for
9      LLH_best ← SelectionOperator(LLHSet, P)
10     P ← applyLLH(LLH_best, P, β)
11     FE ← FE + β
12 end while
13 return P
```

3.5.2 Hiperheurísticas Derivadas del Modelo MaOLS-HH-UP

- Se han desarrollado a lo largo de la investigación más de 30 hiperheurísticas de entre ellas 28 hiperheurísticas diferentes en un período de un mes utilizando el framework propuesto para una experimentación robusta.
- HH-WFG (9 variantes), HH-DTLZ (9 variantes) y HH-PSP (10 variantes) son las hiperheurísticas generadas para un total 28 hiperheurísticas
- Estas hiperheurísticas están diseñadas para resolver problemas de muchos objetivos, específicamente los conjuntos de prueba DTLZ, WFG, y PSP, utilizando una combinación de heurísticas de bajo nivel (LLHs).
- Los LLHs que componen estas hiperheurísticas incluyen variantes como MOSA/D, CO-MOSA/D, MOSA/D-O, MOSA/D-O-II, MOSA/D-PSP, CO-MOSA/D-PSP, MOSA/D-O-PSP, y MOSA/D-O-II-PSP.

- Los indicadores de desempeño utilizados para la selección de LLHs incluyen promedio, mediana y mínimo de distancia Euclídea y distancia Tchebycheff, además de métricas como IGD (Inverted Generational Distance), Hipervolumen (HV), y Parallel Coordinate Area (PCArea).

Capítulo 4

4 Análisis y Resultados

En el capítulo anterior se presentó el diseño general del framework de software de hiperheurísticas, sus algoritmos hiperheurísticos y sus heurísticas de bajo nivel (LLHs). El framework permite la creación de nuevas hiperheurísticas, LLHs, y sus componentes reutilizables. Para probar el funcionamiento del framework en las problemáticas contempladas en el capítulo 1 (muchos objetivos, gran escala, preferencias e incertidumbre). Se plantea el siguiente diseño experimenta:

4.1 Contexto de la Investigación

La presente investigación parte del objetivo de crear un framework de software para la creación de hiperheurísticas que pueda resolver problemas de optimización de múltiples y muchos objetivos, gran escala, preferencias e incertidumbre. Otra característica es que está enfocado en problemas estándar de variables de tipo continuo y problemas de variables de decisión de tipo binario como PSP. Partiendo de ese objetivo en el transcurso de esta investigación, se ha seguido un camino evolutivo, desde la exploración de algoritmos estándar de la literatura (MOEA/D, FDEA-I. FDEA-II) hasta el desarrollo de heurísticas e hiperheurísticas únicas y adaptables. Este viaje ha sido fundamental para comprender tanto las limitaciones presentes en los enfoques de solución ya establecidos como el potencial aún disponible de la innovación en el diseño de algoritmos de optimización.

Inicialmente, el enfoque se centró en el estudio exhaustivo de algoritmos estándar utilizados en problemas de optimización continua, específicamente Recocido Simulado, MOEA/D y evolución diferencial. Este análisis proporcionó una comprensión profunda de las fortalezas y debilidades de estos enfoques, sirviendo como punto de partida para la exploración de estrategias más avanzadas. Basándose en el conocimiento adquirido durante la fase inicial, se procedió al desarrollo de una heurística de bajo nivel (LLH) propia, denominada MOSA/D (Multi-Objective Simulated Annealing based on Decomposition). Esta LLH fue diseñada para abordar problemas de muchos objetivos en problemas de optimización continua, destacando por su capacidad de convergencia hacia el frente de Pareto y mantener la diversidad de la población. Los resultados presentados (ver anexo A.1) fueron publicados en la revista Mathematical and Computational Applications en 2023 [69].

La creación de MOSA/D no solo representó una aportación novedosa en el proceso de la de esta investigación, sino que también sirvió como punto de partida para la generación de nuevas heurísticas y la formulación de una estrategia hiperheurística. Los algoritmos derivados MOSA/D-O y MOSA/D-O-II basados en preferencias demostraron competitividad en sus resultados. Estos resultados también fueron publicados en la revista SoftwareX en 2024 [71]. Con un conjunto diverso de LLHS disponibles, se procedió a integrarlas en un ambiente de desarrollo con los mecanismos necesarios para crear hiperheurísticas y este ambiente es el framework de hiperheurísticas. Posteriormente, se realizaron experimentos exhaustivos para validar la eficacia de la hiperheurística MaOLS-HH-UP (ver Anexo A.23). Este recorrido, desde el estudio de algoritmos estándar hasta la creación y validación de una hiperheurística como MaOLS-HH-UP, ha proporcionado una base sólida para la investigación actual. Todos estos algoritmos y mecanismos de desarrollo forman el framework de hiperheurísticas que como objetivo persigue esta investigación.

Los resultados del anexo A.3 demuestran que se pueden generar hiperheurísticas competitivas mediante el framework propuesto en problemas de optimización continuos para muchos objetivos. Pero, ¿Es posible aplicar este framework para problemas de optimización binarios o continuos de muchos objetivos, gran escala, preferencias e imprecisión? Con base en esta pregunta, retomamos lo planteado en la hipótesis de la investigación del capítulo 1:

"La integración conjunta de relaciones de superación, intervalos, y descomposición dentro de un framework de optimización permite el desarrollo rápido y versátil de hiperheurísticas, con manejo de preferencias en el proceso de búsqueda, que aproximen eficientemente el frente óptimo de Pareto del Problema de Selección de Proyectos y afines en presencia de imprecisión e incertidumbre, gran escala y muchos objetivos". Y basado en esta se plantea el siguiente diseño experimental.

4.2 Diseño Experimental Propuesto

Para responder la hipótesis de investigación, el diseño experimental contempla resolver 7 problemas estándar (DTLZ) con 5 objetivos y 7 problemas estándar (DTLZ) de 10 objetivos. Estos problemas servirán para medir la capacidad de los algoritmos para enfrentar problemas de muchos objetivos, además de preferencias de un decisor con incertidumbre. La configuración se puede observar en la tabla 4.1.

Tabla 4.1. Diseño experimental para los problemas DTLZ

Variables	**Valores**
Problemas	DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5, DTLZ6, DTLZ7
No. de objetivos	5 y 10 objetivos
No. de variables de decisión	n = m+k, k = 5 para DTLZ1, k=10 para DTLZ2-DTLZ6 y k=20 para DTLZ7.
Modelo de muchos objetivos	Descomposición
Modelo de gran escala	Coevolución cooperativa
Modelo de imprecisión	Intervalos.
Modelo de preferencias	Outranking
Algoritmos Hiperheurísticos	9 variantes de HH_DTLZ (Ver Tabla B.3)
LLHs	MOSA/D, CO-MOSA/D, MOSA/D-O, MOSA/D-O-II

También se contempla resolver 9 problemas estándar (WFG) con 5 objetivos y 9 problemas estándar (WFG) de 10 objetivos. Estos problemas servirán para medir la capacidad de los algoritmos para enfrentar problemas de muchos objetivos, además de preferencias de un decisor con incertidumbre. La configuración se puede observar en la tabla 4.2.

Tabla 4.2. Diseño experimental para los problemas WFG

Variables	**Valores**
Problemas	WFG1, WFG2, WFG3, WFG4, WFG5, WFG6, WFG7, WFG8, WFG9
No. de objetivos	5 y 10 objetivos
No. de variables de decisión	45 para 5 objetivos y 105 para 10 objetivos
Modelo de muchos objetivos	Descomposición
Modelo de gran escala	Coevolución cooperativa
Modelo de imprecisión	Intervalos.
Modelo de preferencias	Outranking
Algoritmos Hiperheurísticos	9 variantes de HH_WFG (Ver Tabla B.4)
LLHs	MOSA/D, CO-MOSA/D, MOSA/D-O, MOSA/D-O-II

Finalmente, se contempló resolver 6 problemas PSP con 3 objetivos y 6 problemas PSP de 5 objetivos. Estos problemas servirán para medir la capacidad de los algoritmos para enfrentar problemas de muchos objetivos, gran escala, además de preferencias de un decisor con incertidumbre. La configuración se puede observar en la tabla 4.3.

Tabla 4.3. Diseño experimental para los problemas PSP

Variables	**Valores**
Problemas	PSP1, PSP2, PSP2, PSP3, PSP4, PSP5
No. de objetivos	3 y 5 objetivos
No. de variables de decisión	500 y 1000
Modelo de muchos objetivos	Descomposición
Modelo de gran escala	Coevolución cooperativa
Modelo de imprecisión	Intervalos.
Modelo de preferencias	Outranking
Algoritmos hiperheurísticos	9 variantes HH_PSP (Ver Tabla B.5)
LLHs	MOSA/D-PSP, CO-MOSA/D-PSP, MOSA/D-O-PSP, MOSA/D-O-II-PSP.

4.2.1 Configuración de los Algoritmos Hiperheurísticos (HHs)

En la tabla 4.4 puede observarse las variables configurables para los algoritmos hiperheurísticos en forma general, como lo son el número de evaluaciones, las LLHs utilizadas por cada algoritmo, el número de evaluaciones α y β, si se necesita conversión continua a binaria, etc. Esto aplicará para cada una de las variantes de los algoritmos generales que se encuentran en las Tablas B.3, B.4 y B.5 (Anexos).

Tabla 4.4. Variables configurables de los hiperheurísticos.

Variables	**HH_DTLZ**	**HH_DTLZ**	**HH_ PSP**
Población	100	100	100
No. de evaluaciones	100000	100000	100000
Heurísticas de bajo nivel (LLHs)	MOSA/D, CO-MOSA/D, MOSA/D-O Y MOSA/D-O-II	MOSA/D, CO-MOSA/D, MOSA/D-O Y MOSA/D-O-II	MOSA/D-PSP, CO-MOSA/D-PSP, MOSA/D-O-PSP Y MOSA/D-O-II-PSP
No. de evaluaciones α	1000	1000	1000
No. de evaluaciones β	6000	6000	6000
Conversión continua a Binaria	No	No	Si
Variantes	9 variantes Ver Tabla B.3	9 variantes Ver Tabla B.4	10 variantes Ver Tabla B.5

Con respecto a las variantes de cada HH general, cada variante implementa un mecanismo distinto para seleccionar la siguiente LLH que se usará en el proceso de búsqueda. Por lo que para DTLZ existen 9 algoritmos hiperheurísticos (Tabla B.3), para WFG 9 algoritmos

hiperheurísticos (Tabla B.4) y para PSP 10 algoritmos hiperheurísticos (Tabla B.5). Una descripción de las métricas utilizadas puede ser observada en la Tabla 4.5.

Tabla 4.5. Métricas utilizadas en el proceso de selección de LLHs en los algoritmos HH-DTLZ, HH-WFG y HH-PSP.

Métrica	Descripción
EUCL-AVG()	Promedio de distancia euclidiana. Esta métrica calcula la distancia media de cada solución en el frente aproximado a su punto más cercano en el frente de Pareto ideal (o referencia).
EUCL-MIN()	Distancia euclidiana mínima. Identifica la menor distancia entre una solución en el frente aproximado y el frente ideal.
EUCL-MED()	Mediana de la distancia euclideana. Ordena las distancias euclidianas de todas las soluciones al frente ideal y toma la mediana, ofreciendo una perspectiva central de qué tan cerca están las soluciones en general del FP.
TCHE-AVG()	Promedio de distancia Tchebycheff. Similar al promedio de distancia euclidiana, esta métrica calcula la distancia media de cada solución en el frente aproximado al frente ideal usando la distancia Tchebycheff.
TCHE-MIN()	Distancia Tchebycheff mínima. Registra la distancia Tchebycheff más corta entre una solución del frente aproximado y el frente ideal.
TCHE-MED()	Mediana de la distancia Tchebycheff. Ordena las distancias Tchebycheff de todas las soluciones al frente ideal y toma la mediana, brindando una medida central para la precisión respecto a la peor desviación de cada objetivo en general.
PR()	Región de preferencia. Define una región específica en el espacio objetivo donde se concentran las preferencias del decisor. Las soluciones dentro de esta región son las más deseables, y el algoritmo se evalúa en función de su capacidad para encontrar soluciones dentro de esta área.
HV()	Hipervolumen. Mide el volumen del espacio objetivo dominado por el conjunto de soluciones no dominadas en relación con un punto de referencia.
IGD()	Distancia Generacional Invertida. Calcula la distancia promedio desde cada punto del frente de referencia al frente aproximado generado por el algoritmo.
PCArea()	Parallel Cordinate Área. Calcula el área entre las líneas que representan las soluciones en un diagrama de coordenadas paralelas.

4.2.2 Configuración de las Heurísticas de Bajo Nivel (LLHs)

Las LLHs son los algoritmos que maneja una hiperheurística, son ellos los que resuelven los MOPs o MaOPS dentro de una estrategia hiperheurística. Para este diseño experimental los algoritmos utilizados son: MOSA/D, MOSA/D-PSP, CO-MOSA/D y CO-MOSA/D-PSP, MOSA/D-O, MOSA/D-O-PSP, MOSA/D-O-II y MOSA/D-O-II-PSP. Las configuraciones de las variables de cada LLH son mostradas en la Tabla 4.6 para los algoritmos que no contemplan preferencias.

Tabla 4.6. Variables configurables de MOSA/D, MOSA/D-PSP, CO-MOSA/D y CO-MOSA/D-PSP.

Variables	**MOSA/D**	**MOSA/D-PSP**	**CO-MOSA/D**	**CO-MOSA/D-PSP**
Población	100	100	100	100
No. de evaluaciones	100000	100000	100000	100000
Temperatura inicial (Ti)	1	1	1	1
Temperatura final (Tf)	0.0000001	0.0000001	0.0000001	0.0000001
Tasa de enfriamiento (α)	0.98	0.98	0.98	0.98
Factor de escala	0.5	0.5	0.5	0.5
Tasa de cruza	0.2	0.2	0.2	0.2
Largo de la cadena de Markov(L)	2	2	2	2
Número de especies	-	-	2	2
Población por especie	-	-	100	100
Evaluaciones por especie	-	-	1000	1000
Tipo de variables de decisión	Continuas	Binarias	Continuas	Binarias

Las configuraciones de las variables de cada LLH son mostradas en la Tabla 4.7 para los algoritmos que contemplan el manejo de preferencias de un decisor con incertidumbre.

Tabla 4.7. Variables configurables de MOSA/D-O, MOSA/D-O-PSP, MOSA/D-O-II y MOSA/D-O-II-PSP.

Parámetro	**MOSA/D-O**	**MOSA/D-O-PSP**	**MOSA/D-O-II**	**MOSA/D-O-II-PSP**
Población	100	100	100	100
No. de evaluaciones	100000	100000	100000	100000
Temperatura inicial (Ti)	1	1	1	1
Temperatura final (Tf)	0.0000001	0.0000001	0.0000001	0.0000001
Tasa de enfriamiento (α)	0.98	0.98	0.98	0.98
Factor de escala	0.5	0.5	0.5	0.5
Tasa de cruza	0.2	0.2	0.2	0.2
Largo de la cadena de Markov(L)	2	2	2	2
Pesos (w)	[w1,w1],[w2,w2],...	[w1,w1],[w2,w2],...	[w1,w1],[w2,w2],...	[w1,w1],[w2,w2],...
Umbrales de veto (v)	[v1,v1],[v2,v2],...	[v1,v1],[v2,v2],...	[v1,v1],[v2,v2],...	[v1,v1],[v2,v2],...
Umbral de mayoría (λ)	[0.6, 0.6]	[0.6, 0.6]	[0.6, 0.6]	[0.6, 0.6]
Tipo de variables de decisión	Continuas	Binarias	Continuas	Binarias

4.2.3 Configuraciones de las Instancias de Prueba

Las instancias de prueba fueron pensadas para probar las hiperheurísticas mencionadas anteriormente (Tabla 4.4) en las características de muchos objetivos, gran escala, preferencias e imprecisión. Se tienen para el benchmark DTLZ alrededor de 14 instancias de prueba divididas en 7 para 5 objetivos y 7 para 10 objetivos (Tabla 4.8).

Tabla 4.8. Configuraciones de las instancias DTLZ (Instancias de muchos objetivos).

Variables	**Configuración 1**	**Configuración 2**
No. de objetivos (m)	5	10
No. de problemas	7	7
No. de variables de decisión por problema (n)	n = m+k, k = 5 para DTLZ1, k=10 para DTLZ2-DTLZ6 y k=20 para DTLZ7.	
No. de decisores	1	1
Total de instancias	14	

En el caso de WFG se cuenta con 18 instancias de prueba (Tabla 4.9), divididas en 9 instancias para 5 objetivos y 9 instancias para 10 objetivos.

Tabla 4.9. Configuraciones de las instancias WFG (Instancias de muchos objetivos).

Variables	**Configuración 1**	**Configuración 2**
Número de objetivos (m)	5	10
Número de problemas	9	9
Número de variables de decisión por problema (n)	n=47 para 5 objetivos, n=105 para 10 objetivos	
Número de decisores	1	1
Total de instancias	18	

Las instancias de prueba para PSP son 12 divididas en 3 instancias de 500 variables de decisión con 3 objetivos, 3 instancias de 500 variables de decisión con 5 objetivos, 3 instancias de 1000 variables de decisión con 3 objetivos y 3 instancias de 1000 variables de decisión con 5 objetivos (Tabla 4.10). Estas instancias se enfocan en características como gran escala, muchos objetivos, y preferencias de un decisor con incertidumbre.

Tabla 4.10. Configuraciones de las instancias PSP (Instancias de gran escala).

Variables	**Configuración 1**	**Configuración 2**	**Configuración 3**	**Configuración 4**
Número de objetivos	3	5	3	5
Número de problemas	3	3	3	3
Número de variables de decisión por problema	500	500	1000	1000
Número de decisores	1	1	1	1
Total de instancias	12			

4.3 Análisis de Resultados

En esta sección se analizan los resultados en dos enfoques: primero la capacidad de los algoritmos de aproximarse el frente de Pareto (PF) y segundo la capacidad de los algoritmos de aproximarse a la región de interés de un decisor.

- Para el primer enfoque se utilizará la métrica de hipervolumen (HV) la cual en un solo valor indica la proximidad al frente de Pareto y la dispersión sobre este (Tabla 4.11). Para cada instancia se calcula el promedio y la desviación estándar de los resultados de las 30 ejecuciones que se llevaron a cabo. Entre mayor sea el valor de HV, mejores son los resultados obtenidos para cada algoritmo. Para observar si hay diferencia significativa entre los algoritmos, se utilizó la prueba no paramétrica de Wilcoxon de 2 colas.

Tabla 4.11. Métricas utilizadas para observar si los algoritmos se aproximan al PF.

Métrica	Descripción
Hipervolumen (HV)	Hipervolumen. Mide el volumen del espacio objetivo dominado por el conjunto de soluciones no dominadas en relación con un punto de referencia.

- Para el segundo enfoque se utilizarán métricas descritas en la Tabla 4.12 y el análisis Borda mediante un ranking indicara cual es la mejor estrategia por métrica. Si alguna estrategia empata con otra esto indicara que no hay diferencia significativa entre ellas.

Tabla 4.12. Métricas utilizadas por el análisis Borda para observar si los algoritmos se aproximan a la ROI representada por el mejor compromiso (BC).

Métrica	Descripción
avg_ eucl	Promedio de distancia euclidiana. Esta métrica calcula la distancia media de cada solución en el frente aproximado a su punto más cercano en el frente de Pareto ideal (o referencia).
min_ eucl	Distancia euclidiana mínima. Identifica la menor distancia entre una solución en el frente aproximado y el frente ideal.
med_ eucl	Mediana de la distancia euclideana. Ordena las distancias euclidianas de todas las soluciones al frente ideal y toma la mediana, ofreciendo una perspectiva central de qué tan cerca están las soluciones en general del FP.
avg_ tche	Promedio de distancia Tchebycheff. Similar al promedio de distancia euclidiana, esta métrica calcula la distancia media de cada solución en el frente aproximado al frente ideal usando la distancia Tchebycheff.
min_ tche	Distancia Tchebycheff mínima. Registra la distancia Tchebycheff más corta entre una solución del frente aproximado y el frente ideal.
med_ tche	Mediana de la distancia Tchebycheff. Ordena las distancias Tchebycheff de todas las soluciones al frente ideal y toma la mediana, brindando una medida central para la precisión respecto a la peor desviación de cada objetivo en general.
xSz_out1	La métrica xSz_out1 mide la cantidad de soluciones que son capaces de superar al mejor compromiso (BC) en una población de soluciones.
zPx_out2	La métrica zPx_out2 mide la capacidad del mejor compromiso de superar a las soluciones pertenecientes a una población de soluciones.

4.3.1 Análisis de la Capacidad de los Algoritmos de Alcanzar el Frente de Pareto.

4.3.1.1 DTLZ con 5 y 10 Objetivos

En la tabla 4.13 se observan los resultados de cada algoritmo representados por la media y desviación estándar de HV, los resultados de mejor desempeño están resaltados en fondo gris. Se realizó una prueba de hipótesis para observar si hay diferencias significativas mediante la prueba de pares de Wilcoxon con dos colas con grado de significancia de 5%. Para cada algoritmo de mejor desempeño en términos de HV se aplicó la prueba contra todos sus pares. Los algoritmos que tienen diferencia significativa con el mejor resultado están marcados con # y los que no tienen diferencia significativa con *.

Tabla 4.13. Resultados términos de HV de 9 hiperheurísticas para las 14 instancias de prueba DTLZ.

PROBLEMA	M	HH_DTLZ-EUCL-AVG	HH_DTLZ-EUCL-MED	HH_DTLZ-EUCL-MIN	HH_DTLZ-TCHE-AVG	HH_DTLZ-TCHE-MED	HH_DTLZ-TCHE-MIN	HH_DTLZ-PR	HH_DTLZ-IGD	HH_DTLZ-PCArea
DTLZ1	5	1.3554e-01# (2.9062e-01)	6.3546e-02# (1.7644e-01)	4.0257e-02# (1.7559e-01)	4.0821e-02# (1.7735e-01)	2.6637e-02# (9.3182e-02)	2.2661e-02# (8.3392e-02)	1.8834e-02# (6.2298e-02)	0.0000e+00# (0.0000e+00)	9.8966e-01 (1.4180e-03)
	10	9.9533e-02# (2.4553e-01)	3.6622e-02# (1.4280e-01)	7.1992e-02# (2.1474e-01)	1.2784e-01# (2.9795e-01)	1.2590e-01# (2.8554e-01)	1.4644e-01# (2.8031e-01)	0.0000e+00# (0.0000e+00)	1.1265e-01# (2.3061e-01)	9.4004e-01 (8.6538e-02)
DTLZ2	5	3.0448e+01# (2.5128e-01)	3.0488e+01# (4.5664e-01)	3.0966e+01# (1.9573e-01)	3.0709e+01# (2.9465e-01)	3.0579e+01# (4.7242e-01)	3.0892e+01# (2.1829e-01)	3.0457e+01# (2.4691e-01)	3.1004e+01 (1.6712e-01)	3.0940e+01* (1.5462e-01)
	10	9.2477e+02# (3.1695e+01)	9.3226e+02# (4.0916e+01)	9.2198e+02# (2.2543e+01)	9.2873e+02# (3.1510e+01)	9.3461e+02# (3.4195e+01)	9.2515e+02# (2.5866e+01)	9.1363e+02# (2.0928e+01)	9.5848e+02 (1.8170e+01)	9.5530e+02* (2.0791e+01)
DTLZ3	5	0.0000e+00# (0.0000e+00)	0.0000e+00# (0.0000e+00)	0.0000e+00# (0.0000e+00)	0.0000e+00# (0.0000e+00)	0.0000e+00# (0.0000e+00)	0.0000e+00# (0.0000e+00)	0.0000e+00# (0.0000e+00)	0.0000e+00# (0.0000e+00)	5.3410e+03 (4.9522e+03)
	10	0.0000e+00* (0.0000e+00)	0.0000e+00* (0.0000e+00)	0.0000e+00* (0.0000e+00)	0.0000e+00* (0.0000e+00)	0.0000e+00* (0.0000e+00)	0.0000e+00* (0.0000e+00)	0.0000e+00* (0.0000e+00)	0.0000e+00* (0.0000e+00)	7.5170e+05 (3.8191e+06)
DTLZ4	5	2.9559e+01* (1.8736e+00)	2.9526e+01* (2.0954e+00)	2.9659e+01* (2.2066e+00)	2.7800e+01* (4.8406e+00)	2.5171e+01# (4.8764e+00)	2.9672e+01 (2.4855e+00)	1.9007e+01# (4.2288e+00)	2.7087e+01* (5.4767e+00)	2.7519e+01# (4.7998e+00)
	10	8.5577e+02* (1.6886e+02)	9.5990e+02* (4.9018e+01)	9.4982e+02* (5.3258e+01)	8.5562e+02# (1.7672e+02)	7.2490e+02# (2.0285e+02)	8.8766e+02# (1.3214e+02)	5.4882e+02# (9.2735e+01)	6.0438e+02# (1.2292e+02)	9.6139e+02 (4.3841e+01)
DTLZ5	5	9.2733e+02# (1.0838e+01)	9.4010e+02# (8.4791e+00)	9.4650e+02# (2.3033e+00)	9.3419e+02# (1.2233e+01)	9.3767e+02# (8.3438e+00)	9.4558e+02# (2.5381e+00)	9.3107e+02# (1.1124e+01)	9.4943e+02 (1.2505e+00)	9.4243e+02# (3.5922e+00)
	10	9.5104e+05# (7.4024e+03)	9.4040e+05# (1.5045e+04)	9.4353e+05# (1.1077e+04)	9.4885e+05# (6.4583e+03)	9.4335e+05# (1.2232e+04)	9.4688e+05# (9.9656e+03)	9.4897e+05# (7.8715e+03)	9.4510e+05# (1.1007e+04)	9.5609e+05 (5.7172e+03)
DTLZ6	5	1.2265e+05# (8.7332e+03)	1.2667e+05* (1.1925e+04)	1.2328e+05# (9.1050e+03)	1.2337e+05# (8.5112e+03)	1.3116e+05 (8.6004e+03)	1.2698e+05* (8.9950e+03)	1.1996e+05# (5.7836e+03)	1.2656e+05* (7.2094e+03)	1.3084e+05* (1.1419e+04)
	10	1.8194e+10# (1.2098e+09)	1.6469e+10# (1.6634e+09)	1.8028e+10# (2.3537e+09)	1.8152e+10# (5.2127e+08)	1.7855e+10# (2.3489e+09)	1.8356e+10# (1.9854e+09)	1.7128e+10# (8.8902e+08)	1.9452e+10 (1.4042e+09)	1.9311e+10* (1.8260e+09)
DTLZ7	5	1.0507e+01* (7.9807e-01)	1.0304e+01# (8.5076e-01)	9.7175e+00# (7.4730e-01)	1.0254e+01# (7.8088e-01)	1.0154e+01# (8.7074e-01)	9.5507e+00# (8.8203e-01)	1.0880e+01 (1.1113e-01)	9.6268e+00# (6.6748e-01)	1.0556e+01# (3.1990e-01)
	10	2.8014e-02* (5.0642e-02)	3.1968e-02 (5.1841e-02)	1.5463e-02* (3.0543e-02)	1.5851e-02* (2.4367e-02)	1.3862e-02* (2.9347e-02)	1.8781e-02* (2.1470e-02)	2.1861e-03# (4.9838e-03)	5.5583e-03# (8.6512e-03)	1.8967e-02* (1.6261e-02)

En la tabla 4.14 muestra el mejor algoritmo por instancia (diferencia significativa) o mejores algoritmos por instancia (sin diferencia significativa).

- Algoritmos HH_DTLZ-PCArea y HH_DTLZ-IGD. Estos algoritmos se destacan en la mayoría de los problemas DTLZ para 5 y 10 objetivos. HH_DTLZ-PCArea, en particular, fue el mejor algoritmo en términos de HV para múltiples problemas, como DTLZ1, DTLZ3, DTLZ5, y DTLZ6. Esto sugiere que ambos algoritmos son altamente efectivos para cubrir el frente de Pareto y proporcionar una buena aproximación del mismo, incluso en problemas con muchos objetivos. La métrica de Hipervolumen resalta la capacidad de estos algoritmos para alcanzar una gran cobertura del espacio objetivo, lo cual es crucial para evaluar el rendimiento general de los algoritmos en la aproximación al frente de Pareto.
- Rendimiento consistente en problemas de muchos objetivos. Tanto en los problemas DTLZ con 5 objetivos como en los de 10 objetivos, se observa que los mismos algoritmos tienden a destacar. Esto indica que estos enfoques son robustos y capaces de mantener un buen rendimiento, incluso cuando aumenta la complejidad del problema al incrementar el número de objetivos.
- Problemas complejos y empates frecuentes. En problemas como DTLZ2, DTLZ4 y DTLZ7 se observan numerosos empates entre los algoritmos, lo cual indica que estos problemas presentan mayores desafíos y dificultan la diferenciación clara del rendimiento de los algoritmos. Esto sugiere que la naturaleza de estos problemas, con una estructura de frente de Pareto más complicada, hace que los enfoques evaluados tengan desempeños similares y comparables.

En resumen, el análisis resalta la capacidad de los algoritmos para aproximarse al frente de Pareto, enfatizando la efectividad y consistencia de HH_DTLZ-PCArea y HH_DTLZ-IGD en diferentes niveles de complejidad de los problemas DTLZ. Estos algoritmos muestran un rendimiento sólido en términos de maximización del hipervolumen, lo cual es fundamental para la evaluación de la calidad de las soluciones en la optimización multi-objetivo.

Tabla 4.14. Resumen de los mejores algoritmos por problema.

PROBLEMA	Mejor algoritmo en 5 objetivos	Mejor algoritmo en 10 objetivos
DTLZ1	**HH_DTLZ-PCArea**	**HH_DTLZ-PCArea**
DTLZ2	HH_DTLZ-IGD HH_DTLZ-EUCL-MIN HH_DTLZ-PCArea	HH_DTLZ-IGD HH_DTLZ-PCArea
DTLZ3	**HH_DTLZ-PCArea**	TODOS LOS ALGORITMOS
DTLZ4	HH_DTLZ-TCHE-MIN HH_DTLZ-EUCL-AVG HH_DTLZ-EUCL-MED HH_DTLZ-EUCL-MIN HH_DTLZ-TCHE-AVG HH_DTLZ-IGD	HH_DTLZ-PCArea HH_DTLZ-EUCL-MED HH_DTLZ-EUCL-MIN
DTLZ5	**HH_DTLZ-IGD**	**HH_DTLZ-PCArea**
DTLZ6	HH_DTLZ-TCHE-MED HH_DTLZ-EUCL-MED HH_DTLZ-TCHE-MIN HH_DTLZ-IGD HH_DTLZ-PCArea	HH_DTLZ-IGD HH_DTLZ-PCArea
DTLZ7	**HH_DTLZ-IGD**	HH_DTLZ-EUCL-MED HH_DTLZ-EUCL-AVG HH_DTLZ-EUCL-MIN HH_DTLZ-TCHE-AVG HH_DTLZ-TCHE-MED HH_DTLZ-TCHE-MIN HH_DTLZ-PCArea

4.3.1.2 WFG con 5 y 10 Objetivos.

En la tabla 4.15 se observan los resultados de cada algoritmo representados por la media y desviación estándar de HV, los resultados de mejor desempeño están resaltados en fondo gris. Se realizó una prueba de hipótesis para observar si hay diferencias significativas mediante la prueba de pares de Wilcoxon con dos colas con grado de significancia de 5%. Para cada algoritmo de mejor desempeño en términos de HV se aplicó la prueba contra todos

sus pares. Los algoritmos que tienen diferencia significativa con el mejor resultado están marcados con # y los que no tienen diferencia significativa con *.

Tabla 4.15. Resultados términos de HV de 9 hiperheurísticas para las 18 instancias de prueba WFG.

PROBLEMA	M	HH_WFG-EUCL-AVG	HH_WFG-EUCL-MED	HH_WFG-EUCL-MIN	HH_WFG-TCHE-AVG	HH_WFG-TCHE-MED	HH_WFG-TCHE-MIN	HH_WFG-PR	HH_WFG-IGD	HH_WFG-PCArea
WFG1	5	1.8768e+03# (1.0489e+02)	1.8946e+03# (8.7931e+01)	1.9812e+03# (1.1479e+02)	1.9224e+03# (2.0774e+02)	1.9181e+03# (1.5824e+02)	1.9926e+03# (2.1560e+02)	1.8038e+03# (9.3785e+01)	2.0470e+03 (1.6891e+02)	1.8273e+03# (9.8292e+01)
	10	2.5525e+09* (1.4129e+08)	2.5711e+09 (7.9587e+07)	2.4343e+09# (1.0764e+08)	2.5087e+09* (1.2635e+08)	2.5602e+09* (1.2318e+08)	2.4636e+09* (1.8390e+08)	2.2230e+09# (9.1233e+07)	2.5421e+09* (1.7653e+08)	2.3703e+09# (7.8644e+07)
WFG2	5	8.0522e+03# (4.2063e+02)	8.0512e+03# (4.0793e+02)	8.2210e+03# (3.4915e+02)	8.0092e+03# (4.2408e+02)	8.1035e+03# (3.7638e+02)	8.2540e+03# (2.7055e+02)	7.9967e+03# (3.9969e+02)	8.3612e+03* (3.3600e+02)	8.4591e+03 (2.7568e+02)
	10	9.7550e+09# (1.6827e+08)	9.8188e+09# (2.2919e+08)	1.0188e+10# (3.8845e+08)	9.7895e+09# (3.2871e+08)	9.7632e+09# (2.6898e+08)	1.0022e+10# (4.1022e+08)	9.8616e+09# (5.9595e+08)	9.9423e+09# (4.2940e+08)	1.0434e+10 (3.7883e+08)
WFG3	5	5.6935e+03* (1.5435e+02)	5.6049e+03# (1.6193e+02)	5.3716e+03# (1.4615e+02)	5.6916e+03* (1.8507e+02)	5.7056e+03 (1.2430e+02)	5.3374e+03# (1.4438e+02)	5.3595e+03# (2.0273e+02)	5.3226e+03# (1.5088e+02)	5.5847e+03# (1.5863e+02)
	10	6.0532e+09* (3.2240e+08)	6.0351e+09* (2.8593e+08)	6.0375e+09* (2.7054e+08)	6.0633e+09* (2.7941e+08)	6.1536e+09* (2.6520e+08)	6.0272e+09* (2.8887e+08)	6.1248e+09* (2.5371e+08)	6.1226e+09* (2.4739e+08)	6.1609e+09 (3.2103e+08)
WFG4	5	3.2537e+03# (1.5519e+02)	3.4091e+03# (3.0343e+02)	4.3342e+03# (3.6100e+02)	3.2866e+03# (2.9087e+02)	3.3264e+03# (1.9661e+02)	4.3722e+03# (3.7116e+02)	3.4023e+03# (4.1956e+02)	4.4283e+03# (3.4973e+02)	4.8463e+03 (4.0912e+02)
	10	3.8197e+09# (1.7877e+08)	3.9114e+09* (2.2693e+08)	3.9435e+09* (1.7239e+08)	3.8462e+09# (1.7978e+08)	3.7791e+09# (1.9390e+08)	3.8470e+09# (2.3149e+08)	3.7966e+09# (1.7421e+08)	4.0546e+09 (2.5025e+08)	4.0285e+09* (2.8748e+08)
WFG5	5	4.0383e+03# (3.9135e+02)	4.0999e+03# (4.6678e+02)	4.0994e+03# (3.5106e+02)	4.0685e+03# (3.3620e+02)	4.1276e+03# (4.1012e+02)	4.0690e+03# (3.6384e+02)	4.0219e+03# (3.5661e+02)	4.4566e+03* (2.9583e+02)	4.5352e+03 (5.4833e+02)
	10	3.2856e+09# (1.1350e+08)	3.2060e+09# (2.9080e+08)	3.3373e+09# (2.5547e+08)	3.3285e+09# (1.5784e+08)	3.2623e+09# (2.5629e+08)	3.4284e+09* (2.5313e+08)	3.3308e+09* (2.9527e+08)	3.4745e+09 (1.8776e+08)	3.2805e+09# (3.6291e+08)
WFG6	5	3.7321e+03# (2.5110e+02)	3.9481e+03* (2.9299e+02)	3.7974e+03# (2.8476e+02)	3.7266e+03# (2.5843e+02)	3.9055e+03* (3.5941e+02)	3.7070e+03 (2.5332e+02)	3.6488e+03# (1.9769e+02)	4.0318e+03* (3.2229e+02)	4.1222e+03 (4.5527e+02)
	10	2.9741e+09* (1.8581e+08)	2.8321e+09# (2.0214e+08)	3.0199e+09 (1.5336e+08)	2.8387e+09# (2.2958e+08)	2.8341e+09# (2.2838e+08)	2.9104e+09# (2.6094e+08)	2.8814e+09# (2.5598e+08)	2.9638e+09* (2.8363e+08)	2.9524e+09* (2.8294e+08)
WFG7	5	4.2702e+03* (1.7409e+02)	4.2554e+03* (1.8438e+02)	4.1127e+03# (1.8566e+02)	4.2524e+03* (1.8522e+02)	4.2645e+03* (2.0836e+02)	4.1118e+03# (2.5071e+02)	4.0890e+03# (1.7209e+02)	4.3158e+03 (2.1543e+02)	4.2711e+03* (2.3272e+02)
	10	3.0898e+09# (2.0983e+08)	3.1882e+09* (2.5171e+08)	3.2410e+09* (3.1738e+08)	3.1079e+09# (1.8951e+08)	3.2036e+09# (2.2075e+08)	3.0647e+09# (2.3193e+08)	3.2490e+09* (2.1833e+08)	3.3342e+09 (2.7319e+08)	3.3292e+09* (2.6514e+08)
WFG8	5	3.5823e+03# (1.5197e+02)	3.5870e+03# (2.3036e+02)	4.1893e+03* (3.9317e+02)	3.6087e+03# (1.8932e+02)	3.6618e+03# (1.6098e+02)	4.2719e+03* (4.2509e+02)	3.6767e+03# (1.7444e+02)	4.3550e+03 (4.1765e+02)	4.3257e+03* (6.1360e+02)
	10	3.2954e+09# (4.4626e+08)	3.4040e+09# (5.5683e+08)	3.0139e+09# (4.6344e+08)	3.1780e+09# (4.2066e+08)	3.4228e+09# (3.9919e+08)	3.1000e+09# (3.9790e+08)	2.9937e+09# (3.4014e+08)	4.0129e+09 (4.2812e+08)	3.5477e+09# (6.7356e+08)
WFG9	5	3.8540e+03* (2.1541e+02)	3.8537e+03* (1.7071e+02)	3.7122e+03# (1.8901e+02)	3.8360e+03* (1.2832e+02)	3.8549e+03* (1.5955e+02)	3.6547e+03# (1.9588e+02)	3.8319e+03* (2.7105e+02)	3.6652e+03# (1.7155e+02)	3.9044e+03 (2.9502e+02)
	10	3.3253e+09 (2.2127e+08)	3.2130e+09* (2.5790e+08)	3.0339e+09# (1.6110e+08)	3.1592e+09# (2.5573e+08)	3.1586e+09# (2.2386e+08)	3.0507e+09# (1.8674e+08)	3.2970e+09* (2.5229e+08)	3.1654e+09# (1.7754e+08)	3.0834e+09# (2.1625e+08)

En la tabla 4.16 muestra el mejor algoritmo por instancia (diferencia significativa) o mejores algoritmos por instancia (sin diferencia significativa). A continuación, se describe algunos hallazgos de la Tabla 4.16.

- Problemas WFG1 y WFG4. En WFG1, el algoritmo HH_WFG-IGD aparece consistentemente tanto en 5 como en 10 objetivos, acompañado por otros algoritmos en el caso de 10 objetivos. En WFG4, HH_WFG-PCArea es el mejor en 5 objetivos, pero en 10 objetivos está acompañado por otros algoritmos, lo que podría sugerir que el aumento en el número de objetivos hace que otros enfoques sean competitivos. Los algoritmos HH_WFG-IGD y HH_WFG-PCArea aparecen frecuentemente como los mejores en ambos niveles de objetivos (5 y 10), lo cual indica su capacidad de obtener una buena cobertura del frente de Pareto en términos de hipervolumen
- Escalabilidad: En general, para problemas de 10 objetivos se observa un incremento en la cantidad de algoritmos que empatan como mejores en comparación con los problemas de 5 objetivos. Esto podría reflejar que la dificultad de aproximarse al frente de Pareto aumenta con la cantidad de objetivos, y diferentes algoritmos logran resultados competitivos debido a la mayor complejidad.
- Complejidad en WFG3: En el caso del problema WFG3 con 10 objetivos, todos los algoritmos muestran un rendimiento equivalente, lo cual indica que este problema tiene una complejidad tal que ningún algoritmo logra destacarse claramente en términos de rendimiento para aproximar el frente de Pareto.
- Consistencia entre Algoritmos para Problemas Similares: Se observa que, en varios problemas, algoritmos como HH_WFG-EUCL-AVG, HH_WFG-TCHE-AVG, y HH_WFG-IGD aparecen repetidamente, lo cual podría indicar su efectividad general en problemas de optimización multi-objetivo. Estos patrones sugieren una consistencia en el rendimiento de estos enfoques para cubrir adecuadamente las soluciones en el espacio objetivo.

En resumen, la variedad de algoritmos que destacan indica que la aproximación al frente de Pareto en problemas WFG depende de la complejidad del problema y del número de objetivos. Ningún algoritmo es claramente superior en todos los casos, y en los problemas con mayor número de objetivos, varios enfoques logran un rendimiento similar. Sin embargo,

destacan HH_WFG-IGD y HH_WFG-PCA en WFG1 y WFG4 de 5 objetivos respectivamente.

Tabla 4.16. Resumen de los mejores algoritmos por problema WFG.

PROBLEMA	Mejor algoritmo en 5 objetivos	Mejor algoritmo en 10 objetivos
WFG1	**HH_WFG-IGD**	HH_WFG-EUCL-MED HH_WFG-EUCL-AVG HH_WFG-TCHE-AVG HH_WFG-TCHE-MED HH_WFG-IGD
WFG2	HH_WFG-PCArea HH_WFG-IGD	**HH_WFG-PCArea**
WFG3	HH_WFG-TCHE-MED HH_WFG-EUCL-AVG HH_WFG-TCHE-AVG	TODOS LOS ALGORITMOS
WFG4	**HH_WFG-PCArea**	HH_WFG-IGD HH_WFG-EUCL-MED HH_WFG-EUCL-MIN HH_WFG-PCArea
WFG5	HH_WFG-PCArea HH_WFG-IGD	HH_WFG-IGD HH_WFG-TCHE-MIN HH_WFG-PR
WFG6	HH_WFG-PCArea HH_WFG-EUCL-MED HH_WFG-TCHE-MED HH_WFG-IGD	HH_WFG-EUCL-MIN HH_WFG-EUCL-AVG HH_WFG-IGD HH_WFG-PCArea
WFG7	HH_WFG-IGD HH_WFG-EUCL-AVG HH_WFG-EUCL-MED HH_WFG-TCHE-AVG HH_WFG-TCHE-MED HH_WFG-PCArea	HH_WFG-IGD HH_WFG-EUCL-MED HH_WFG-EUCL-MIN HH_WFG-PR HH_WFG-PCArea
WFG8	HH_WFG-IGD HH_WFG-EUCL-MIN HH_WFG-TCHE-MIN HH_WFG-PCArea	**HH_WFG-IGD**
WFG9	HH_WFG-PCArea HH_WFG-EUCL-AVG HH_WFG-EUCL-MED HH_WFG-TCHE-AVG HH_WFG-TCHE-MED HH_WFG-PR	HH_WFG-EUCL-AVG HH_WFG-EUCL-MED HH_WFG-PR

4.3.1.3 PSP para 3 y 5 Objetivos.

En la tabla 4.17 se observan los resultados para PSP de cada algoritmo representados por la media y desviación estándar de HV. los resultados de mejor desempeño están resaltados en fondo gris. Se realizó una prueba de hipótesis para observar si hay diferencias significativas mediante la prueba de pares de Wilcoxon con dos colas con grado de significancia de 5%. Para cada algoritmo de mejor desempeño en términos de HV se aplicó la prueba contra todos sus pares. Los algoritmos que tienen diferencia significativa con el mejor resultado están marcados con # y los que no tienen diferencia significativa con *.

Tabla 4.17. Resultados términos de HV de 10 hiperheurísticas para las 12 instancias de prueba de gran escala PSP.

PROBLEMA	M	HH_PSP-EUCL-AVG	HH_PSP-EUCL-MED	HH_PSP-EUCL-MIN	HH_PSP-TCHE-AVG	HH_PSP-TCHE-MED	HH_PSP-TCHE-MIN	HH_PSP-PR	HH_PSP-HV	HH_PSP-IGD	HH_PSP-PCArea
PSP1	3	5.4452e+12# (5.4409e+11)	5.5506e+12# (7.6646e+11)	6.5119e+12# (6.4195e+11)	5.5544e+12# (5.7325e+11)	5.4351e+12# (6.9514e+11)	6.3897e+12# (6.4532e+11)	5.6440e+12# (3.0954e+11)	7.1851e+12* (1.5274e+11)	7.2053e+12* (1.7191e+11)	7.2740e+12 (1.8029e+11)
	5	1.3276e+21# (1.7922e+20)	1.3300e+21# (2.2405e+20)	1.6787e+21# (2.9146e+20)	1.3898e+21# (1.8830e+20)	1.3834e+21# (2.4493e+20)	1.7653e+21# (3.3713e+20)	1.9616e+21# (9.7341e+19)	2.2868e+21* (9.9230e+19)	2.2422e+21# (1.0314e+20)	2.3186e+21 (9.0342e+19)
PSP2	3	3.5004e+12# (1.7807e+10)	3.5069e+12# (1.8843e+10)	3.5296e+12# (1.4116e+10)	3.5103e+12# (2.3138e+10)	3.5079e+12# (1.5706e+10)	3.5347e+12# (1.5788e+10)	3.5812e+12# (2.1152e+10)	3.5836e+12# (2.3835e+10)	3.5900e+12# (2.3983e+10)	3.6120e+12 (1.8242e+10)
	5	7.8631e+20# (4.9497e+18)	7.8786e+20# (6.4663e+18)	7.9063e+20# (4.8243e+18)	7.8693e+20# (8.7588e+18)	7.8670e+20# (4.7586e+18)	7.9049e+20# (7.9011e+18)	8.0435e+20# (6.5133e+18)	8.0917e+20# (6.5280e+18)	8.0036e+20# (7.1399e+18)	8.1528e+20 (3.7772e+18)
PSP3	3	5.4654e+12# (5.3228e+10)	5.4896e+12# (6.0162e+10)	5.4896e+12# (2.7898e+10)	5.4686e+12# (6.0943e+10)	5.4582e+12# (5.1725e+10)	5.4887e+12# (3.7767e+10)	5.5907e+12# (3.3265e+10)	5.5875e+12# (3.3938e+10)	5.6699e+12# (2.7503e+10)	5.7354e+12 (2.9554e+10)
	5	1.5385e+21# (2.3017e+19)	1.5322e+21# (1.5082e+19)	1.5457e+21# (7.4833e+18)	1.5390e+21# (2.1017e+19)	1.5293e+21# (1.5717e+19)	1.5425e+21# (9.0499e+18)	1.5820e+21# (2.0244e+19)	1.5817e+21# (1.0131e+19)	1.6142e+21# (1.2866e+19)	1.6439e+21 (1.4240e+19)
PSP4	3	3.6035e+13# (3.7062e+12)	3.5542e+13# (4.6780e+12)	4.3805e+13# (4.7126e+12)	3.6827e+13# (3.9794e+12)	3.4466e+13# (3.1633e+12)	4.4589e+13# (4.6716e+12)	4.4692e+13# (2.7178e+12)	5.0573e+13* (1.4653e+12)	5.0320e+13* (1.1815e+12)	5.0865e+13 (1.2839e+12)
	5	3.5555e+22# (6.4587e+21)	3.1433e+22# (2.5619e+21)	4.3209e+22# (8.6477e+21)	3.4731e+22# (5.4822e+21)	3.1909e+22# (3.4537e+21)	4.9488e+22# (9.9441e+21)	4.9258e+22# (6.5788e+21)	6.2460e+22# (2.8308e+21)	6.1660e+22# (2.4896e+21)	6.3975e+22 (2.5523e+21)
PSP5	3	2.7680e+13# (1.6147e+11)	2.7617e+13# (1.2434e+11)	2.7838e+13# (1.7035e+11)	2.7648e+13# (1.2605e+11)	2.7568e+13# (1.1889e+11)	2.7812e+13# (1.5860e+11)	2.8047e+13# (1.5362e+11)	2.8152e+13# (1.5418e+11)	2.8201e+13# (1.5346e+11)	2.8351e+13 (9.1757e+10)
	5	2.4623e+22# (1.3812e+20)	2.4800e+22# (1.6658e+20)	2.4994e+22# (1.7442e+20)	2.4697e+22# (2.2532e+20)	2.4792e+22# (1.7623e+20)	2.4940e+22# (1.9864e+20)	2.5034e+22# (1.7094e+20)	2.5271e+22# (1.2460e+20)	2.5063e+22# (1.3923e+20)	2.5356e+22 (1.3250e+20)
PSP6	3	4.3902e+13# (4.1363e+11)	4.4297e+13# (6.2184e+11)	4.5157e+13# (5.0916e+11)	4.3820e+13# (4.0585e+11)	4.4308e+13# (6.7814e+11)	4.5030e+13# (4.4256e+11)	4.5225e+13# (1.9898e+11)	4.5278e+13# (3.1929e+11)	4.5865e+13# (2.4218e+11)	4.6266e+13 (2.0350e+11)
	5	5.5348e+22# (8.3035e+20)	5.5352e+22# (1.0539e+21)	5.6474e+22# (7.7101e+20)	5.5614e+22# (7.7320e+20)	5.5228e+22# (1.0540e+21)	5.6616e+22# (8.8246e+20)	5.7369e+22# (3.5440e+20)	5.7598e+22# (3.3159e+20)	5.8442e+22# (4.1427e+20)	5.9554e+22 (4.2335e+20)

En la tabla 4.18 muestra el mejor algoritmo por instancia (diferencia significativa) o mejores algoritmos (sin diferencia significativa.

Tabla 4.18. Resumen de los mejores algoritmos de la Tabla 4.17 para el problema PSP.

PROBLEMA	**Mejor algoritmo (m=3, D=500)**	**Mejor algoritmo (m=5, D=500)**
PSP1	HH_PSP-PCArea HH_PSP-HV HH_PSP-IGD	HH_PSP-PCArea HH_PSP-HV
PSP2	HH_PSP-PCArea	HH_PSP-PCArea
PSP3	HH_PSP-PCArea	HH_PSP-PCArea

A continuación, se describe algunos hallazgos de la Tabla 4.18:

- Consistencia de un algoritmo: En todos los problemas (PSP1 a PSP6) con 500 variables de decisión (3 y 5 objetivos) y 1000 variables de decisión (3 y 5 objetivos) para ambos niveles de objetivos (3 y 5 objetivos), el algoritmo HH_PSP-PCArea aparecen consistentemente como el mejor o uno de los mejores.
- Mejor algoritmo: En la prueba de pares mediante Wilcoxon se encontró que en 9 de 12 pruebas de gran escala el algoritmo hiperheurístico HH_PSP-PCArea.
- PSP1 y PSP4 (m=3, D=500 y m=3, D=1000)
    - No hay diferencia significativa entre HH_PSP-PCArea, HH_PSP-HV y HH_PSP-IGD.
    - Indica que las tres métricas son equivalentes en estos casos.

En la tabla 4.19 se muestra el mejor algoritmo por instancia (diferencia significativa) o mejores algoritmos (sin diferencia significativa).

Tabla 4.19. Resumen de los mejores algoritmos de la Tabla 4.17 para el problema PSP.

PROBLEMA	**Mejor algoritmo (m=3, D=1000)**	**Mejor algoritmo (m=5, D=1000)**
PSP4	HH_PSP-PCArea HH_PSP-HV HH_PSP-IGD	HH_PSP-PCArea
PSP5	HH_PSP-PCArea	HH_PSP-PCArea
PSP6	HH_PSP-PCArea	HH_PSP-PCArea

A continuación, se describe algunos hallazgos de la Tabla 4.19:

- PSP1 (m=5, D=500) y PSP4 (m=5, D=1000)
    - HH_PSP-IGD ya no es competitivo cuando m=5.
    - HH_PSP-HV deja de ser relevante cuando D=1000.
    - HH_PSP-PCArea es el único algoritmo que mantiene su buen desempeño en todos los escenarios.
- PSP2, PSP3, PSP5 y PSP6
    - HH_PSP-PCArea es el único mejor algoritmo en todos los casos.
    - Esto refuerza su robustez y superioridad en estos problemas.

En resumen, PCArea es competitivo en todos los casos y dimensiones. HV e IGD son útiles en algunos casos, pero pierden relevancia con más objetivos o mayor escala. Cuando hay múltiples mejores algoritmos, PCArea siempre está presente.

4.3.2 Análisis de la Capacidad de los Algoritmos para Aproximarse a la Hacia la Región de Interés (ROI) de un Decisor

Mediante un análisis de Borda, se analiza la capacidad de los algoritmos de alcanzar la región de interés determinada por el mejor compromiso. El mejor compromiso (BC) refleja las metas, objetivos y deseos del decisor (DM) tanto como sea posible. Como se sabe, en problemas reales de optimización no es posible saber de antemano cuál es el BC; por lo tanto, los algoritmos HH que emplean preferencias calculan el mejor compromiso en tiempo de ejecución. Este sirve como referencia al momento de aplicar métodos de selección de LLHs basados en preferencias para guiar al algoritmo hacia la ROI. A continuación, se describen los hallazgos que fueron encontrados como resultado de las pruebas para DTLZ, WFG y PSP con preferencias de un decisor.

4.3.2.1 DTLZ para 5 Objetivos con Preferencias.

La tabla 4.20 muestra los mejores algoritmos para 5 objetivos, un solo algoritmo (diferencia significativa) y varios algoritmos (sin diferencia significativa). Podemos observar en la tabla lo siguiente:

- El algoritmo HH_DTLZ_PCArea, no utiliza un método de selección enfocado en aproximar la ROI, pero destaca en algunas métricas y problemas. Este comportamiento se puede observar en DTLZ1, DTLZ3 y DTLZ5, donde se muestra como el mejor en varias métricas (avg_eucl, min_eucl, avg_tche, min_tche, med_eucl, med_tche, y zPx_out2).
- HH_DTLZ_IGD no está diseñado para aproximar la ROI. Sin embargo, tiene un buen rendimiento en la métrica xSz_out1, apareciendo consistentemente como el mejor para los problemas DTLZ1, DTLZ2, DTLZ3 y DTLZ5.
- Las hiperheurísticas que poseen métodos de selección basados en preferencias (como HH_DTLZ_TCHE_MED, HH_DTLZ_TCHE_MIN, HH_DTLZ_PR, etc.) muestran un rendimiento consistente al aproximar la ROI. Esto se puede observar, en DTLZ2 y DTLZ4, donde las métricas de promedio, mediana, y mínimo (distancia euclidiana y Tchebycheff) presentan en su mayoría a los algoritmos basados en preferencias, sugiriendo que, cuando se requiere precisión sobre la región preferida, estos algoritmos pueden ser más adecuados, dado lo mostrado en la Tabla 4.20.

Tabla 4.20. Mejores algoritmos por métrica de desempeño para el problema DTLZ con preferencia de un decisor para 5 objetivos.

Problema	avg_eucl	min_eucl	avg_tche	min_tche	med_eucl	med_tche	xSz_out1	zPx_out2
DTLZ1	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_IGD	HH_DTLZ_PCArea
DTLZ2	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_MED	HH_DTLZ_EUCL_MED	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_MED	HH_DTLZ_IGD	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_MED	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_MED	HH_DTLZ_IGD	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_PR
DTLZ3	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_TCHE_MED	HH_DTLZ_TCHE_MIN	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_PR
DTLZ4	HH_DTLZ_TCHE_MED, HH_DTLZ_PR	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MIN, HH_DTLZ_TCHE_MIN, HH_DTLZ_IGD	HH_DTLZ_TCHE_MED, HH_DTLZ_PR	HH_DTLZ_TCHE_MIN, HH_DTLZ_PR	HH_DTLZ_TCHE_MED, HH_DTLZ_PR	HH_DTLZ_TCHE_MED, HH_DTLZ_PR	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MIN	Todos los algoritmos
DTLZ5	HH_DTLZ_EUCL_MIN	HH_DTLZ_EUCL_MIN, HH_DTLZ_TCHE_MIN, HH_DTLZ_PCArea	HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_MED, HH_DTLZ_PR, HH_DTLZ_PCArea	HH_DTLZ_PR	HH_DTLZ_EUCL_MIN, HH_DTLZ_TCHE_MIN	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_AVG, HH_DTLZ_TCHE_MED, HH_DTLZ_TCHE_MIN, HH_DTLZ_PR, HH_DTLZ_PCArea	HH_DTLZ_IGD	HH_DTLZ_EUCL_AVG, HH_DTLZ_PR
DTLZ6	HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_MED	HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_MED HH_DTLZ_IGD, HH_DTLZ_PCArea	HH_DTLZ_EUCL_MED	HH_DTLZ_TCHE_MED, HH_DTLZ_PCArea	HH_DTLZ_EUCL_MED	HH_DTLZ_EUCL_MED	Todos los algoritmos	HH_DTLZ_PCArea
DTLZ7	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_AVG, HH_DTLZ_TCHE_MED, HH_DTLZ_PR	HH_DTLZ_PR	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_AVG, HH_DTLZ_TCHE_MED, HH_DTLZ_PR	HH_DTLZ_PR	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_AVG, HH_DTLZ_TCHE_MED, HH_DTLZ_PR	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_AVG, HH_DTLZ_TCHE_MED, HH_DTLZ_PR	Todos los algoritmos	HH_DTLZ_PR

4.3.2.2 DTLZ para 10 Objetivos con Preferencias.

La tabla 4.21 muestra los mejores algoritmos para instancias DTLZ de 10 objetivos. En cada celda se muestra un solo algoritmo (diferencia significativa) o varios algoritmos (sin diferencia significativa). Podemos observar en la tabla 4.21 lo siguiente:

- Se observa que, HH_DTLZ_PCArea y HH_DTLZ_IGD (no basados en preferencias) tienen un rendimiento sobresaliente (diferencia significativa) en varias métricas y problemas. Esto puede indicar que su enfoque generalista sigue siendo efectivo en algunos contextos. Las pruebas realizadas enfocadas en aproximar el PF (Tablas 4.15. 4.17 y 4.19) muestran un destacado desempeño de ambas y esto tiene repercusiones en el proceso de aproximar la ROI.
- Hiperheurísticas con operadores de selección basados en preferencias como HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_MED, y HH_DTLZ_PR, logran destacarse (diferencia significativa), en varias de las instancias de prueba DTLZ.
- También podemos observar en general muchos empates (sin diferencia significativa) de algoritmos en ciertos problemas como DTLZ2**,** DTLZ4**, y** DTLZ7 parecen ser los problemas más complejos, ya que presentan muchos empates y no hay un algoritmo que domine claramente en todas las métricas, pero a la vez podemos observar que en su mayoría dominan los algoritmos basados en preferencias al aproximar a la ROI.

En resumen, algoritmos como HH_DTLZ_EUCL_MED, HH_DTLZ_TCHE_MED, HH_DTLZ_PR, HH_DTLZ_PCArea y HH_DTLZ_IGD tienen rendimiento con diferencia significativa en varios problemas y métricas. Sin embargo, existen rendimientos similares en varios problemas y métricas, no encontrando diferencia significativa entre algoritmos que en su mayoría están basados en preferencias.

Tabla 4.21. Mejores algoritmos por métrica de desempeño para el problema DTLZ con preferencia de un decisor para 10 objetivos.

Problema	avg_eucl	min_eucl	avg_tche	min_tche	med_eucl	med_tche	xSz_out1	zPx_out2
DTLZ1	HH_DTLZ_PCArea	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_EUCL_MIN, HH_DTLZ_TCHE_AVG, HH_DTLZ_TCHE_MED, HH_DTLZ_TCHE_MIN, HH_DTLZ_IGD, HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_EUCL_MED, HH_DTLZ_EUCL_MIN, HH_DTLZ_IGD
DTLZ2	Todos los algoritmos	HH_DTLZ_IGD, HH_DTLZ_PCArea	HH_DTLZ_PR	HH_DTLZ_PCArea	Todos los algoritmos	Todos los algoritmos	HH_DTLZ_PR	HH_DTLZ_EUCL_MIN
DTLZ3	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	HH_DTLZ_PCArea	Todos los algoritmos	Todos los algoritmos
DTLZ4	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_EUCL_MIN	HH_DTLZ_EUCL_MED	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_EUCL_MIN	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_EUCL_MIN, HH_DTLZ_TCHE_AVG, HH_DTLZ_TCHE_MIN, HH_DTLZ_PCArea	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_EUCL_MIN	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_EUCL_MIN	HH_DTLZ_IGD	HH_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED
DTLZ5	HH_DTLZ_EUCL_AVG	HH_DTLZ_EUCL_AVG	HH_DTLZ_EUCL_AVG	HH_DTLZ_EUCL_AVG	HH_DTLZ_EUCL_AVG	HH_DTLZ_EUCL_AVG	HH_DTLZ_EUCL_AVG, HH_DTLZ_TCHE_AVG, HH_DTLZ_PCArea	HH_DTLZ_IGD
DTLZ6	HH_DTLZ_IGD	HH_DTLZ_EUCL_AVG, HH_DTLZ_IGD, HH_DTLZ_PCArea	HH_DTLZ_IGD	HH_DTLZ_EUCL_AVG, HH_DTLZ_IGD, HH_DTLZ_PCArea	HH_DTLZ_IGD	HH_DTLZ_IGD	Todos los algoritmos	Todos los algoritmos
DTLZ7	HH_DTLZ_EUCL_AVG, HH_DTLZ_PR	H_DTLZ_EUCL_AVG, HH_DTLZ_EUCL_MED, HH_DTLZ_PR	HH_DTLZ_PR	HH_DTLZ_PR	HH_DTLZ_PR	HH_DTLZ_PR	Todos los algoritmos	Todos los algoritmos

4.3.2.3 WFG para 5 Objetivos con Preferencias.

Los problemas WFG son problemas más desafiantes que la suite de problemas DTLZ, por lo que son un benchmark adecuado para problemas de múltiples y muchos objetivos. La tabla 4.22 muestra los mejores algoritmos para instancias WFG de 5 objetivos. En cada celda se muestra un solo algoritmo (diferencia significativa) o varios algoritmos (sin diferencia significativa). Podemos observar en la tabla 4.22 lo siguiente:

- Se observa que, los algoritmos HH_WFG_PR y HH_WFG_IGD parecen destacarse frecuentemente en varias métricas, particularmente en WFG1 y WFG2, lo que sugiere que son buenos para aproximarse a la ROI en estas instancias. En forma similar, HH_WFG_PR aproxima la ROI en forma destacada en los problemas WFG1 y WFG2 en varias métricas.
- HH_WFG_EUCL_AVG, HH_WFG_EUCL_MED, HH_WFG_TCHE_AVG y HH_WFG_TCHE_AVG aparecen consistentemente en varios problemas (WFG3, WFG4, WFG5, WFG6, WFG7, WFG8, y WFG9), destacando en métricas de promedio tanto euclidiano como de Tchebycheff.
- Los problemas desafiantes WFG para 5 objetivos son WFG4, WFG7, WFG8, y WFG9 donde se presentan muchos empates de los algoritmos, destacando que son en su mayoría estos algoritmos basados en preferencias y muy pocas veces de corte general**.**

En resumen, HH_WFG_PR (basado en preferencias) **y** HH_WFG_IGD (no basado en preferencias) son algoritmos que tiene diferencia significativa con el resto de algoritmos en más problemas y métricas, sugiriendo que son opciones robustas para la optimización en el contexto de las instancias WFG presentadas de 5 objetivos. Por su parte, HH_WFG_EUCL_AVG, HH_WFG_EUCL_MED, HH_WFG_TCHE_AVG y HH_WFG_TCHE_AVG son consistentes en WFG3, WFG4, WFG5, WFG6, WFG7, WFG8, y WFG9, de entre los cuales hay problemas más desafiantes que otros.

Tabla 4.22. Mejores algoritmos por métrica de desempeño para el problema WFG con preferencia de un decisor para 5 objetivos.

Problema	avg_eucl	min_eucl	avg_tche	min_tche	med_eucl	med_tche	xSz_out1	zPx_out2
WFG1	HH_WFG_PR	Todos los algoritmos	HH_WFG_IGD	HH_WFG_IGD	HH_WFG_PR	HH_WFG_IGD	Todos los algoritmos	Todos los algoritmos
WFG2	HH_WFG_PR	Todos los algoritmos	HH_WFG_PR	Todos los algoritmos	HH_WFG_PR	HH_WFG_TCHE_MED HH_WFG_PR	Todos los algoritmos	HH_WFG_PR
WFG3	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_EUCL_AVG	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	Todos los algoritmos	HH_WFG_PCArea
WFG4	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_EUCL_AVG HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_EUCL_MED	HH_WFG_EUCL_MIN HH_WFG_TCHE_MIN	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	Todos los algoritmos	HH_WFG_EUCL_MIN HH_WFG_TCHE_MIN HH_WFG_IGD HH_WFG_PCArea
WFG5	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED	Todos los algoritmos	HH_WFG_EUCL_AVG	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED	Todos los algoritmos	HH_WFG_EUCL_AVG	Todos los algoritmos	HH_WFG_PR
WFG6	HH_WFG_EUCL_MED	HH_WFG_EUCL_MED	HH_WFG_EUCL_MED	HH_WFG_TCHE_MED	HH_WFG_EUCL_AVG, HH_WFG_PR	Todos los algoritmos	Todos los algoritmos	HH_WFG_PCArea
WFG7	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_EUCL_MIN HH_WFG_TCHE_MIN HH_WFG_PR, HH_WFG_IGD, HH_WFG_PCArea	HH_WFG_TCHE_MIN HH_WFG_PR HH_WFG_IGD HH_WFG_PCArea
WFG8	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG	HH_WFG_PCArea	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_EUCL_MIN HH_WFG_TCHE_AVG HH_WFG_TCHE_MED HH_WFG_TCHE_MIN HH_WFG_IGD	HH_WFG_TCHE_MIN	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	Todos los algoritmos	HH_WFG_EUCL_MIN HH_WFG_TCHE_MIN HH_WFG_IGD HH_WFG_PCArea	HH_WFG_EUCL_AVG HH_WFG_TCHE_AVG
WFG9	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	HH_WFG_EUCL_MED HH_WFG_TCHE_MED	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_MED	HH_WFG_EUCL_MED	HH_WFG_EUCL_AVG HH_WFG_EUCL_MED HH_WFG_TCHE_AVG HH_WFG_TCHE_MED	Todos los algoritmos	HH_WFG_EUCL_MIN HH_WFG_TCHE_MIN HH_WFG_PR HH_WFG_IGD HH_WFG_PCArea	HH_WFG_TCHE_MIN HH_WFG_PR HH_WFG_IGD HH_WFG_PCArea

4.3.2.4 WFG para 10 Objetivos con Preferencias

La tabla 4.23 muestra los mejores algoritmos para instancias WFG de 10 objetivos. En cada celda se muestra un solo algoritmo (diferencia significativa) o varios algoritmos (sin diferencia significativa). Podemos observar en la tabla 4.23 lo siguiente:

- El algoritmo HH_WFG_PR se destaca claramente en muchas de las métricas a lo largo de la mayoría de los problemas WFG con 10 objetivos, particularmente en WFG1**,** WFG2**,** WFG4**,** WFG6**, y** WFG7. Esto indica que la capacidad de aproximarse a la ROI de HH_WFG_PR tiene un rendimiento consistente y efectivo en múltiples situaciones.
- En ciertos problemas, como WFG3, WFG5, y WFG9, se observa que hay un empate entre varios algoritmos para algunas métricas, incluso llegando al caso de "Todos los algoritmos" teniendo el mismo desempeño. Esto indica que estos problemas presentan un nivel de complejidad considerable, donde todos los enfoques presentan dificultades similares para diferenciarse significativamente.
- Los algoritmos HH_WFG_IGD y HH_WFG_PCArea, tienen un desempeño mixto. En algunos casos, como en WFG3 y WFG8, estos algoritmos aún logran un buen rendimiento, pero no son los algoritmos dominantes en la mayoría de las métricas.

En general, HH_WFG_PR es el algoritmo basado en preferencias que se destaca en la mayoría de los problemas y métricas. Los problemas WFG3, WFG5, y WFG9 presentan más complejidad, dado que múltiples algoritmos tienen un desempeño similar, mientras que en otros problemas los algoritmos basados en preferencias se destacan claramente. Los algoritmos no basados en preferencias (HH_WFG_IGD y HH_WFG_PCArea) no son tan competitivos cuando se busca la región de interés del decisor para estas instancias de prueba, lo cual resalta la importancia de utilizar enfoques diseñados específicamente para el problema en cuestión.

Tabla 4.23. Mejores algoritmos por métrica de desempeño para el problema WFG con preferencia de un decisor para 10 objetivos.

Problema	avg_eucl	min_eucl	avg_tche	min_tche	med_eucl	med_tche	xSz_out1	zPx_out2
WFG1	HH_WFG_PR	HH_WFG_PR	HH_WFG_EUCL_MED	HH_WFG_EUCL_MED	HH_WFG_PR	HH_WFG_EUCL_MED	Todos los algoritmos	Todos los algoritmos
WFG2	HH_WFG_PR	HH_WFG_PR	HH_WFG_PR	HH_WFG_PR	HH_WFG_PR	HH_WFG_TCHE_MED	Todos los algoritmos	HH_WFG_PCArea
WFG3	HH_WFG_PR	HH_WFG_EUCL_MIN, HH_WFG_TCHE_MIN, HH_WFG_PR, HH_WFG_PCArea	HH_WFG_PR	HH_WFG_PR	HH_WFG_EUCL_MIN, HH_WFG_TCHE_MIN, HH_WFG_PR, HH_WFG_PCArea	Todos los algoritmos	Todos los algoritmos	HH_WFG_EUCL_AVG, HH_WFG_EUCL_MED, HH_WFG_EUCL_MIN, HH_WFG_TCHE_AVG, HH_WFG_TCHE_MED, HH_WFG_TCHE_MIN, HH_WFG_IGD, HH_WFG_PCArea
WFG4	HH_WFG_PR	HH_WFG_PR	HH_WFG_IGD	HH_WFG_IGD	HH_WFG_PR	HH_WFG_PR	Todos los algoritmos	HH_WFG_PCArea
WFG5	Todos los algoritmos	HH_WFG_EUCL_AVG	HH_WFG_EUCL_AVG	HH_WFG_EUCL_AVG	Todos los algoritmos	HH_WFG_EUCL_AVG, HH_WFG_IGD	Todos los algoritmos	HH_WFG_EUCL_AVG
WFG6	HH_WFG_PR	HH_WFG_EUCL_MIN, HH_WFG_TCHE_MIN, HH_WFG_PR, HH_WFG_PCArea	Todos los algoritmos	Todos los algoritmos	HH_WFG_PR	HH_WFG_TCHE_MIN, HH_WFG_PR	Todos los algoritmos	HH_WFG_PCArea
WFG7	HH_WFG_PR	Todos los algoritmos	HH_WFG_PR	HH_WFG_PR	HH_WFG_PR	HH_WFG_PR	Todos los algoritmos	HH_WFG_PCArea
WFG8	HH_WFG_TCHE_MED	Todos los algoritmos	HH_WFG_IGD	HH_WFG_EUCL_AVG	HH_WFG_TCHE_MED	HH_WFG_PR, HH_WFG_PCArea	Todos los algoritmos	Todos los algoritmos
WFG9	Todos los algoritmos	Todos los algoritmos	HH_WFG_EUCL_AVG, HH_WFG_TCHE_AVG, HH_WFG_TCHE_MED, HH_WFG_PR	Todos los algoritmos	HH_WFG_PCArea	HH_WFG_EUCL_AVG, HH_WFG_TCHE_MED, HH_WFG_PR	Todos los algoritmos	HH_WFG_PR

4.3.2.5 PSP para 3 Objetivos con Preferencias.

Las instancias de prueba de PSP son problemas de gran escala, lo cual añade complejidad a la búsqueda, pues como se menciona en esta sección estas instancias cuentan con preferencias de un DM. La tabla 4.24 muestra los mejores algoritmos para instancias PSP de 3 objetivos. En cada celda se muestra un solo algoritmo (diferencia significativa) o varios algoritmos (sin diferencia significativa) que son los mejores para la métrica y problema. Podemos observar en la Tabla 4.24 lo siguiente:

- Solo algoritmos basados en preferencias tienen los mejores resultados al acercarse a la región de interés en 6 de 8 métricas.
- Las métricas xSz_out1 y zPx_out2 muestran que todos los algoritmos tienen el mismo desempeño, lo que sugiere que estas métricas no son lo suficientemente discriminativas para evaluar la calidad de los algoritmos en estos problemas.
- Los algoritmos HH_PSP_EUCL_MED y HH_PSP_TCHE_MED son los más consistentes en PSP1.
- HH_PSP_EUCL_AVG es dominante en PSP2, pero en algunas métricas comparte con HH_PSP_EUCL_MED y HH_PSP_TCHE_MED.
- En PSP3 los mejores algoritmos son HH_PSP_EUCL_AVG, HH_PSP_EUCL_MED, HH_PSP_TCHE_AVG y HH_PSP_TCHE_MED.
- En PSP4, HH_PSP_EUCL_AVG, HH_PSP_EUCL_MED y HH_PSP_TCHE_MED son los algoritmos más consistentes.
- H_PSP_TCHE_MED domina en PSP5, lo que indica que este problema favorece estrategias basadas en la métrica mediana de Tchebycheff
- HH_PSP_EUCL_AVG, HH_PSP_TCHE_AVG son los más consistentes en PSP6.

Tabla 4.24. Mejores algoritmos por métrica de desempeño para el problema PSP con preferencia de un decisor para 3 objetivos.

Problema	avg_eucl	min_eucl	avg_tche	min_tche	med_eucl	med_tche	xSz_out1	zPx_out2
PSP1	HH_PSP_TCHE_MED	HH_PSP_EUCL_AVG HH_PSP_EUCL_MED HH_PSP_TCHE_MED	H_PSP_EUCL_MED H_PSP_TCHE_MED	HH_PSP_EUCL_AVG HH_PSP_EUCL_MED H_PSP_TCHE_MED	H_PSP_EUCL_MED H_PSP_TCHE_MED	HH_PSP_EUCL_MED HH_PSP_TCHE_MED	Todos los algoritmos	Todos los algoritmos
PSP2	HH_PSP_EUCL_AVG	HH_PSP_EUCL_AVG H_PSP_EUCL_MED HH_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG	HH_PSP_EUCL_AVG HH_PSP_EUCL_MED HH_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG	HH_PSP_EUCL_AVG	Todos los algoritmos	Todos los algoritmos
PSP3	HH_PSP_EUCL_AVG H_PSP_EUCL_MED H_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG H_PSP_EUCL_MED HH_PSP_TCHE_AVG HH_PSP_TCHE_MED	HH_PSP_EUCL_AVG HH_PSP_EUCL_MED H_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG H_PSP_EUCL_MED H_PSP_TCHE_AVG HH_PSP_TCHE_MED	HH_PSP_EUCL_AVG H_PSP_EUCL_MED HH_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG H_PSP_EUCL_MED HH_PSP_TCHE_AVG HH_PSP_TCHE_MED	Todos los algoritmos	Todos los algoritmos
PSP4	HH_PSP_EUCL_AVG H_PSP_EUCL_MED HH_PSP_TCHE_MED	HH_PSP_EUCL_AVG H_PSP_EUCL_MED HH_PSP_TCHE_MED	HH_PSP_EUCL_AVG HH_PSP_EUCL_MED H_PSP_TCHE_MED	HH_PSP_EUCL_AVG HH_PSP_EUCL_MED H_PSP_TCHE_MED	H_PSP_TCHE_MED	HH_PSP_EUCL_AVG H_PSP_EUCL_MED H_PSP_TCHE_MED	Todos los algoritmos	Todos los algoritmos
PSP5	H_PSP_TCHE_MED	HH_PSP_TCHE_MED	HH_PSP_TCHE_MED	H_PSP_TCHE_MED	H_PSP_TCHE_MED	H_PSP_TCHE_MED	Todos los algoritmos	Todos los algoritmos
PSP6	HH_PSP_EUCL_AVG H_PSP_TCHE_AVG	HH_PSP_EUCL_AVG H_PSP_EUCL_MED H_PSP_TCHE_AVG	HH_PSP_EUCL_AVG H_PSP_TCHE_AVG	HH_PSP_EUCL_AVG H_PSP_EUCL_MED H_PSP_TCHE_AVG HH_PSP_TCHE_MED	HH_PSP_EUCL_AVG H_PSP_EUCL_MED H_PSP_TCHE_AVG HH_PSP_TCHE_MED	HH_PSP_EUCL_AVG HH_PSP_TCHE_AVG	Todos los algoritmos	Todos los algoritmos

En resumen, HH_PSP_EUCL_AVG y HH_PSP_EUCL_MED son los algoritmos más consistentes en todas las métricas. HH_PSP_TCHE_MED también es competitivo, especialmente en problemas PSP1, PSP2 y PSP4. H_PSP_TCHE_MED domina PSP5, lo que sugiere que este problema se adapta mejor a estrategias basadas en Tchebycheff Mediana. PSP3 y PSP6 no presentan diferencias significativas entre varios algoritmos, lo que indica que múltiples estrategias pueden ser viables en estos casos. Las métricas xSz_out1 y zPx_out2 no diferencian entre algoritmos, lo que sugiere que podrían no ser adecuadas para evaluar el desempeño en PSP.

4.3.2.6 PSP para 5 Objetivos con Preferencias.

Las instancias de prueba de PSP son problemas de gran escala, lo cual añade complejidad a la búsqueda, pues como se menciona en esta sección estas instancias cuentan con preferencias de un DM. La característica de muchos objetivos está presente en ellas al ser de 5 objetivos.

La tabla 4.25 muestra los mejores algoritmos para instancias PSP de 5 objetivos. En cada celda se muestra un solo algoritmo (diferencia significativa) o varios algoritmos (sin diferencia significativa). Podemos observar en la tabla 4.25 lo siguiente:

- En PSP1 los algoritmos HH_PSP_EUCL_AVG y HH_PSP_EUCL_MED dominan en casi todas las métricas.
- En PSP2 una combinación de HH_PSP_EUCL_AVG, HH_PSP_EUCL_MED, HH_PSP_TCHE_AVG y H_PSP_TCHE_MED domina la mayoría de las métricas.
- En PSP3 una combinación de HH_PSP_EUCL_AVG, HH_PSP_EUCL_MED, HH_PSP_TCHE_AVG, H_PSP_TCHE_MED y HH_PSP_TCHE_MIN domina la mayoría de las métricas.
- Los algoritmos HH_PSP_TCHE_AVG y HH_PSP_TCHE_MED junto con H_PSP_EUCL_MED son los mejores algoritmos en casi todas las métricas en PSP4.
- En PSP5 los algoritmos HH_PSP_EUCL_AVG y HH_PSP_TCHE_AVG dominan casi todas las metricas.
- Similar a PSP3 y PSP4 en PSP6, existe predominio de HH_PSP_EUCL_AVG, HH_PSP_TCHE_AVG y H_PSP_TCHE_MED.
- En las métricas xSz_out1 y zPx_out2 todos los algoritmos tienen el mismo desempeño por lo cual estas métricas no sirven para discriminar entre los algoritmos.
- También se observa que los algoritmos de mejor desempeño están basados en preferencias, y que aquellos que no están basados en preferencias desaparecen.

Tabla 4.25. Mejores algoritmos por métrica de desempeño para el problema PSP con preferencia de un decisor para 5 objetivos.

Problema	avg_eucl	min_eucl	avg_tche	min_tche	med_eucl	med_tche	xSz_out1	zPx_out2
PSP1	HH_PSP_EUCL_AVG H_PSP_EUCL_MED	HH_PSP_EUCL_AVG HH_PSP_EUCL_MED	HH_PSP_EUCL_AVG HH_PSP_EUCL_MED	HH_PSP_EUCL_AVG H_PSP_EUCL_MED	HH_PSP_EUCL_AVG H_PSP_EUCL_MED	HH_PSP_EUCL_MED	Todos los algoritmos	Todos los algoritmos
PSP2	HH_PSP_EUCL_AVG HH_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG HH_PSP_TCHE_MED	HH_PSP_EUCL_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_MED H_PSP_TCHE_MED	HH_PSP_EUCL_AVG HH_PSP_TCHE_AVG	HH_PSP_EUCL_AVG H_PSP_TCHE_AVG HH_PSP_TCHE_MED	Todos los algoritmos	Todos los algoritmos
PSP3	HH_PSP_EUCL_AVG H_PSP_EUCL_MED H_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG HH_PSP_EUCL_MED HH_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG H_PSP_EUCL_MED HH_PSP_TCHE_AVG H_PSP_TCHE_MED H_PSP_TCHE_MIN	HH_PSP_EUCL_AVG HH_PSP_EUCL_MED HH_PSP_TCHE_AVG H_PSP_TCHE_MED HH_PSP_TCHE_MIN	H_PSP_EUCL_AVG HH_PSP_EUCL_MED H_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG H_PSP_EUCL_MED HH_PSP_TCHE_AVG HH_PSP_TCHE_MED H_PSP_TCHE_MIN	Todos los algoritmos	Todos los algoritmos
PSP4	H_PSP_EUCL_MED HH_PSP_TCHE_AVG H_PSP_TCHE_MED	H_PSP_EUCL_MED HH_PSP_TCHE_AVG H_PSP_TCHE_MED	H_PSP_EUCL_MED H_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_MED HH_PSP_TCHE_AVG HH_PSP_TCHE_MED	HH_PSP_EUCL_MED HH_PSP_TCHE_AVG H_PSP_TCHE_MED	H_PSP_EUCL_MED H_PSP_TCHE_AVG HH_PSP_TCHE_MED	Todos los algoritmos	Todos los algoritmos
PSP5	HH_PSP_EUCL_AVG H_PSP_TCHE_AVG	HH_PSP_EUCL_AVG H_PSP_EUCL_MED HH_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG	HH_PSP_EUCL_AVG HH_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG H_PSP_TCHE_AVG	HH_PSP_EUCL_AVG	Todos los algoritmos	Todos los algoritmos
PSP6	HH_PSP_EUCL_AVG HH_PSP_EUCL_MED H_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG HH_PSP_TCHE_MED	H_PSP_EUCL_AVG HH_PSP_EUCL_MED HH_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG HH_PSP_TCHE_MED	HH_PSP_EUCL_AVG HH_PSP_EUCL_MED HH_PSP_TCHE_AVG H_PSP_TCHE_MED	HH_PSP_EUCL_AVG HH_PSP_EUCL_MED HH_PSP_TCHE_AVG HH_PSP_TCHE_MED	Todos los algoritmos	Todos los algoritmos

Los algoritmos HH_PSP_EUCL_AVG y HH_PSP_EUCL_MED siguen siendo las opciones más robustas en PSP con 5 objetivos. Las estrategias basadas en Tchebycheff (HH_PSP_TCHE_AVG y HH_PSP_TCHE_MED) son más relevantes con 5 objetivos que con 3 objetivos. El algoritmo H_PSP_TCHE_MED es competitivo en problemas PSP2, PSP3, PSP4 y PSP5. Las métricas xSz_out1 y zPx_out2 no son discriminativas para evaluar el desempeño de los algoritmos. En general, PSP los algoritmos basados en preferencias tienen un buen desempeño al acercarse a la ROI, mientras que los que no están basados en preferencias como HH_PSP_TCHE_IGD, HH_PSP_TCHE_HV y HH_PSP_TCHE_PCArea sufren una pérdida de efectividad.

Capítulo 5

5 Conclusiones y Recomendaciones

En la presente investigación se desarrolló un framework de software para la construcción de hiperheurísticas, este está diseñado para facilitar el desarrollo, la implementación y la evaluación de técnicas avanzadas de optimización. Se estructura en torno a una arquitectura de clases clave como Hyperheuristic, LowLevelHeuristic, LowLevelHeuristicSet, y HHSelectionOperator, entre otras, las cuales colaboran para proporcionar un entorno flexible y potente para la solución de problemas de optimización complejos, que involucran muchos objetivos, gran escala, preferencias e imprecisión.

La clase Hyperheuristic es la encargada de generar nuevas hiperheurísticas y aplica mecanismos de selección que permiten la adaptación dinámica de las heurísticas de bajo nivel durante el proceso de optimización. Para la selección de las heurísticas de bajo nivel, se emplea el operador HHSelectionOperator, el cual se define de manera que el usuario pueda experimentar con diferentes modalidades de selección basadas en indicadores específicos de rendimiento. La clase LowLevelHeuristic permite definir y configurar diferentes heurísticas de bajo nivel (LLH) que son las encargadas de buscar nuevas soluciones a los problemas planteados. Por ejemplo, el framework proporciona algoritmos como MOSA/D, CO-MOSA/D, MOSA/D-O, y MOSA/D-O-II.

Esta arquitectura ha permitido el desarrollo de una gran cantidad de algoritmos entre LLHs e Hiperheurísticas: alrededor de 12 LLHs distintas, entre las que se cuentan MOSA/D (publicado en revista), CO-MOSA/D (en proceso de publicación), MOSA/D-O y MOSA/D-O-II (publicados en revista). Además de más de 30 hiperheurísticas, de las cuales 28 fueron del ultimo experimento y que tomaron un mes de desarrollo en el framework. En el capítulo 4 se describen las pruebas realizadas con el fin de probar la capacidad del framework de generar hiperheurísticas enfocadas en los problemas descritos anteriormente. Las pruebas incluyen la resolución de problemas de optimización tipo DTLZ, WFG y PSP. Estos problemas se utilizaron para evaluar la capacidad de los algoritmos de alcanzar el frente de Pareto, así como la eficiencia en alcanzar la región de interés del decisor en presencia de muchos objetivos, gran escala, preferencias e imprecisión. En este caso fueron utilizados 28 algoritmos hiperheurísticos, 44 instancias de prueba (14 DTLZ, 18 WFG y 12 PSP), y 30 ejecuciones por cada instancia de prueba.

5.1 Conclusiones Finales

Las conclusiones finales que se han generado dada la metodología y pruebas realizadas son las siguientes:

- Aproximación al Frente de Pareto: Uno de los principales objetivos de la investigación era evaluar la capacidad de los algoritmos desarrollados para aproximarse al frente de Pareto en problemas de optimización multi-objetivo, incluyendo la búsqueda en la región de interés del decisor (ROI). Los resultados de las pruebas, especialmente en los problemas DTLZ, WFG y PSP con diferentes números de objetivos y variables, muestran que los algoritmos HH_IGD y HH_PCArea logran un desempeño consistente en la aproximación al frente de Pareto. Cabe destacar que el problema PSP HH_PCArea tiene un predomino que lo muestra como un algoritmo fuerte. Estos algoritmos se destacaron reiteradamente en las métricas de hipervolumen, lo cual indica que tienen un buen rendimiento general en términos de convergencia y cobertura del espacio de soluciones. Esto sugiere que los objetivos de diseñar y probar algoritmos capaces de aproximarse al frente de Pareto se cumplieron adecuadamente.

- Impacto de la Escalabilidad: Otro objetivo era evaluar cómo los algoritmos responden a la escalabilidad del problema, en particular, cuando se incrementa el número de objetivos y variables de decisión. Los problemas DTLZ, WFG Y PSP con diferentes cantidades de objetivos y variables de decisión muestran resultados en los que, aunque algunos algoritmos presentan consistencia en los problemas más simples (como los de 3 o 5 objetivos), en problemas de mayor complejidad (10 objetivos o 1000 variables) se observa un mayor número de empates entre algoritmos. Sin embargo, también se demostró especialmente en PSP (sin preferencias y con preferencias) donde algoritmos como HH_PSP-PCArea y algoritmos basados en preferncias son lo suficientemente robustos como para mantener un rendimiento competitivo en condiciones de muchos objetivos y gran apoyando así la hipótesis de que los algoritmos diseñados podrían ser efectivos en condiciones de alta dimensionaldad.
- Optimización en la Región de Interés (ROI): Un aspecto clave de la hipótesis era si los algoritmos serían capaces de aproximarse efectivamente a la región de interés especificada por un decisor. Los resultados indican que las hiperheurísticas variantes basadas los operadores de selección basados en preferencias como HH_PSP_EUCL_AVG y HH_PSP_EUCL_MED aparecen frecuentemente entre los mejores algoritmos, especialmente en PSP en problemas con 500 y 1000 variables de decisión donde los algoritmos no basados en preferencias pierden efectividad. En el caso de DTLZ las estrategias como HH_DTLZ_PR en el caso de 5 y 10 objetivos muestra diferencia significativa frente a otras estrategias basadas en preferencias. Esto sugiere que estos algoritmos son efectivos para explorar la región de interés y evitar soluciones que no satisfacen las preferencias del decisor. Además, el hecho de que HH_IGD también destaque en métricas relacionadas con la aproximación a la ROI muestra que la minimización de la distancia generacional invertida es una estrategia útil para mantener la proximidad al frente de referencia y, por lo tanto, a la región de interés. En consecuencia, se puede concluir que los objetivos relacionados con la búsqueda focalizada en la ROI también fueron alcanzados.

- Hipótesis sobre el Rendimiento de los Algoritmos: La hipótesis principal de la investigación era que los algoritmos desarrollados serían efectivos tanto en la aproximación al frente de Pareto como en la búsqueda de la región de interés. Los resultados obtenidos de los problemas WFG y PSP indican que los algoritmos desarrollados muestran un rendimiento robusto y adecuado para ambos aspectos. Los empates observados en las métricas de evaluación y la falta de un claro ganador absoluto entre todos los algoritmos en ciertos casos, como en las métricas xSz_out1 y zPx_out2, da evidencia de que no hay una solución universal que funcione mejor para todos los tipos de problemas y métricas. Sin embargo, la presencia recurrente de algoritmos como HH_PCArea y HH_IGD, y algoritmos basados en preferencias entre los mejores sugiere que la hipótesis es verdadera: existen algoritmos que son efectivamente capaces de aproximarse tanto al frente de Pareto como a la región de interés.
- Los resultados obtenidos demuestran claramente que el framework propuesto facilita el desarrollo rápido y versátil de hiperheurísticas, tal como se planteó en la hipótesis inicial de la investigación. Durante un periodo de tan solo un mes, se lograron desarrollar alrededor de 28 hiperheurísticas diferentes, mientras que en los 3 años anteriores se crearon un número mucho menor de algoritmos, lo cual es evidencia contundente de la rapidez y eficiencia del framework en la generación de estrategias de optimización. Además, la publicación de tres heurísticas de bajo nivel (LLHs) creadas para el framework —una en un artículo en una revista JCR y dos más en otro artículo publicado en la revista SoftwareX, también JCR— refuerza la versatilidad del framework, indicando que los algoritmos generados no solo se crean de forma rápida, sino que también poseen la calidad necesaria para ser aceptados por la comunidad científica.

5.2 Trabajos Futuros

En esta sección se mencionan trabajos futuros que pueden ser tomados por la comunidad científica, así como por el propio autor:

- Ampliación del conjunto de LLHs: Desarrollar y agregar nuevas heurísticas de bajo nivel (LLHs) al framework para diversificar las técnicas de búsqueda y mejorar la capacidad de los algoritmos al abordar diferentes problemas de optimización.
- Exploración de nuevos problemas de optimización: Aplicar el framework en problemas de optimización adicionales, incluyendo problemas industriales y del mundo real que involucren un gran número de restricciones, con el fin de validar la capacidad del framework para manejar la complejidad inherente a estos desafíos.
- Evaluación de facilidades de uso en ambientes colaborativos: Realizar estudios que evalúen la facilidad de uso del framework en equipos de desarrollo colaborativo, verificando cómo el framework se adapta al trabajo en equipo y si proporciona mejoras en la productividad y eficiencia del equipo.
- Integración de preferencias del decisor: Profundizar en la integración de preferencias del decisor a lo largo de todo el proceso de optimización, y no solo en la fase de evaluación de soluciones, con el fin de mejorar la adaptabilidad de las soluciones generadas a los intereses específicos de los decisores.
- Incorporación de técnicas de aprendizaje automático: Explorar la incorporación de técnicas de aprendizaje automático, como redes neuronales o aprendizaje por refuerzo, para la selección adaptativa de LLHs durante la ejecución de las hiperheurísticas, con el fin de mejorar la adaptabilidad de los algoritmos en diferentes fases del proceso de optimización.
- Comparación extensiva con frameworks existentes: Realizar comparaciones más detalladas del rendimiento y la facilidad de uso del framework con otros frameworks existentes, para identificar ventajas competitivas y áreas de mejora respecto a herramientas similares.
- Publicación y difusión de nuevas hiperheurísticas: Continuar publicando nuevos resultados obtenidos mediante la aplicación del framework, no solo a problemas de prueba, sino también a problemas aplicados, con el fin de demostrar la efectividad y aplicabilidad del enfoque en contextos industriales y científicos.
- Evaluación del impacto del framework en la productividad científica: Desarrollar estudios que midan de manera cuantitativa el impacto del framework en la

productividad científica, evaluando aspectos como la cantidad y calidad de artículos científicos generados gracias a las soluciones desarrolladas con el uso del framework.

5.3 Publicaciones Derivadas del Presente Trabajo de Investigación

M. Vargas-Martinez, N. Rangel-Valdez, E. Fernández, G. Rivera, and L. Cruz-Reyes, **"Selective Pressure through Differential Evolution and Decomposition in Multi-Objective Simulated Annealing,"** in *Proceedings of MOL2NET'22, Conference on Molecular, Biomedical & Computational Sciences and Engineering, 8th ed. - MOL2NET: FROM MOLECULES TO NETWORKS*, **Dec. 2022**, p. 13871, doi: 10.3390/mol2net-08-13871.

M. Vargas-Martínez, N. Rangel-Valdez, E. Fernández, C. Gómez-Santillán, and M. L. Morales-Rodríguez, **"Performance Analysis of Multi-Objective Simulated Annealing Based on Decomposition,"** *Math. Comput. Appl.*, vol. 28, no. 2, p. 38, **Mar. 2023**, doi: 10.3390/mca28020038.

M. Vargas-Martínez, N. Rangel-Valdez, E. Fernández, C. Gómez-Santillán, G. Rivera, and F. Balderas, **"MOSA/D-O and MOSAD/D-O-II: Performance analysis of decomposition-based algorithms in many objective problems,"** SoftwareX, vol. 25, no. November 2023, p. 101610, **Feb. 2024**, doi: 10.1016/j.softx.2023.101610.

Glosario

2. **Optimización Multi-objetivo (MOP)** – Proceso de encontrar soluciones que optimizan múltiples objetivos en conflicto simultáneamente.
3. **Optimización de Muchos Objetivos (MaOPs)** – Extensión de la optimización multi-objetivo cuando hay más de tres objetivos.
4. **Optimización de Gran Escala (LSGO)** – Optimización de problemas con cientos o miles de variables de decisión.
5. **Hiperheurística** – Método de optimización que selecciona o genera heurísticas para resolver problemas computacionales.
6. **Metaheurística** – Estrategia de búsqueda para optimización que guía heurísticas en la exploración de soluciones.
7. **Heurística** – Método basado en reglas simples para encontrar soluciones aproximadas de forma eficiente.
8. **Descomposición** – Técnica para dividir un problema grande en subproblemas más manejables.
9. **Coevolución** – Método en el que múltiples poblaciones evolucionan simultáneamente e interactúan entre sí.
10. **Dominancia de Pareto** – Relación entre soluciones en la que una solución domina a otra si es mejor en al menos un objetivo sin empeorar en otros.
11. **Frente de Pareto** – Conjunto de soluciones óptimas en una optimización multi-objetivo donde ninguna solución domina a otra.
12. **Región de Interés (ROI)** – Subconjunto del frente de Pareto que satisface mejor las preferencias de un decisor.
13. **Intervalos** – Representación matemática de valores inciertos o imprecisos mediante rangos en lugar de números exactos.
14. **Toma de Decisiones Multicriterio (MCDM)** – Proceso de selección de la mejor alternativa considerando múltiples criterios.
15. **Recocido Simulado (SA)** – Algoritmo de optimización

inspirado en el enfriamiento gradual de metales.

16. **Algoritmo Evolutivo (EA)** – Técnica de optimización basada en la evolución natural y la selección de soluciones.
17. **MOEA/D** – Algoritmo evolutivo multi-objetivo basado en descomposición.
18. **Preferencias del Decisor** – Criterios subjetivos utilizados por un decisor para seleccionar soluciones óptimas.
19. **Outranking** – Método de comparación de soluciones en problemas multicriterio basado en relaciones de preferencia.
20. **Hipervolumen (HV)** – Indicador de desempeño que mide el volumen cubierto por un conjunto de soluciones en el espacio de objetivos.
21. **PCArea** – Nueva métrica de desempeño basada en coordenadas paralelas, utilizada para evaluar la calidad de soluciones en optimización multi-objetivo.
22. **Selección de Proyectos (PSP)** – Problema de optimización en el que se elige la mejor combinación de proyectos bajo restricciones de recursos y objetivos.

Bibliografía

[1] L. Li, I. Yevseyeva, V. Basto-Fernandes, H. Trautmann, N. Jing, and M. Emmerich, “An Ontology of Preference-Based Multiobjective Metaheuristics,” no. August 2018, 2016,

[2] Z. M. Gu and G. G. Wang, “Improving NSGA-III algorithms with information feedback models for large-scale many-objective optimization,” *Futur. Gener. Comput. Syst.*, vol. 107, pp. 49–69, 2020, doi: 10.1016/j.future.2020.01.048.

[3] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, “Test Problems for Large-Scale Multiobjective and Many-Objective Optimization,” *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4108–4121, 2017, doi: 10.1109/TCYB.2016.2600577.

[4] S. Chand and M. Wagner, “Evolutionary Many-Objective Optimization : A Quick-Start Guide,” *Surv. Oper. Res. Manag. Sci.*, vol. 20, no. 2, pp. 1–28, 2015.

[5] S. Bechikh, M. Elarbi, and L. Ben Said, *Many-objective optimization using evolutionary algorithms: A survey*, vol. 20. 2017.

[6] B. Li, J. Li, K. Tang, and X. Yao, “Many-objective evolutionary algorithms: A survey,” ACM Comput. Surv., vol. 48, no. 1, 2015, doi: 10.1145/2792984.

[7] E. Fernandez, E. Lopez, S. Bernal, C. A. Coello Coello, and J. Navarro, “Evolutionary multiobjective optimization using an outranking-based dominance generalization,” *Comput. Oper. Res.*, vol. 37, no. 2, pp. 390–395, 2010, doi: 10.1016/j.cor.2009.06.004.

[8] L. Li, I. Yevseyeva, V. Basto-Fernandes, H. Trautmann, N. Jing, and M. Emmerich, “An Ontology of Preference-Based Multiobjective Metaheuristics,” no. August 2018, 2016, [Online]. Available: http://arxiv.org/abs/1609.08082.

[9] A. Ramírez, J. R. Romero, and S. Ventura, “A survey of many-objective optimisation in search-based software engineering,” *J. Syst. Softw.*, vol. 149, pp. 382–395, 2019, doi: 10.1016/j.jss.2018.12.015.

[10] X. Yang, J. Zou, S. Yang, J. Zheng, and Y. Liu, “A Fuzzy Decision Variables Framework for Large-Scale Multiobjective Optimization,” IEEE Transactions on Evolutionary Computation, vol. 27, no. 3, pp. 445–459, Jun. 2023, doi: 10.1109/TEVC.2021.3118593.

[11] B. Cao *et al.*, “Distributed Parallel Particle Swarm Optimization for Multi-Objective and Many-Objective Large-Scale Optimization,” *IEEE Access*, vol. 5, pp. 8214–8221, 2017, doi: 10.1109/ACCESS.2017.2702561.

[12] J. Del Ser *et al.*, “Bio-inspired computation: Where we stand and what’s next,” *Swarm Evol. Comput.*, vol. 48, no. December 2018, pp. 220–250, 2019, doi: 10.1016/j.swevo.2019.04.008.

[13] T. Kunakote *et al.*, “Comparative Performance of Twelve Metaheuristics for Wind Farm Layout Optimisation,” *Arch. Comput. Methods Eng.*, vol. 29, no. 1, pp. 717–730, 2022, doi: 10.1007/s11831-021-09586-7.

[14] W. Li, E. Özcan, and R. John, “Multi-objective evolutionary algorithms and hyper-heuristics for wind farm layout optimisation,” *Renew. Energy*, vol. 105, pp. 473–482, 2017, doi: 10.1016/j.renene.2016.12.022.

[15] C. A. C. Coello, “Handling preferences in evolutionary multiobjective optimization: A survey,” *Proc. 2000 Congr. Evol. Comput. CEC 2000*, vol. 1, pp. 30–37, 2000, doi: 10.1109/CEC.2000.870272.

[16] H. Wang, M. Olhofer, and Y. Jin, “A mini-review on preference modeling and articulation in multi-objective optimization: current status and challenges,” *Complex Intell. Syst.*, vol. 3, no. 4, pp. 233–245, 2017, doi: 10.1007/s40747-017-0053-9.

[17] Y. J. and B. Sendhoff, “INCORPORATION OF FUZZY PREFERENCES INTO EVOLUTIONARY MULTIOBJECTIVE OPTIMIZATION,” *GECCO 2002*, vol. 1, no. November, pp. 26–30, 2002.

[18] F. Balderas, E. Fernandez, C. Gomez-Santillan, N. Rangel-Valdez, and L. Cruz, *An Interval-Based Approach for Evolutionary Multi-Objective Optimization of Project Portfolios*, vol. 18, no. 4. 2019.

[19] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Metaheuristics in large-scale global continues optimization: A survey," *Inf. Sci. (Ny).*, vol. 295, pp. 407–428, 2015, doi: 10.1016/j.ins.2014.10.042.

[20] L. Xiaodong, T. Ke, S. P. N., Y. Zhenyu, and W. Thomas, "Benchmark Functions for the CEC'2013 Special Session and Competition on Large-Scale Global Optimization," *Tech. report, Univ. Sci. Technol. China*, no. 1, pp. 1–21, 2010, [Online]. Available: http://goanna.cs.rmit.edu.au/~xiaodong/cec13-lsgo/competition/cec2013-lsgo-benchmark-tech-report.pdf.

[21] C. Qian, K. Tang, and Z. H. Zhou, "Selection hyper-heuristics can provably be helpful in evolutionary multi-objective optimization," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9921 LNCS, pp. 835–846, 2016, doi: 10.1007/978-3-319-45823-6_78.

[22] G. O. Edmund K. Burke, Matthew Hyde, Graham Kendall and and J. R. W. Ender ̇Ozcan, "Metaheurisic Hybrids," in *International Series in Operations Research & Management Science*, vol. 146, 2010, p. 648.

[23] P. J. Fleming, R. C. Purshouse, and R. J. Lygoe, "Many-objective optimization: An engineering design perspective," *Lect. Notes Comput. Sci.*, vol. 3410, pp. 14–32, 2005, doi: 10.1007/978-3-540-31880-4_2.

[24] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary Many-Objective Optimization: A Short Review," pp. 2424–2431, 2008.

[25] O. Schütze, A. Lara, and C. A. C. Coello, “On the influence of the number of objectives on the hardness of a multiobjective optimization problem,” *IEEE Trans. Evol. Comput.*, vol. 15, no. 4, pp. 444–455, 2011, doi: 10.1109/TEVC.2010.2064321.

[26] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, “A Decision Variable Clustering-Based Evolutionary Algorithm for Large-scale,” *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 97--112, 2016.

[27] A. Fernández Carazo, T. Gómez Núñez, F. M. Guerrero Casas, and R. Caballero Fernández, “Evaluación y clasificación de las técnicas utilizadas por las organizaciones, en las últimas décadas, para seleccionar proyectos,” *REVISTA DE MÉTODOS CUANTITATIVOS PARA LA ECONOMÍA Y LA EMPRESA*, no. 5, pp. 67–115, 2008.

[28] M. Melián Batista, J. Moreno Vega, and J. Moreno Pérez, “Metaheuristicas: Una visión global,” *Intel. Artif. Rev. Iberoam. Intel. Artif.*, vol. 7, no. 19, pp. 7–28, 2003.

[29] F. Glover, “Future paths for integer programming and links to artificial intelligence,” *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, 1986.

[30] J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke, “Recent advances in selection hyper-heuristics,” *European Journal of Operational Research*, vol. 285, no. 2. Elsevier B.V., pp. 405–428, Sep. 01, 2020. doi: 10.1016/j.ejor.2019.07.073.

[31] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing,” *Sci. 220(4598)*, vol. 220, no. 4598, pp. 671–680, 1983.

[32] V. Černý, “Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm,” *J. Optim. Theory Appl.*, vol. 45, no. 1, pp. 41–51, 1985, doi: 10.1007/BF00940812.

[33] J. W. Escobar and R. Linfati, “a Meta-Heuristic Algorithm Based on the Simulated Annealing With Granular Search Space for the Capacitated Location Routing Problem,” *Rev.*

Ing. Univ. Medellín, vol. 11, no. 21, pp. 139–150, 2012, [Online]. Available: http://www.scielo.org.co/pdf/rium/v11n21/v11n21a12.pdf.

[34] R. Storn and K. Price, "Stochastic optimization, nonlinear optimization, global optimization, genetic algorithm, evolution strategy," *J. Glob. Optim.*, vol. 11, pp. 241–359, 1997, doi: 10.1023/A:1008202821328.

[35] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, 2011, doi: 10.1109/TEVC.2010.2059031.

[36] I. M. Ali, D. Essam, and K. Kasmarik, "Novel binary differential evolution algorithm for knapsack problems," *Inf. Sci. (Ny).*, vol. 542, pp. 177–194, 2021, doi: 10.1016/j.ins.2020.07.013.

[37] Q. Zhang y H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, 2007, doi: 10.1109/TEVC.2007.892759.

[38] H. Ishibuchi, K. Doi, and Y. Nojima, "Reference point specification in MOEA/D for multi-objective and many-objective problems," *2016 IEEE Int. Conf. Syst. Man, Cybern. SMC 2016 - Conf. Proc.*, pp. 4015–4020, 2017, doi: 10.1109/SMC.2016.7844861.

[39] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 866 LNCS, pp. 249–257, 1994, doi: 10.1007/3-540-58484-6_269.

[40] Yang, Z., Tang, K., \& Yao, X. (2008). Multilevel cooperative coevolution for large scale optimization. 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), 1663–1670. https://doi.org/10.1109/CEC.2008.4631014

[41] L. M. Antonio and C. A. C. Coello, “Use of cooperative coevolution for solving large scale multiobjective optimization problems,” 2013 IEEE Congr. Evol. Comput. CEC 2013, no. 2, pp. 2758–2765, 2013.

[42] J. Fülöp, “Introduction to decision making,” in Digital Decision Making, London: Springer London, 2007, pp. 13–32. doi: 10.1007/978-1-84628-673-5_2.

[43] C. A. C. Coello, “Handling preferences in evolutionary multiobjective optimization: A survey,” *Proc. 2000 Congr. Evol. Comput. CEC 2000*, vol. 1, pp. 30–37, 2000, doi: 10.1109/CEC.2000.870272.

[44] H. Wang, M. Olhofer, and Y. Jin, “A mini-review on preference modeling and articulation in multi-objective optimization: current status and challenges,” *Complex Intell. Syst.*, vol. 3, no. 4, pp. 233–245, Dec. 2017, doi: 10.1007/s40747-017-0053-9.

[45] B. Roy and P. Vincke, “Multicriteria analysis: survey and new directions,” *Eur. J. Oper. Res.*, vol. 8, no. 3, pp. 207–218, 1981, doi: 10.1016/0377-2217(81)90168-5.

[46] B. Roy, “The outranking approach and the foundations of electre methods,” *Theory Decis.*, vol. 31, no. 1, pp. 49–73, 1991, doi: 10.1007/BF00134132.

[47] A. V. Cristina Gómez, Jesús Ladevesa, Purushotham Muniganti Karina Gibert, “USE AND EVALUATI ON OF ELECTRE TRI 2.0a,” 2007. doi: 10.1093/swra/23.1.14.

[48] E. Fernandez, E. Lopez, F. Lopez, and C. A. Coello Coello, “Increasing selective pressure towards the best compromise in evolutionary multiobjective optimization: The extended NOSGA method,” *Inf. Sci. (Ny).*, vol. 181, no. 1, pp. 44–56, 2011, doi: 10.1016/j.ins.2010.09.007.

[49] B. Roy, *Multicriteria Methodology for Decision Aiding*. 1996.

[50] J. R. Figueira, V. Mousseau, and B. Roy, “ELECTRE methods,” in *International Series in Operations Research and Management Science*, vol. 233, 2005, pp. 155–185.

[51] J. P. Brans, P. Vincke, and B. Mareschal, “How to select and how to rank projects : The PROMETHEE method,” *Eur. J. Oper. Res.*, vol. 24, pp. 228–238, 1986.

[52] P. Smets, “IMPERFECT INFORMATION: IMPRECISION AND UNCERTAINTY,” in *Uncertainty Management in Information Systems*, vol. 17, no. 8, 1997, pp. 225–254.

[53] R.E. Moore. Methods and Applications of Interval Analysis. Siam, Philadelphia, 1979.

[54] J. Swan, E. Özcan, and G. Kendall, “Hyperion – A Recursive Hyper-Heuristic Framework,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 6683 LNCS, Springer, Berlin, Heidelberg, 2011, pp. 616–630. doi: 10.1007/978-3-642-25566-3_48.

[55] G. Ochoa et al., “HyFlex: A benchmark framework for cross-domain heuristic search,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7245 LNCS, 2012, pp. 136–147. doi: 10.1007/978-3-642-29124-1_12.

[56] J. H. Drake, E. Ozcan, and E. K. Burke, “A modified choice function hyper-heuristic controlling unary and binary operators,” in 2015 IEEE Congress on Evolutionary Computation (CEC), IEEE, May 2015, pp. 3389–3396. doi: 10.1109/CEC.2015.7257315.

[57] W. Van Onsem and B. Demoen, “ParHyFlex: A framework for parallel hyper-heuristics,” Belgian/Netherlands Artif. Intell. Conf., pp. 231–238, 2013.

[58] N. Pillay and D. Beckedahl, “EvoHyp - a Java toolkit for evolutionary algorithm hyper-heuristics,” in 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, Jun. 2017, pp. 2706–2713. doi: 10.1109/CEC.2017.7969636.

[59] E. Urra, C. Cubillos, D. Cabrera-Paniagua, and R. Mellado, “hMod : A software framework for assembling highly detailed heuristics algorithms,” Software: Practice and Experience, vol. 49, no. 6, pp. 971–994, Jun. 2019, doi: 10.1002/spe.2690.

[60] H. K. Cora, H. T. Uyar, and A. Ş. Etaner-Uyar, “HH-DSL: a domain specific language for selection hyper-heuristics,” in Proceedings of the 15th annual conference companion on Genetic and evolutionary computation, New York, NY, USA: ACM, Jul. 2013, pp. 1317–1324. doi: 10.1145/2464576.2482711.

[61] A. Kheiri and E. Özcan, “An iterated multi-stage selection hyper-heuristic,” European Journal of Operational Research, vol. 250, no. 1, pp. 77–90, Apr. 2016, doi: 10.1016/j.ejor.2015.09.003.

[62] J. M. Cruz-Duarte, J. C. Ortiz-Bayliss, and I. Amaya, “MatHH: A Matlab-based Hyper-Heuristic framework,” SoftwareX, vol. 18, p. 101047, Jun. 2022, doi: 10.1016/j.softx.2022.101047.

[63] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, “Scalable Test Problems for Evolutionary Multiobjective Optimization,” *Evol. Multiobjective Optim.*, no. 1990, pp. 105–145, 2005, doi: 10.1007/1-84628-137-7_6.

[64] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, “HYPER-HEURISTICS: AN EMERGING DIRECTION IN MODERN SEARCH TECHNOLOGY.”

[65] S. Huband, L. Barone, L. While, and P. Hingston, “A Scalable Multi-objective Test Problem Toolkit,” in *Lecture Notes in Computer Science*, vol. 3410, 2005, pp. 280–295. doi: 10.1007/978-3-540-31880-4_20.

[66] M. Maashi, E. Özcan, y G. Kendall, “A multi-objective hyper-heuristic based on choice function,” *Expert Syst. Appl.*, vol. 41, no. 9, pp. 4475–4493, 2014, doi: 10.1016/j.eswa.2013.12.050.

[67] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, “Evolutionary Many-Objective Optimization: A Short Review,” pp. 2424–2431, 2008.

[68] W. Qiu, J. Zhu, G. Wu, M. Fan, and P. N. Suganthan, “Evolutionary many-Objective algorithm based on fractional dominance relation and improved objective space

decomposition strategy," *Swarm Evol. Comput.*, vol. 60, p. 100776, 2021, doi: 10.1016/j.swevo.2020.100776.

[69] M. Vargas-Martínez, N. Rangel-Valdez, E. Fernández, C. Gómez-Santillán, and M. L. Morales-Rodríguez, "Performance Analysis of Multi-Objective Simulated Annealing Based on Decomposition," *Math. Comput. Appl.*, vol. 28, no. 2, p. 38, Mar. 2023, doi: 10.3390/mca28020038.

[70] P. Engrand, "Multi-objective optimization approach based on simulated annealing and its application to nuclear fuel management," *Int. Conf. Nucl. Eng. Proceedings, ICONE*, p. 477, 1997.

[71] M. Vargas-Martínez, N. Rangel-Valdez, E. Fernández, C. Gómez-Santillán, G. Rivera, and F. Balderas, "MOSA/D-O and MOSAD/D-O-II: Performance analysis of decomposition-based algorithms in many objective problems," *SoftwareX*, vol. 25, no. November 2023, p. 101610, Feb. 2024, doi: 10.1016/j.softx.2023.101610.

[72] J. G. Falcón-Cardona y C. A. Coello Coello, "A new indicator-based many-objective ant colony optimizer for continuous search spaces," Swarm Intell., vol. 11, no. 1, pp. 71–100, 2017, doi: 10.1007/s11721-017-0133-x.

Anexos

Anexo A. Experimentos

A.1 Experimento MOSA/D: MOSA/D-CGO y MOSA/D-DE

Tabla A.1. Condiciones experimentales para la validación de MOSA/D-CGO versus MOSA/D-DE.

Algoritmo	MOSA/D-CGO, MOSA/D-DE
Problema estándar	DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5, DTLZ6, DTLZ7
Número de objetivos	3, 5 y 10
Tamaño de la población	100
Evaluaciones por ejecución	100100
Ejecuciones por variante	30
Indicadores	HV e IGD

A.1.1 Resumen de los Resultados

Se realizó una prueba de Wilcoxon de dos colas con una confianza del 0.05% para este experimento. Los símbolos "↑", "↓" y "--" indican que hay diferencia significativa en favor de MOSA/D-DE, diferencia significativa en contra de MOSA/D-DE y no hay diferencia significativa entre ambos algoritmos respectivamente. Los resultados con fondo gris representan los mejores resultados para cada problema. Para el cálculo del HV se utiliza los valores máximos por objetivo para cada problema (Véase Tabla A.10). La tabla A.2 muestra los resultados del experimento en términos de HV con resultados positivos en favor de MOSA/D-DE en la mayoría de los casos. MOSA/D-DE es notable en los problemas DTLZ1, DTLZ2, DTLZ3, DTLZ5 y DTLZ 7 mientras que MOSA/D-CGO lo es en DTLZ4 y DTLZ6.

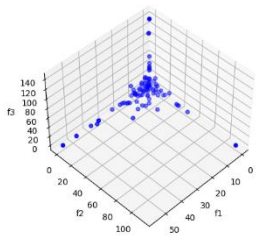
Tabla A.2. Comparaciones de MOSA/D-CGO y MOSA/D-DE dada la media y desviación estándar (entre paréntesis) del Hipervolumen.

PROBLEMA	M	MOSA/D-CGO	MOSA/D-DE
DTLZ1	3	1.411396E+07(7.856170E+01)	↑1.411407E+07(2.077398E+01)
	5	3.013240E+11(7.546308E+04)	↑3.013241E+11(1.109078E+04)
	10	1.667917E+16(2.419205E+13)	↑1.668377E+16(3.215349E+08)
DTLZ2	3	1.508330E+01(2.781792E-02)	↑1.517692E+01(7.637901E-03)
	5	7.361979E+01(7.759884E-01)	↑7.387519E+01(3.847436E-02)
	10	2.600539E+01(4.958777E-01)	↓2.569999E+01(1.980545E-01)
DTLZ3	3	8.163884E+08(6.516296E+04)	↑8.165865E+08(4.513141E+02)
	5	1.689445E+14(8.081423E+12)	↑1.704707E+14(9.718498E+07)
	10	1.699675E+24(4.226203E+22)	↑1.707961E+24(8.685232E+18)
DTLZ4	3	7.958441E+00(1.283030E-02)	↑8.002655E+00(1.583281E-02)
	5	4.522733E+01(3.629265E-02)	↓4.509505E+01(8.641444E-02)
	10	2.327196E+03(3.100255E+00)	↓2.298456E+03(1.139568E+01)
DTLZ5	3	9.944000E+00(1.078512E-02)	↑9.990017E+00(2.501031E-02)
	5	7.652259E+01(3.121247E-01)	--7.657619E+01(1.983684E-01)
	10	3.412050E+00(2.043577E-02)	↑3.428953E+00(8.094812E-03)
DTLZ6	3	9.009791E+02(1.712226E+01)	↓7.219506E+02(8.185913E+00)
	5	1.053329E+05(4.977173E+03)	↓7.647568E+04(1.856713E+03)
	10	4.179011E+08(6.692261E+06)	↓2.876860E+08(9.403331E+06)
DTLZ7	3	1.536594E+01(6.362204E-01)	↑1.699378E+01(2.933636E-01)
	5	2.219751E+01(1.606481E+00)	--2.263389E+01(9.949435E-01)
	10	0.000000E+00(0.000000E+00)	↑5.649215E-06(1.130646E-05)

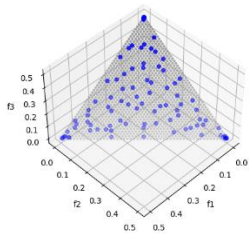
En la tabla A.3 se observan los resultados del cálculo del IGD del experimento. Los resultados son positivos en favor de MOSA/D-DE en la mayoría de los casos. MOSA/D-DE es notable en los problemas DTLZ1, DTLZ3, DTLZ4 y DTLZ 7 mientras que MOSA/D-CGO lo es en DTLZ2, DTLZ5 y DTLZ6.

Tabla A.3. Comparaciones de MOSA/D-CGO y MOSA/D-DE dada la media y desviación estándar (entre paréntesis) del IGD.

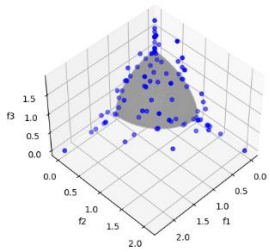
PROBLEMA	M	MOSA/D_CGO	MOSA/D-DE
DTLZ1	3	1.698462E+00(5.231658E-01)	↑2.567411E-02(3.988398E-04)
	5	1.848446E+00(7.371114E-01)	↑1.362515E-01(8.662087E-03)
	10	2.605653E+00(8.858904E-01)	↑2.746888E-01(2.055289E-02)
DTLZ2	3	1.338869E-01(1.050833E-02)	↑6.539183E-02(8.887102E-04)
	5	3.879495E-01(2.383225E-02)	↓4.315983E-01(1.731880E-02)
	10	8.086104E-01(3.491915E-02)	↓8.740702E-01(3.886488E-02)
DTLZ3	3	3.943222E+01(1.066523E+01)	↑8.927530E-02(8.582940E-03)
	5	3.527925E+01(1.256126E+01)	↑2.402407E+00(1.052387E+00)
	10	3.857988E+01(8.728455E+00)	↑1.452693E+01(3.864857E+00)
DTLZ4	3	2.193973E-01(2.708875E-02)	↑7.370770E-02(2.095534E-03)
	5	4.264948E-01(2.618424E-02)	↑3.453491E-01(1.259533E-02)
	10	7.485737E-01(2.873640E-02)	--7.565568E-01(2.740685E-02)
DTLZ5	3	3.692489E-02(4.482185E-03)	↑8.796775E-03(4.662547E-04)
	5	2.173594E-01(2.151767E-02)	↓3.198228E-01(3.323426E-03)
	10	8.274774E-01(3.498018E-02)	↓9.278413E-01(8.544034E-03)
DTLZ6	3	2.360560E+00(4.875849E-01)	↓5.755759E+00(1.295398E-01)
	5	3.233184E+00(5.565340E-01)	↓7.249152E+00(1.815938E-01)
	10	4.311238E+00(3.579400E-01)	↓7.249090E+00(1.507697E-01)
DTLZ7	3	2.640245E+00(5.499560E-01)	↑6.763103E-01(4.596304E-02)
	5	5.536790E+00(6.953019E-01)	↑1.562072E+00(2.362084E-01)
	10	5.536790E+00(6.953019E-01)	↑1.562072E+00(2.362084E-01)



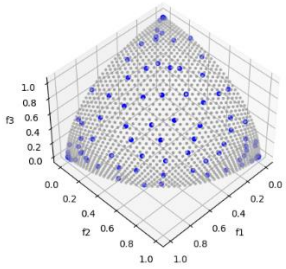
DTLZ1 - MOSA/D-CGO



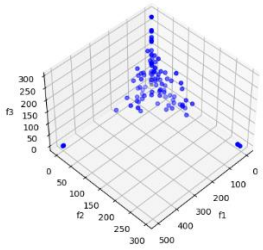
DTLZ1 - MOSA/D-DE



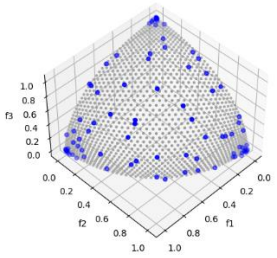
DTLZ2 - MOSA/D-CGO



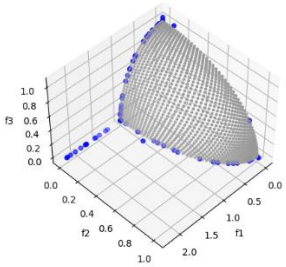
DTLZ2 - MOSA/D-DE



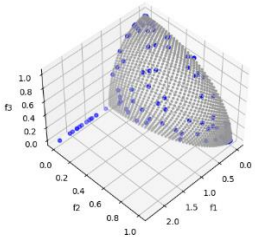
DTLZ3 - MOSA/D-CGO



DTLZ3 - MOSA/D-DE

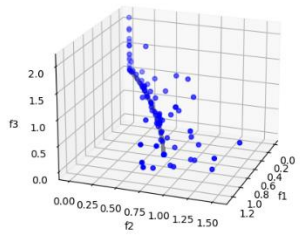


DTLZ4 - MOSA/D-CGO

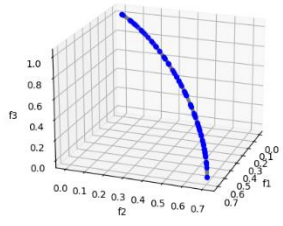


DTLZ4 - MOSA/D-DE

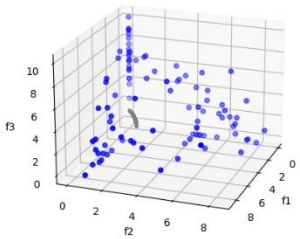
Figura A.1. Gráficas de las poblaciones aproximadas contra el Frente de Pareto de los problemas DTLZ1, DTLZ2, DTLZ3 y DTLZ4 con 3 objetivos.



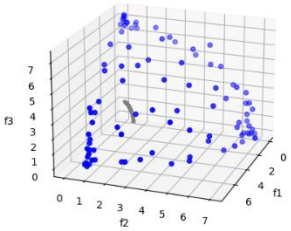
DTLZ5 - MOSA/D-CGO



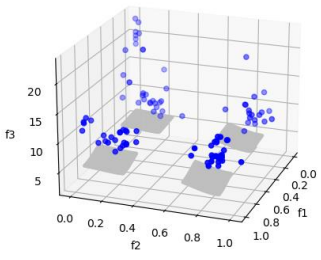
DTLZ5 - MOSA/D-DE



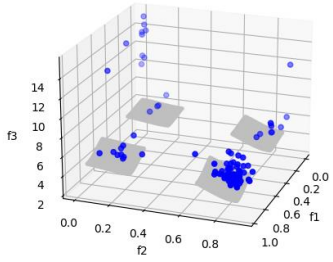
DTLZ6 - MOSA/D-CGO



DTLZ6 - MOSA/D-DE



DTLZ7 - MOSA/D-CGO



DTLZ7 - MOSA/D-DE

Figura A.2. Gráficas de las poblaciones aproximadas contra el Frente de Pareto de los problemas DTLZ5, DTLZ6 y DTLZ7 con 3 objetivos.

En las figuras A.1 y A.2 se observan las mejores aproximaciones (basado en HV) al frente de Pareto de ambos algoritmos para los 7 problemas con 3 objetivos. MOSA/D-DE muestra un mejor desempeño que MOSA/D-CGO en DTLZ1, DTLZ2, DTLZ3, DTLZ4 y DTLZ5. Pero en DTLZ6 y DTLZ7 ambos tienen problemas para aproximarse al frente de Pareto (PF). Cada problema DTLZ tiene características especiales, por ejemplo, DTLZ1 y DTLZ3 tiene una gran cantidad de frentes locales de Pareto. En la figura A.1 se observa que MOSA/D-CGO tiene problemas para escapar de esos frentes. En el caso de DTLZ6 este tiene la particularidad de tener un PF en forma de curva. Ambos algoritmos sufren perdida de

convergencia en DTLZ6 con ligera ventaja para MOSA/D-CGO según la tabla A.2 y A.3. DTLZ7 por su parte ofrece el reto de tener un PF separado en regiones. Ambos algoritmos logran hacer grupos de soluciones hacia estas regiones, pero sufren de perdida de convergencia con una ligera ventaja de MOSA/D-DE según la tabla A.2 y A.3.

A.2 Experimento del Algoritmo Hiperheurístico MaOLS-HH-UP

En el capítulo 3 se describió la arquitectura del HH-Framework para la generación de Hiperheurísticas. Este ya tiene su primera implementación funcional. Por lo cual se diseñó el algoritmo MaOLS-HH-UP basado en el HH-Framework (descrito en la sección 3.2). Entonces para probar la funcionalidad del HH-Framework se implementó MaOLS-HH-UP. Una vez implementado se procedió a diseñar un experimento para observar su desempeño. En la tabla A.3 se muestran las condiciones del experimento donde se compara el algoritmo Hiperheurístico MaOLS-HH-UP y sus LLHs (MOSA/D-CGO y MOSA/D-DE) utilizando los problemas estándar DTTLZ (DTLZ1 a DTLZ7).

Tabla A.3. Condiciones experimentales para la validación de MaOLS-HH-UP

Algoritmo	MaOLS-HH-UP	MOSA/D-DE	MOSA/D-CGO
Heurísticas de bajo nivel (LLHs)	MOSA/D-DE, MOSA/D-CGO	No aplica	No aplica
No. de evaluaciones α	1000	No aplica	No aplica
No. de evaluaciones β	8000	No aplica	No aplica
Evaluaciones totales	100000		
Problema estándar	DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5, DTLZ6 y DTLZ7		
Número de objetivos	3, 5 y 10		
Tamaño de la población	100		
Ejecuciones por variante	30		
Indicadores	Hipervolumen (HV)		
Prueba estadística	Wilcoxon de dos colas, con un nivel de significancia del 5%		

5.3.1 Resumen de los Resultados

Se realizó una prueba de Wilcoxon de dos colas con una confianza del 0.05% para este experimento. Los símbolos "↑", "↓" y "--" indican que hay diferencia significativa en favor del algoritmo X sobre MaOLS-HH-UP, hay diferencia significativa en favor del MaOLS-HH-UP sobre el algoritmo X y no hay diferencia significativa entre ambos algoritmos respectivamente. Los resultados resaltados en color azul representan los mejores resultados globales, los resaltados en verde los mejores segundos y los resaltados en blanco los terceros para cada problema. . Para el cálculo del HV se utilizó la tabla de referencia proporcionada en el trabajo de Falcón-Cardona y Coello [72] (Véase tabla B.2).

Tabla A.4. Comparaciones de MaOLS-HH-UP con sus LLHs (MOSA/D-CGO y MOSA/D-DE) dada la media y desviación estándar (entre paréntesis) del Hipervolumen.

PROBLEMA	M	MaOLS-HH-UP	MOSA/D-DE	MOSA/D-CGO
DTLZ1	3	9.674129E-01(9.345314E-04)	↓9.669436E-01(6.699539E-04)	↓4.067315E-02(9.454652E-02)
	5	9.907649E-01(1.377662E-03)	--9.900387E-01(1.354862E-03)	↓6.568351E-03(2.041957E-02)
	10	9.644969E-01(4.062294E-02)	--9.737273E-01(4.383044E-03)	↓5.927621E-04(3.192121E-03)
DTLZ2	3	7.372309E+00(4.347433E-03)	↓7.363545E+00(6.175829E-03)	--7.373764E+00(8.057621E-03)
	5	3.126759E+01(4.569430E-02)	--3.125473E+01(3.517449E-02)	↑3.140164E+01(7.939782E-02)
	10	9.671418E+02(1.166160E+01)	↓9.553585E+02(1.039615E+01)	↑9.998694E+02(7.507674E+00)
DTLZ3	3	3.423347E+02(4.565722E-02)	↓3.422957E+02(4.362511E-02)	↓0.000000E+00(0.000000E+00)
	5	1.534983E+04(4.044869E+03)	↑1.680608E+04(7.798920E-02)	↓0.000000E+00(0.000000E+00)
	10	3.975743E+07(7.681018E+07)	↑2.823115E+08(7.199193E+04)	↓0.000000E+00(0.000000E+00)
DTLZ4	3	7.366120E+00(2.083426E-02)	↓7.361308E+00(1.053105E-02)	↓7.308177E+00(2.452472E-02)
	5	3.135000E+01(1.018412E-01)	--3.136912E+01(6.722750E-02)	↓3.129514E+01(8.709710E-02)
	10	1.011942E+03(9.096954E+00)	↓9.980188E+02(8.883283E+00)	↓9.818580E+02(2.312441E+01)
DTLZ5	3	5.983794E+01(1.392400E-02)	↓5.977835E+01(5.256280E-02)	↓5.978122E+01(9.329715E-02)
	5	9.528937E+02(1.991997E+00)	↓9.525813E+02(5.920149E-01)	↓9.468219E+02(3.571125E+00)
	10	9.606504E+05(7.662636E+03)	--9.615494E+05(1.567743E+03)	↑9.683289E+05(2.746768E+03)
DTLZ6	3	1.311044E+03(8.902515E-01)	↑1.312884E+03(7.089666E-01)	↓1.303067E+03(2.843756E+00)
	5	1.583765E+05(2.039310E+02)	↓1.537122E+05(5.808567E+02)	↓1.572284E+05(4.616764E+02)
	10	2.525560E+10(7.873795E+07)	↓2.277242E+10(3.283114E+08)	↓2.490287E+10(1.424205E+08)
DTLZ7	3	1.613258E+01(3.281796E-02)	--1.613191E+01(3.146374E-02)	↓1.365189E+01(7.497044E-01)
	5	1.201377E+01(1.089791E-01)	↑1.211803E+01(8.413816E-02)	↓5.363649E+00(7.703187E-01)
	10	7.207963E-01(1.633323E-01)	--7.741105E-01(1.857967E-01)	↓1.326185E-02(1.036532E-02)

Tabla A.5. Comparaciones de MaOLS-HH-UP con sus LLHs (MOSA/D-CGO y MOSA/D-DE) dada la media y desviación estándar (entre paréntesis) de IGD.

PROBLEMA	M	MaOLS-HH-UP	MOSA/D-DE	MOSAD/-CGO
DTLZ1	3	2.681142E-02(7.042141E-04)	↑2.621375E-02(4.027662E-04)	↓1.341020E+00(5.978901E-01)
	5	1.381509E-01(9.603491E-03)	↓1.465996E-01(8.958837E-03)	↓2.005655E+00(8.219922E-01)
	10	2.934714E-01(2.317264E-02)	--2.857720E-01(2.246790E-02)	↓3.282472E+00(1.035805E+00)
DTLZ2	3	6.815032E-02(9.049263E-04)	↑6.617358E-02(8.161511E-04)	↓7.394387E-02(2.766878E-03)
	5	4.378970E-01(1.187116E-02)	--4.425269E-01(1.350893E-02)	↑3.275153E-01(1.335729E-02)
	10	8.310793E-01(3.997759E-02)	--8.486789E-01(4.507075E-02)	↑7.594846E-01(3.109494E-02)
DTLZ3	3	6.822445E-02(1.717869E-03)	↑6.706030E-02(1.339223E-03)	↓2.996838E+01(5.951387E+00)
	5	1.502059E+00(3.493634E+00)	--4.611076E-01(1.894087E-02)	↓3.395139E+01(7.797159E+00)
	10	1.463451E+01(1.042342E+01)	↑8.756291E-01(7.212207E-02)	↓3.790234E+01(9.021997E+00)
DTLZ4	3	8.456204E-02(3.809943E-02)	↑6.975498E-02(1.925206E-03)	↓1.502864E-01(1.812882E-02)
	5	3.335917E-01(3.171575E-02)	↑3.109226E-01(1.355738E-02)	↓4.204751E-01(2.128065E-02)
	10	7.748494E-01(4.960594E-02)	↑7.066725E-01(2.802164E-02)	↓9.152894E-01(5.359206E-02)
DTLZ5	3	9.485766E-03(5.327348E-04)	↑8.629234E-03(4.788150E-04)	↓1.105848E-02(1.377695E-03)
	5	3.280400E-01(9.354291E-03)	--3.295649E-01(8.960257E-04)	↑2.985578E-01(2.885526E-02)
	10	9.349564E-01(2.112533E-02)	↓9.425905E-01(5.449832E-03)	↑8.971702E-01(2.911648E-02)
DTLZ6	3	4.712562E-01(3.186590E-02)	↑3.348389E-01(1.130428E-02)	↓7.561742E-01(1.157655E-01)
	5	5.973972E-01(6.268972E-02)	↓2.615454E+00(1.541354E-01)	↓1.152206E+00(2.391291E-01)
	10	1.234885E+00(1.650059E-01)	↓4.011483E+00(3.090405E-01)	↓2.297590E+00(3.338552E-01)
DTLZ7	3	2.030995E-01(1.470382E-02)	--1.986315E-01(1.694858E-02)	↓5.230282E-01(1.150968E-01)
	5	8.432066E-01(7.415795E-02)	↑8.006570E-01(6.646019E-02)	↓1.314676E+00(2.939690E-01)
	10	1.315179E+00(9.350786E-02)	--1.274734E+00(7.318006E-02)	↑2.998833E+00(7.601165E-01)

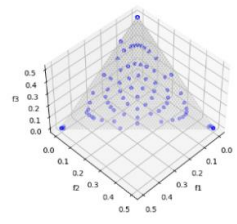
Los resultados mostrados por HV e IGD (Tablas A.4 y A.5) resultan ser prometedores para algoritmo MaOLS-HH-UP. La tabla A.4 muestra los resultados en términos de HV donde el hiperheurístico tiene 10 de 21 de los mejores resultados globales, MOSA/D-DE 7 de 21 y MOSA/D-CGO 4 de 21. En las comparaciones entre MaOLS-HH-UP y MOSA/D-CGO hay diferencia significativa a favor de MaOLS-HH-UP en 17 de 21 instancias, diferencia significativa a favor de MOSA/D-CGO en 3 de 21 instancias, y sin diferencia significativa en 1 de 21 instancias.

En las comparaciones entre MaOLS-HH-UP y MOSA/D-DE hay diferencia significativa a favor de MaOLS-HH-UP en 4 de 21 instancias, diferencia significativa a favor de MOSA/D-

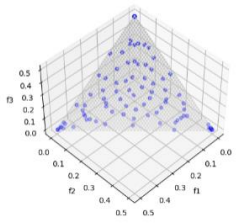
DE en 10 de 21 instancias, y sin diferencia significativa en 7 de 21 instancias. Por lo cual el hiperheurístico tiene una ventaja comparado con sus LLHs en términos de HV.

La tabla A.5 muestra los resultados en términos de IGD donde MOSA/D-DE tiene 14 de 21 de los mejores resultados globales, MOSA/D-CGO 4 de 21 y MaOLS-HH-UP 3 de 21. En las comparaciones entre MaOLS-HH-UP y MOSA/D-CGO hay diferencia significativa a favor de MaOLS-HH-UP en 16 de 21 instancias, diferencia significativa a favor de MOSA/D-CGO en 5 de 21 instancias, y sin diferencia significativa en 0 de 21 instancias. En las comparaciones entre MaOLS-HH-UP y MOSA/D-DE hay diferencia significativa a favor de MaOLS-HH-UP en 4 de 21 instancias, diferencia significativa a favor de MOSA/D-DE en 4 de 21 instancias, y sin diferencia significativa en 7 de 21 instancias. Por lo cual MOSA/D-DE tiene una ventaja comparado IGD. Cabe señalar que el hiperheurístico tiene los mejores segundos resultados globales en IGD por lo cual estos resultados son prometedores.

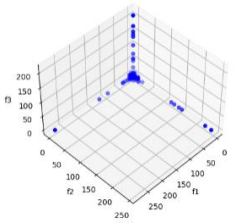
Los resultados de la tabla A.4 indican que al algoritmo MaOLS-HH-UP tiene una mejor cobertura del espacio de soluciones no dominadas que MOSA/D-DE y MOSA/D-CGO. Mientras que los resultados de la tabla A.5 muestran que MaOLS-HH-UP tiene un desempeño prometedor en la proximidad de las soluciones generadas y el conjunto de referencia superando MOSA/D-CGO, pero siendo superado por MOSA/D-DE. Una de las fortalezas con las que cuenta MOSA/D-DE es que puede aproximar su conjunto de soluciones al Frente de Pareto mejor que MOSA/D-CGO y esa característica puede ser aprovechada por el Hiperheurístico en una forma más eficiente.



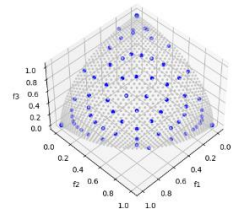
DTLZ1 – MaOLS-HH-UP



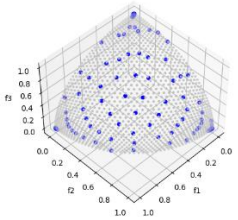
DTLZ1 - MOSA/D-DE



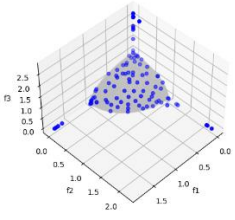
DTLZ1 - MOSA/D-CGO



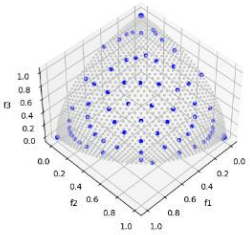
DTLZ2 – MaOLS-HH-UP



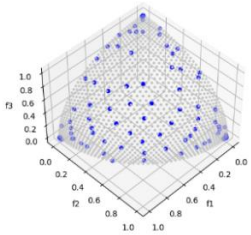
DTLZ2 - MOSA/D-DE



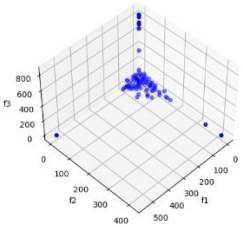
DTLZ2 - MOSA/D-CGO



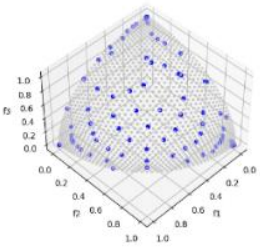
DTLZ3 – MaOLS-HH-UP



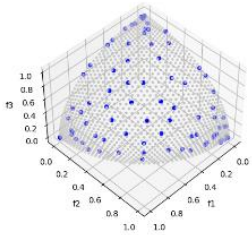
DTLZ3 - MOSA/D-DE



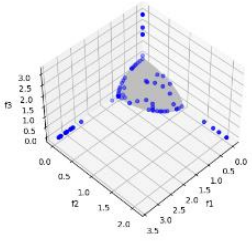
DTLZ3 - MOSA/D-CGO



DTLZ4 – MaOLS-HH-UP

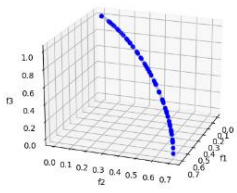


DTLZ4 - MOSA/D-DE

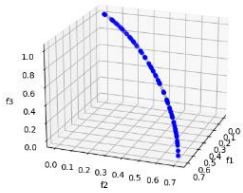


DTLZ4 - MOSA/D-CGO

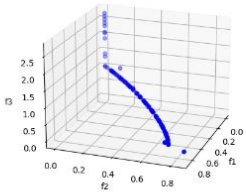
Figure A.3. Gráficas de las poblaciones aproximadas contra el Frente de Pareto de los problemas DTLZ1, DTLZ2, DTLZ3 y DTLZ4 con 3 objetivos.



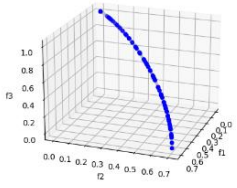
DTLZ5 – MaOLS-HH-UP



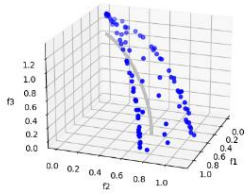
DTLZ5 - MOSA/D-DE



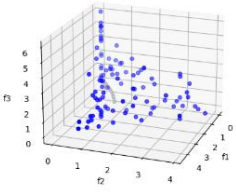
DTLZ5 - MOSA/D-CGO



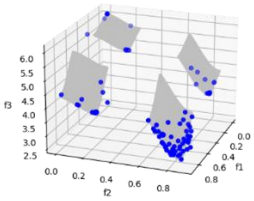
DTLZ6 – MaOLS-HH-UP



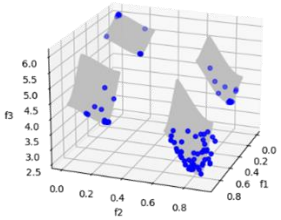
DTLZ6 - MOSA/D-DE



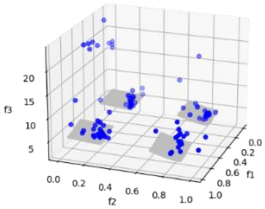
DTLZ6 - MOSA/D-CGO



DTLZ7 – MaOLS-HH-UP



DTLZ7 - MOSA/D-DE



DTLZ7 - MOSA/D-CGO

Figure A.4. Gráficas de las poblaciones aproximadas contra el Frente de Pareto de los problemas DTLZ5, DTLZ6 y DTLZ7 con 3 objetivos.

A.3 Experimentación de MOSA/D-O y MOSA/D-O-II

Para evaluar el desempeño de los algoritmos basados en preferencias MOSA/D-O y MOSA/D-O-II en su habilidad para resolver problemas de muchos objetivos. Se realizó un diseño experimental donde también se incluye al algoritmo FDEA-II que en el estado del arte tiene buenos resultados con problemas de muchos objetivos. MOSA/D es incluido también por su habilidad prometedora en problemas de muchos objetivos.

Tabla A.6. Diseño experimental para MOSA/D-O y MOSA/D-O-II.

Parámetro	FDEA-II	MOSA/D	MOSA/D-O	MOSA/D-O-II
a	0.10	-		
b	0.40	-		
α	-	0.98		
Ti	-	1		
Tf	-	0.0000001		
Factor de escala F	-	0.5		
Tasa de cruza Cr	-	0.2		
λ	-		0.6, 0.6	
MFE	100000			
N	100			
Corridas por instancia	15			
Problemas	DTLZ1, DTLZ2,...,DTLZ7, WFG1, WFG2,...,WFG9			
Número de objetivos	5, 10			

Se compararon FDEA-II, MOSA/D, MOSA/D-O y MOSA/D-O-II en cuanto a su capacidad para aproximarse a la ROI. Para medir esta capacidad se consideró el indicador PR(P, z) descrito en la tabla A.6. PR(P, z) mide la proporción de soluciones en las que la relación xSz es verdadera, donde $x \in P$ y z es el mejor compromiso. Para observar si existe una diferencia significativa entre el conjunto de datos de cada algoritmo, se empleó la prueba estadística no paramétrica de Friedman y el análisis post-hoc de Nemenyi con un nivel de significancia de 0,01. En la Tabla 4.10, cada sub-tabla contiene un problema y la clasificación de desempeño para 5 y 10 objetivos. Si uno o más algoritmos tienen buen desempeño, ocuparán el primer lugar. FDEA-II, MOSA/D, MOSA/D-O y MOSA/D-O-II están representados por F, M, M1 y M2, respectivamente.

Tabla A.7. Resultados de las pruebas con DTLZ and WFG

		PR(P,z)	
PROBLEM	RANK	5	10
DTLZ5	1	M1	F, M1, M2
	1.5	-	-
	2	M2	-
	2.5		-
	3	M	-
	3.5	-	-
	4	F	M2

		PR(P,z)	
PROBLEM	RANK	5	10
DTLZ6	1	M1	F
	1.5	-	-
	2	M2	M
	2.5		-
	3	M	-
	3.5	-	M1, M2
	4	F	

		PR(P,z)	
PROBLEM	RANK	5	10
DTLZ7	1	M	F
	1.5	-	-
	2	-	-
	2.5	-	-
	3	F, M1, M2	M, M1, M2
	3.5	-	-
	4	-	-

		PR(P,z)	
PROBLEM	RANK	5	10
WFG1	1	-	F, M, M2
	1.5	-	-
	2	-	-
	2.5	F, M, M1, M2	-
	3	-	-
	3.5	-	-
	4	-	M1

		PR(P,z)	
PROBLEM	RANK	5	10
WFG2	1	M1	F, M, M2
	1.5	-	-
	2	-	-
	2.5	M, M2	-
	3	-	-
	3.5	-	-
	4	F	M1

		PR(P,z)	
PROBLEM	RANK	5	10
WFG3	1	M1	F
	1.5	-	-
	2	-	M
	2.5	M, M2	-
	3	-	M2
	3.5	-	-
	4	F	M1

		PR(P,z)	
PROBLEM	RANK	5	10
WFG4	1	-	-
	1.5	-	-
	2	-	-
	2.5	F,M,M1,M2	F,M,M1,M2
	3	-	-
	3.5	-	-
	4	-	-

		PR(P,z)	
PROBLEM	RANK	5	10
WFG5	1	-	-
	1.5	-	-
	2	-	-
	2.5	F,M,M1,M2	F,M,M1,M2
	3	-	-
	3.5	-	-
	4	-	-

		PR(P,z)	
PROBLEM	RANK	5	10
WFG6	1	-	-
	1.5	-	-
	2	-	-
	2.5	F,M,M1,M2	F,M,M1,M2
	3	-	-
	3.5	-	-
	4	-	-

		PR(P,z)	
PROBLEM	RANK	5	10
WFG7	1	-	F
	1.5	F, M1	-
	2	-	M1
	2.5	-	-
	3	-	-
	3.5	M, M2	M, M2
	4	-	-

		PR(P,z)	
PROBLEM	RANK	5	10
WFG8	1	M1	F, M1, M2
	1.5	-	-
	2	-	-
	2.5	M, M2	-
	3	-	-
	3.5	-	-
	4	F	M

		PR(P,z)	
PROBLEM	RANK	5	10
WFG9	1	-	F
	1.5	F, M1	-
	2	-	-
	2.5	-	M1, M2
	3	-	-
	3.5	M, M2	-
	4	-	M

		PR(P,z)	
PROBLEM	RANK	5	10
DTLZ1	1	M1	F, M, M2
	1.5	-	-
	2	-	-
	2.5	M, M2	-
	3	-	-
	3.5	-	-
	4	F	M1

		PR(P,z)	
PROBLEM	RANK	5	10
DTLZ2	1	M1	F, M, M2
	1.5	-	-
	2	-	-
	2.5	M, M2	-
	3	-	-
	3.5	-	-
	4	F	M1

		PR(P,z)	
PROBLEM	RANK	5	10
DTLZ3	1	M1	F
	1.5	-	-
	2	-	M
	2.5	M, M2	-
	3	-	M2
	3.5	-	-
	4	F	M1

		PR(P,z)	
PROBLEM	RANK	5	10
DTLZ4	1	F, M1, M2	F, M, M2
	1.5	-	-
	2	-	-
	2.5	-	-
	3	-	-
	3.5	-	-
	4	M	M1

A.3.1 Análisis de Resultados: Cinco Objetivos

Los resultados para los cinco objetivos de la tabla A.7 se resumen en la Tabla A.8. Ningún algoritmo tiene dominio absoluto sobre los demás en todos los casos. Pero M1 (MOSA/D-O) supera al resto de algoritmos en 8 de 16 casos. En comparaciones de pares, M1 supera a F en 8 de 16 casos, M1 supera a M en 11 de 16 casos y M1 supera a M2 en 10 de 16 casos.

Tabla A.8. Resumen de los resultados de cinco objetivos

	PR(P,z)	
PROBLEMA	Mejores Algoritmos	Peores Algoritmos
DTLZ1	M1	M,M2,F
DTLZ2	M1	M,M2,F
DTLZ3	M1	M,M2,F
DTLZ4	F, M1, M2	M
DTLZ5	M1	M,M2,F
DTLZ6	M1	M,M2,F
DTLZ7	M	F,M1,M2
WFG1	F, M, M1, M2	-
WFG2	M1	M,M2,F
WFG3	M1	M,M2,F
WFG4	F, M, M1, M2	-
WFG5	F, M, M1, M2	-
WFG6	F, M, M1, M2	-
WFG7	F, M1	M,M2
WFG8	M1	M,M2,F
WFG9	F, M1	M,M2

A.3.2 Análisis de Resultados: Diez Objetivos

Los resultados para diez objetivos de la tabla A.7 se resumen en la Tabla A.9. Ningún algoritmo tiene dominio absoluto sobre los demás en todos los casos. Pero F (FDEA-II) supera al resto de algoritmos en 6 de 16 casos. En comparaciones de pares, F supera a M1 en 11 de 16 casos, F supera a M en 8 de 16 casos y F supera a M2 en 6 de 16 casos. Es importante mencionar que M2 supera a M en 2 de 16 casos y ambos tienen el mismo rendimiento en 8 de 16 casos.

Tabla A.9. Resumen de los resultados de diez objetivos

	PR(P,z)	
PROBLEMA	Mejores Algoritmos	Peores Algoritmos
DTLZ1	F, M, M2	M1
DTLZ2	F, M, M2	M1
DTLZ3	F	M, M1, M2
DTLZ4	F, M, M2	M1
DTLZ5	F, M1, M2	M
DTLZ6	F	M, M1, M2
DTLZ7	F	M, M1, M2
WFG1	F, M, M2	M1
WFG2	F, M, M2	M1
WFG3	F	M, M1, M2
WFG4	F, M, M1, M2	-
WFG5	F, M, M1, M2	-
WFG6	F, M, M1, M2	-
WFG7	F	M, M1, M2
WFG8	F, M1, M2	M
WFG9	F	M, M1, M2

Los resultados mostrados en la tabla A.8 y A.9 muestran que la incorporación de preferencias (outranking) en el marco MOSA/D produce algoritmos que superan a MOSA/D en cinco y diez objetivos para los problemas de referencia DTLZ y WFG en su habilidad para aproximarse a la ROI. Y puede superar a algoritmos del estado del arte de última generación como FDEA-II en cinco objetivos para los problemas de referencia DTLZ y WFG.

Anexo B. Tablas

En esta sección se encuentran tablas de referencia o de datos utilizadas por los algortimos o el análisis de los mismos.

B.1 Tablas y Datos de Prueba MOSA/D-DE y MOSA/D-CGO

La tabla B.1 muestra los puntos de referencia utilizados para el cálculo de hiperbolicen utilizados en el experimento A.1.

Tabla B.1. Puntos de referencia para el cálculo de Hipervolumen.

Test Problem	3 objectives	5 objectives	10 objectives	
DTLZ1	216.790132039933, 245.572126619941, 265.114937485812	80.8714391297344, 227.104643547038, 235.570177240396, 201.976698417331, 344.81892944792	4.06142015762104, 28.0355439457454, 48.6777918069853, 133.939834807773, 222.454058357031,	2.25156894368627, 18.0131641679773, 55.3772497318585, 139.713524060566, 321.944734395031
DTLZ2	2.38622940286774, 2.47201723363377, 2.6806935648516	2.05896735099775, 2.07725455911758, 2.30808795675219, 2.57449333535658, 2.9380095369211	0.6594305076443, 0.840342066735919, 1.54612890881046, 1.8764060443055, 2.29490194623458,	0.648350703222635, 1.24805308667332, 1.70777551383235, 1.97914511371776, 2.65985264422613
DTLZ3	822.509995384644, 799.103543751443, 1242.39097615921	689.016436482976, 547.613912205914, 500.989149866526, 827.81074606278, 1089.39809110161	49.243814823296, 133.64045805898, 217.750852407648, 565.932643419409, 829.140836249926,	78.5147608980575, 150.770000538526, 340.827375270017, 495.059846407958, 1271.6885408782,
DTLZ4	2.58950883612183, 1.65645180809823, 2.02042410538188	2.66379101299125, 2.13852567972279, 2.04131611330391, 1.99090557186032, 1.98125761314518,	2.64822143939513, 2.2491760776445,2.083066369613, 2.1065144928458, 1.99635654481651, 2.01705020385728,	2.20030288801471, 2.09386422171755, 2.24556199410806, 2.14178980883573
DTLZ5	2.06970298570428, 2.11909090597952, 2.74503917772623	1.31770044511346, 1.68921445015483, 3.31646087578869, 3.41224588178889, 3.49994217680484	0.291869064968389, 0.525290602300518, 0.751594578115702, 2.68788397225164, 3.41214316344024,	0.49338116154924, 0.611762301869089, 1.10935039999348, 3.29311486089528, 3.49975337563819
DTLZ6	9.71802048756533, 9.50035002572186, 10.5676062054648	9.56330574133902, 9.59158225806973, 10.477256602242, 10.9414071660156, 10.923652672406,	3.6503932212918, 5.62792594695437, 7.97964101213103, 9.89622372940959, 10.9073924572044,	3.29696637156103, 7.16542587886431, 9.77412868397095, 10.6364664206205, 10.9367989501483
DTLZ7	1.0, 0.999999999998507, 24.2793251715135	1.0, 0.999999994782373, 0.999999999986268, 0.999999999997808, 42.0189690066051,	1.0, 0.999999999990235, 0.999999998676653, 1.0,0.999999997160675,	0.999999991586879, 1.0, 0.99999999981814, 21.0

B.2 Puntos de Referencia Utilizados en el Trabajo de Falcon-Cardona y Coello

Tabla B.2. Puntos de referencia para el cálculo de Hipervolumen usados en el trabajo de Falcón-Cardona y Coello [72]

Problema	Punto de referencia
DTLZ1	(1, 1, 1, …)
DTLZ2, DTLZ4	(2, 2, 2, …)
DTLZ3	(7, 7, 7, …)
DTLZ5	(4, 4, 4, …)
DTLZ6	(11, 11, 11, …)
DTLZ7	(1, 1, 1, …, 21)

B.3 Lista de Variantes de los Algoritmos: HH-DTLZ, HH-WFG y HH-PSP

Tabla B.3. Se muestran las nueve variantes del hiperheurístico HH_DTLZ.

Algoritmos derivados para DTLZ	**Métrica que integra el método de selección**
HH_DTLZ_EUCL-AVG	EUCL-AVG()
HH_DTLZ_EUCL-MIN	EUCL-MIN()
HH_DTLZ_EUCL-MED	EUCL-MED()
HH_DTLZ_TCHE-AVG	TCHE-AVG()
HH_DTLZ_TCHE-MIN	TCHE-MIN()
HH_DTLZ_TCHE -MED	TCHE-MED()
HH_DTLZ_PR	PR()
HH_DTLZ_IGD	IGD()
HH_DTLZ_PCArea	PCArea()

Tabla B.4. Se muestran las nueve variantes del hiperheurístico HH_WFG

Algoritmos derivados para WFG	Métrica que integra el método de selección
HH_WFG_EUCL-AVG	EUCL-AVG()
HH_ WFG _EUCL-MIN	EUCL-MIN()
HH_ WFG _EUCL-MED	EUCL-MED()
HH_ WFG _TCHE-AVG	TCHE-AVG()
HH_ WFG _TCHE-MIN	TCHE-MIN()
HH_ WFG _TCHE -MED	TCHE-MED()
HH_ WFG _PR	PR()
HH_ WFG _IGD	IGD()
HH_ WFG _PCArea	PCArea()

Tabla B.5. Se muestran las diez variantes del hiperheurístico HH_PSP.

Algoritmos derivados para PSP	Métrica que integra el método de selección
HH_PSP_EUCL-AVG	EUCL-AVG()
HH_PSP_EUCL-MIN	EUCL-MIN()
HH_PSP_EUCL-MED	EUCL-MED()
HH_PSP_TCHE-AVG	TCHE-AVG()
HH_PSP_TCHE-MIN	TCHE-MIN()
HH_PSP_TCHE -MED	TCHE-MED()
HH_PSP_PR	PR()
HH_PSP_HV	HV()
HH_PSP_IGD	IGD()
HH_PSP_PCArea	PCArea()