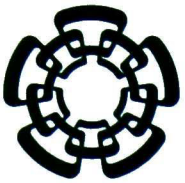
UT-T00086-551

Don-- 5015



Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Laboratorio de Tecnologías de Información,
CINVESTAV-Tamaulipas

Estudio sobre esquemas de hibridación para el uso simultáneo de diferentes formulaciones de un problema de optimización multi-objetivo

Tesis que presenta:

Auraham Sinhué Camacho García

Para obtener el grado de:

**Maestro en Ciencias
en Computación**

Director de la Tesis:
Dr. Gregorio Toscano-Pulido

Cd. Victoria, Tamaulipas, México

Diciembre, 2014

223817-2001

La tesis presentada por Auraham Sinhué Camacho García fue aprobada por:

_____

_____

Dr. Luis Gerardo de la Fraga

_____

Dr. Ricardo Landa Becerra

_____

Dr. Gregorio Toscano-Pulido, Director

Cd. Victoria, Tamaulipas, México, 9 de Diciembre de 2014

A la pandilla.

Agradecimientos

- Agradezco a mi familia por brindarme su apoyo desde casa cada semana. Sin ellos habría perdido la cordura rápidamente.
- Le doy las gracias a mis maestros en CINVESTAV-Tamaulipas por acercarme a la investigación, en especial al Dr. Gregorio Toscano-Pulido por su activa asesoría durante el año de tesis. Asimismo, al Dr. Ricardo Landa Becerra y al Dr. Luis Gerardo de la Fraga por los comentarios realizados durante la etapa de revisión.
- Le agradezco al personal administrativo de CINVESTAV-Tamaulipas por todas las facilidades que me brindaron a lo largo de mi estadía y por el soporte económico al término de la misma.
- Le agradezco a CONACyT por brindarme el apoyo económico necesario para concluir mi investigación en CINVESTAV-Tamaulipas.

Índice General

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

Resumen

Estudio sobre esquemas de hibridación para el uso simultáneo de diferentes formulaciones de un problema de optimización multi-objetivo

por

Auraham Sinhué Camacho García

Laboratorio de Tecnologías de Información, CINVESTAV-Tamaulipas
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2014
Dr. Gregorio Toscano-Pulido, Director

Un problema de optimización multi-objetivo puede ser transformado mediante diferentes técnicas, tales como escalarización y multiobjetivización, con el fin de generar una formulación alternativa. Existe un espacio de búsqueda asociado a cada formulación alternativa. De esta manera, es posible explorar diferentes espacios al emplear diferentes representaciones del mismo problema. Se ha encontrado en la literatura de optimización combinatoria que el desempeño de un algoritmo evolutivo se puede mejorar al utilizar probabilísticamente diferentes formulaciones, es decir, por medio de un esquema de selección probabilística. A partir de esta idea, en esta tesis se proponen dos nuevos esquemas de hibridación aplicados a optimización multi-objetivo. Nuestro primer enfoque está basado en un modelo dinámico de islas capaz de utilizar simultáneamente diferentes formulaciones de un mismo problema. Por otro lado, nuestro segundo enfoque está basado en aprendizaje por refuerzo y es capaz de utilizar diferentes formulaciones mediante un mecanismo de recompensa. La principal diferencia entre ambos enfoques es el tamaño de la población afectada por una formulación determinada. De esta manera, exploramos un nivel de granularidad distinto por cada propuesta. Estos tres esquemas (esquema de selección probabilística y nuestras dos propuestas) han sido evaluados mediante diferentes problemas de optimización multi-objetivo. A partir de resultados experimentales, se concluye que el uso adecuado de diferentes formulaciones puede mejorar el desempeño de un algoritmo evolutivo en diferentes problemas de prueba.

Abstract

Study of hybrid schemes for the simultaneous use of different formulations of a multi-objective optimization problem

by

Auraham Sinhué Camacho García

Information Technology Laboratory, CINVESTAV-Tamaulipas

Research Center for Advanced Study from the National Polytechnic Institute, 2014

Dr. Gregorio Toscano-Pulido, Advisor

A multi-objective optimization problem can be transformed through different techniques, such as scalarization or multiobjectivization, in order to generate an alternative formulation. There is a search space associated to each alternative formulation. This way, it is possible to explore different landscapes through several formulations of the same problem. It has been found in combinatorial optimization literature that the performance of an evolutionary algorithm can be improved using probabilistically different formulations, that is, by means of a probabilistic selection scheme. Following this idea, we propose two new hybrid schemes applied to multi-objective optimization. Our first approach is based on a dynamic island model which is able to use different formulations of the same problem simultaneously. On other hand, our second approach is based on reinforcement learning and it is able to use different formulations by a reward mechanism. The main difference between both approaches is the size of the population affected by a specific formulation. Thus, we explore a distinct level of granularity with each proposal. These three schemes (probabilistic selection scheme and our two approaches) have been evaluated through several multi-objective optimization problems. From our experimental results, we conclude that the proper use of different formulations can improve the performance of an evolutionary algorithm in different test problems.

Nomenclatura

DTLZ	Deb-Thiele-Leumman-Zitzler test suite
EA	Evolutionary Algorithm
EA+RL	Evolutionary Algorithm guided by Reinforcement Learning
EC	Evolutionary Computation
ES	Evolutionary Strategy
EP	Evolutionary Programming
DE	Differential Evolution
DIM	Dynamic Island Model
GA	Genetic Algorithm
MOEA	Multi-objective Evolutionary Algorithm
MOP	Multi-objective Optimization Problem
NSGA-II	Non-dominated Sorting Genetic Algorithm
OP	Operation Research
RL	Reinforcement Learning
ZDT	Zitzler-Deb-Thiele test suite

1

Introducción

1.1 Definición del problema y motivación

La optimización es el acto de encontrar la mejor solución bajo ciertas circunstancias [41]. La calidad de una solución se define a partir de una función conocida como *función objetivo* o simplemente *objetivo*. En la práctica, este objetivo podría representar el costo de fabricación de un artículo, la eficiencia de un proceso, la confiabilidad de un producto, entre otros más.

Un *problema de optimización mono-objetivo* involucra una única función objetivo y generalmente contiene una única solución óptima. Por otra parte, un *problema de optimización multi-objetivo* involucra varios objetivos de manera simultánea, generalmente en conflicto entre sí. En este último caso, usualmente no existe una única solución óptima sino un conjunto de soluciones conocido como *conjunto de óptimos de Pareto* [11].

La mayoría de los problemas de optimización del mundo real son multi-objetivo [13]. Diferentes métodos exactos han sido desarrollados para abordar este tipo de problemas. Sin embargo, dichos

métodos suelen requerir tiempos de cómputo excesivos en la práctica. Por otro lado, los métodos de computación evolutiva han alcanzado un gran éxito en resolver problemas de optimización multi-objetivo al proveer buenas soluciones en tiempos de cómputo razonables [44]. La *computación evolutiva* es un término genérico usado para referirse a métodos estocásticos de búsqueda inspirados en la evolución natural. Comprende una variedad de métodos, tales como algoritmos genéticos, programación evolutiva, evolución diferencial y estrategias evolutivas, todos ellos conocidos como *algoritmos evolutivos*. En general, los algoritmos evolutivos se componen de una población de individuos (posibles soluciones) y de un conjunto de operadores de variación y selección.

Un problema de optimización multi-objetivo, de aquí en adelante conocido como *problema original* o *formulación original*, se define como un vector de m funciones objetivo, denotado por $\boldsymbol{f} = [f_1, \ldots, f_m]^T$ A partir de una transformación, tal como escalarización o multiobjetivización, es posible definir una *formulación alternativa* del problema original, denotada por h. De este modo, se pueden utilizar diferentes representaciones del mismo problema de optimización, las cuales podrían ayudar a mejorar la calidad de las soluciones encontradas por un algoritmo evolutivo al modificar el espacio de búsqueda [32]. Trabajo previo ha demostrado experimentalmente esta posibilidad al emplear probabilísticamente diferentes formulaciones durante el proceso de optimización [24, 27], es decir, mediante un esquema de selección aleatoria. Sin embargo, aún existe un ámbito que no se ha explorado con profundidad: la selección adaptativa de la formulación más adecuada en cada etapa del proceso de búsqueda.

El problema abordado en esta tesis se define de la siguiente manera. Dado un problema de optimización multi-objetivo, $\boldsymbol{f}$, y un conjunto de formulaciones de $\boldsymbol{f}$, denotado por $\boldsymbol{h} = \{h_1, \ldots, h_k\}$, se desea identificar aquella formulación $h \in \boldsymbol{h}$ que brinde una mejora durante diferentes etapas del proceso evolutivo. Es decir, se desea resolver un problema de optimización multi-objetivo mediante el uso simultáneo de diferentes formulaciones del mismo.

Existe poco trabajo en la literatura que aborde exactamente este problema. Sin embargo, es posible encontrar problemas relacionados, entre ellos, la selección adaptativa de operadores y la

selección adaptativa de funciones auxiliares. La selección adaptativa de operadores (*adaptive operator selection*) [8, 10] tiene como objetivo identificar al operador de variación más adecuado para un algoritmo evolutivo mediante un mecanismo adaptativo [8]. Se asume que existen operadores que se desempeñan mejor en un problema determinado; por lo tanto, a partir de un conjunto de operadores de variación, la selección automática de operadores trata de encontrar aquél que se adapte mejor a un problema dado. Por otro lado, la selección adaptativa de funciones auxiliares (*adaptive selection of helper-objectives*) [2, 5] tiene como objetivo identificar a la función auxiliar más adecuada para un algoritmo evolutivo. Se ha reportado en la literatura que es posible facilitar la resolución de un problema de optimización mono-objetivo, denotado por f, al incorporar un conjunto de funciones auxiliares correlacionadas con f. Sin embargo, no es posible conocer *a priori* cuál función auxiliar es la más adecuada en un punto determinado de la búsqueda. Resulta claro que ambos problemas son similares entre sí, ya que ambos desean identificar aquel elemento (operador de variación o función auxiliar) que brinde una mejora al proceso de búsqueda a partir de un conjunto de elementos. Además, existe la posibilidad de adaptar los trabajos previos a nuestro problema de investigación. La falta de un método capaz de seleccionar la formulación h más adecuada orientado a la resolución de problemas de optimización multi-objetivo representa la principal motivación de nuestro trabajo.

A partir de la revisión del estado del arte, en esta tesis se proponen dos nuevos esquemas capaces de abordar diferentes formulaciones de un problema de optimización multi-objetivo: basado en un modelo de islas y basado en aprendizaje por refuerzo. Ambas propuestas han sido comparadas con respecto al esquema de hibridación basado en selección aleatoria existente en la literatura [24, 27].

1.2 Solución propuesta

En la Figura 1.1 se muestra el diagrama de la solución propuesta, compuesto por tres bloques. En el bloque inferior se encuentra el conjunto de formulaciones $\boldsymbol{h}$ que podrían facilitar la resolución del problema original. En este bloque se incluyen tanto la formulación original como aquellas

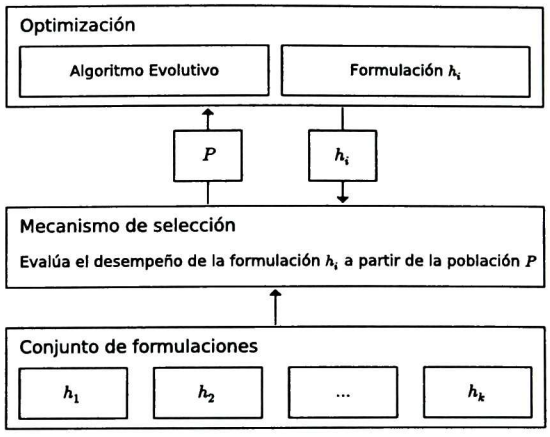


Figura 1.1: Diagrama de bloques de la propuesta.

formulaciones generadas por escalarización. En el bloque intermedio se encuentra el mecanismo de selección, el cual toma una formulación $h_i \in \boldsymbol{h}$ y la dirige al bloque de optimización, el cual devuelve una población P. A partir de dicha población se evalúa el desempeño de h_i en esta etapa de la búsqueda. Por último, el bloque de optimización emplea un algoritmo evolutivo durante ρ generaciones utilizando la formulación h_i seleccionada. Para evitar un conflicto entre términos, cada vez que se haga mención al término *selección* nos referimos a la identificación del elemento más adecuado a partir de un conjunto dado, por lo que no se debe confundir con las etapas de selección de individuos en un algoritmo evolutivo.

1.3 Hipótesis

Se puede incrementar el desempeño promedio de un algoritmo evolutivo mediante el uso adecuado de diferentes formulaciones de un problema de optimización multi-objetivo durante el proceso de búsqueda.

1.4 Objetivos general y específicos

1.4.1 Objetivo general

Proponer un algoritmo evolutivo híbrido que alterne el uso de diferentes formulaciones de un problema de optimización multi-objetivo y que sea competitivo con respecto a algoritmos representativos del estado del arte.

1.4.2 Objetivos específicos

1. Desarrollar un estudio que permita entender los beneficios e inconvenientes del uso conjunto de diferentes formulaciones en el proceso de búsqueda.
2. Proponer un algoritmo híbrido que permita intercambiar el modo de búsqueda para mejorar el desempeño promedio.
3. Desarrollar un estudio que permita conocer en qué problemas es conveniente emplear la hibridación propuesta.

1.5 Estructura de la tesis

Esta tesis está organizada de la siguiente manera:

- El Capítulo 2 aborda conceptos previos relacionados con optimización multi-objetivo, computación evolutiva, así como la notación empleada a lo largo de la tesis.
- El Capítulo 3 presenta la revisión del estado del arte. En este capítulo se describe el primer esquema de hibridación basado en selección aleatoria estudiado en esta tesis.
- El Capítulo 4 describe el algoritmo propuesto basado en un modelo de islas, siendo el segundo esquema de hibridación estudiado en esta tesis.

- El Capítulo 5 describe el algoritmo propuesto basado en aprendizaje por refuerzo, siendo el tercer esquema de hibridación estudiado en esta tesis.

- El Capítulo 6 comprende la experimentación realizada al evaluar los tres esquemas de hibridación mencionados previamente.

- Finalmente, el Capítulo 7 está orientado a la revisión de conclusiones y trabajo futuro.

2

Conceptos previos

2.1 Optimización

La optimización es el acto de encontrar la mejor solución bajo ciertas circunstancias [41]. No obstante, en la práctica usualmente no es posible encontrar dicha solución debido a restricciones de tiempo o de costo, o incluso a la misma dificultad del problema.

Resolver un problema de optimización implica encontrar una solución tal que maximice/minimice un recurso, por ejemplo, la maximización de la producción de un artículo o la minimización del esfuerzo requerido para realizar una tarea. Este recurso (producción, esfuerzo) debe estar definido mediante una función f, llamada *función objetivo*. Un *problema de optimización mono-objetivo* es aquel problema que involucra sólo una función a optimizar, mientras que un *problema de optimización multi-objetivo* involucra múltiples funciones, generalmente en conflicto entre sí. Muchos de los problemas del mundo real son de este último tipo. Por ejemplo, considere la compra de un auto. En este problema existen múltiples aspectos que se desean optimizar, entre ellos, el costo y la comodidad,

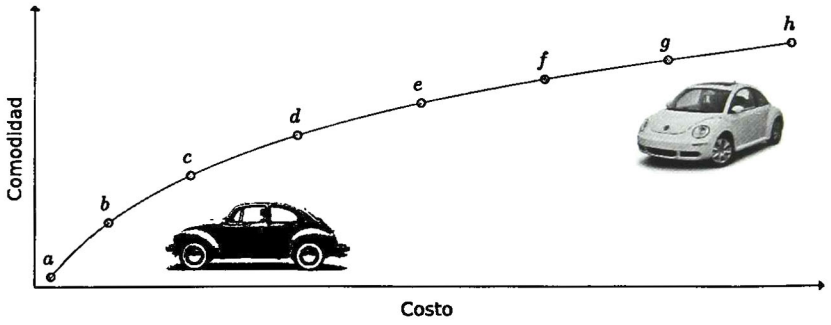


Figura 2.1: Objetivos involucrados en la compra de un auto: costo y comodidad. Cada una de las posibles opciones $\{a, b, c, d, e, f, g, h\}$ representa un compromiso entre ambos objetivos.

tal como se ilustra en la Figura 2.1. Es de esperarse que el auto con el menor costo (solución a) provea el menor nivel de comodidad. Asimismo, el auto que provee el mayor nivel de comodidad (solución h) tendrá el mayor costo. Cada una de las posibles opciones representa un compromiso entre ambos objetivos.

En un problema de optimización mono-objetivo, generalmente sólo existe una única solución óptima. Sin embargo, en un problema de optimización multi-objetivo usualmente no existe una única solución que mejore todos los objetivos de manera simultánea. En su lugar, existe un conjunto de soluciones que representan un compromiso entre objetivos.

A pesar de que existen múltiples soluciones óptimas, en la práctica sólo se requiere una de ellas. Si todos los objetivos tienen la misma importancia, entonces la selección de una solución no es una tarea trivial, motivo por el cual se requiere de un tomador de decisiones. El *tomador de decisiones* es aquella persona capaz de elegir una solución de entre un conjunto de soluciones comparables entre sí, haciendo uso de su experiencia, información de alto nivel y preferencias. De este modo, al disponer de un conjunto de soluciones, éste puede evaluar las ventajas y desventajas de cada una de ellas y así realizar una selección. Por este motivo, la principal finalidad de un método de optimización multi-objetivo es encontrar el conjunto de soluciones compromiso al considerar a todos los objetivos por igual y así un tomador de decisiones pueda elegir la mejor solución con base en sus preferencias. La Figura 2.2 ilustra las etapas descritas previamente.

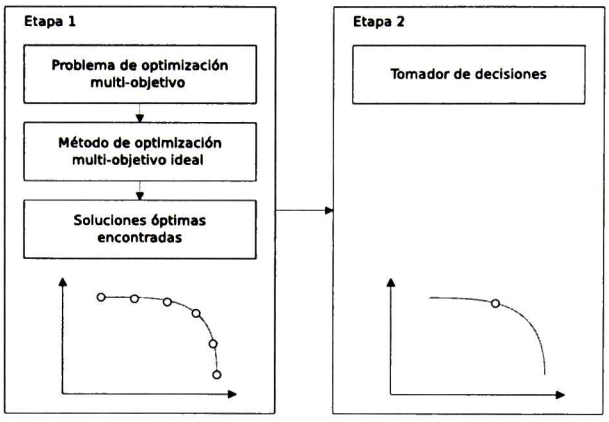


Figura 2.2: Etapas involucradas en la resolución de un problema de optimización multi-objetivo.

2.1.1 Problema de optimización multi-objetivo

Un problema de optimización multi-objetivo involucra un número de funciones objetivo que se desea minimizar o maximizar. El problema puede estar sujeto a un número de restricciones que deben ser satisfechas por cualquier solución factible. Dado que los objetivos pueden ser minimizados o maximizados, se define el problema de optimización multi-objetivo en su forma general de la siguiente manera [13]:

$$\begin{aligned} \text{Minimizar/Maximizar} \quad & \boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})]^T & & \\ \text{sujeto a} \quad & g_j(\boldsymbol{x}) \leq 0 & j &= 1, \ldots, p \\ & h_k(\boldsymbol{x}) = 0 & k &= 1, \ldots, q \\ & L_i \leq x_i \leq U_i & i &= 1, \ldots, n \end{aligned} \tag{2.1}$$

donde $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$ es un vector con n variables de diseño, $\boldsymbol{f}(\boldsymbol{x})$ es el vector de m funciones objetivo, g_j denota p restricciones de desigualdad, h_k denota q restricciones de igualdad, el superíndice T denota transpuesto y las variables L_i y U_i definen los límites inferior y superior para la i-ésima variable de diseño en $\boldsymbol{x}$, respectivamente. Las soluciones que satisfacen las restricciones y los límites de las variables constituyen el *espacio factible de variables de decisión*, $X \subseteq \mathbb{R}^n$, de aquí en adelante

conocido simplemente como *espacio de decisión*. La imagen $\boldsymbol{f}(\boldsymbol{x})$ de cada vector de diseño constituye el *espacio objetivo*, $Z \subseteq \mathbb{R}^m$. Para cada *vector de diseño*, $\boldsymbol{x} \in X$, existe un *vector objetivo*, $\boldsymbol{z} \in Z$, denotado por $\boldsymbol{z} = [z_1, z_2, \cdots, z_m]^T$, en donde $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x})$.

2.1.2 Optimalidad de Pareto

Al existir múltiples funciones objetivo en conflicto, la noción de óptimo cambia debido a que en un problema de optimización multi-objetivo la principal finalidad es encontrar un conjunto de soluciones que representen un compromiso entre los objetivos en lugar de una única solución. La noción de optimalidad comúnmente usada es llamada *optimalidad de Edgeworth-Pareto*, también conocida simplemente como *optimalidad de Pareto*, cuya definición se provee a continuación [12, 21]. En las definiciones siguientes se asume que $\{\boldsymbol{x}, \boldsymbol{y}\} \in X$ denotan dos vectores de diseño y, sin pérdida de generalidad, se asume un problema de minimización.

Definición 2.1. Óptimo de Pareto. Un vector de diseño $\boldsymbol{x}$ es óptimo de Pareto si no existe otro vector de diseño $\boldsymbol{y}$ tal que $f_i(\boldsymbol{y}) \leq f_i(\boldsymbol{x})$, $\forall i \in \{1, \ldots, m\}$ y $f_j(\boldsymbol{y}) < f_j(\boldsymbol{x})$, $\exists j \in \{1, \ldots, m\}$.

Las soluciones óptimas de Pareto son aquellos vectores de diseño, $\boldsymbol{x} \in X$, cuyas componentes de su imagen, $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x})$, no pueden ser mejoradas simultáneamente. Estas soluciones también son llamadas soluciones no-inferiores, admisibles o eficientes. Al conjunto de todos los vectores de diseño óptimos de Pareto se le conoce como *conjunto de óptimos de Pareto*, y se denota por $\mathcal{P}^*$. A la imagen de $\mathcal{P}^*$ en el espacio objetivo se le conoce como *frente óptimo de Pareto*, o simplemente *frente de Pareto*, y se denota por $\mathcal{FP}^*$

Ejemplo 2.1. Considere el siguiente problema de minimización [31] en donde el espacio de decisión X consiste de todos los pares $(x_1, x_2) \in \{0, \ldots, r\} \times \{0, \ldots, r\}$ y el espacio objetivo Z está definido por dos funciones:

$$f_1(x_1, x_2) = x_1 + x_2 \qquad f_2(x_1, x_2) = r - x_2 + x_1 \tag{2.2}$$

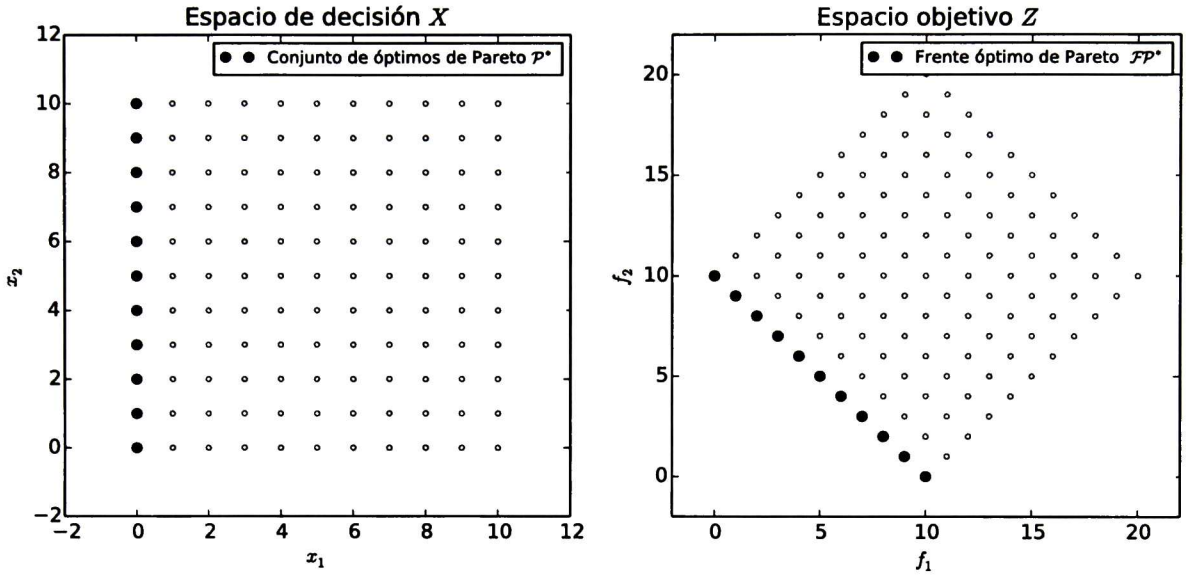


Figura 2.3: Espacio de decisión (izquierda) y espacio objetivo (derecha) de un problema con dos objetivos: $f_1(x_1, x_2) = x_1 + x_2$ y $f_2(x_1, x_2) = 10 - x_2 + x_1$.

La Figura 2.3 muestra ambos espacios para $r = 10$. El conjunto de óptimos de Pareto está definido por $\{(0, x_2) : x_2 \in \{0, \ldots, r\}\}$ y el frente óptimo de Pareto está definido por $\{(z_1, z_2) : x_1 + x_2 = r \wedge (x_1, x_2) \in \{0, \ldots, r\}\}$.

Definición 2.2. Conjunto de óptimos de Pareto. El conjunto de óptimos de Pareto, denotado por $\mathcal{P}^*$, se define como:

$$\mathcal{P}^* = \{\boldsymbol{x} \in X : \boldsymbol{x} \text{ es óptimo de Pareto } \}$$

Definición 2.3. Frente óptimo de Pareto. El frente óptimo de Pareto, denotado por $\mathcal{PF}^*$, corresponde a la imagen del conjunto óptimo de Pareto en el espacio objetivo y se define como:

$$\mathcal{FP}^* = \{\boldsymbol{f}(\boldsymbol{x}) \in Z : \boldsymbol{x} \in \mathcal{P}^*\}$$

El concepto de *dominancia* es utilizado para comparar dos soluciones $\{\boldsymbol{x}, \boldsymbol{y}\} \in X$ y determinar cuál de ellas *domina* a la otra, es decir, cuál solución es mejor al tomar en cuenta todos los objetivos.

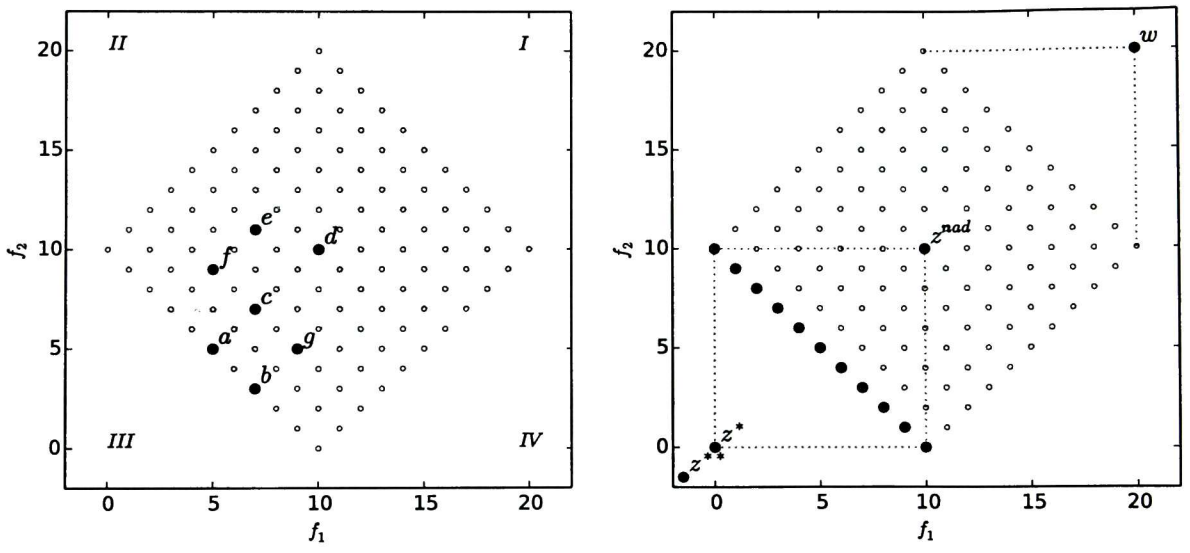


Figura 2.4: (Izquierda) Espacio objetivo dividido en cuatro regiones tomando al vector objetivo de la solución c como referencia. (Derecha) Vectores objetivo ideal $\boldsymbol{z}^*$, utópico $\boldsymbol{z}^{**}$ y de nadir $\boldsymbol{z}^{nad}$ y el vector objetivo compuesto por los peores valores en cada objetivo, w.

Definición 2.4. Dominancia débil de Pareto. Se dice que una solución $\boldsymbol{x}$ *domina débilmente* a una solución $\boldsymbol{y}$, denotado por $\boldsymbol{x} \preceq \boldsymbol{y}$, si y sólo si $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y}), \forall i \in \{1, \ldots, m\}$.

Definición 2.5. Dominancia de Pareto. Se dice que una solución $\boldsymbol{x}$ *domina* a una solución $\boldsymbol{y}$, denotado por $\boldsymbol{x} \prec \boldsymbol{y}$, si y sólo si $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y}) \wedge f_j(\boldsymbol{x}) < f_j(\boldsymbol{y}), \forall i \in \{1, \ldots, m\}, \exists j \in \{1, \ldots, m\}$.

Definición 2.6. Dominancia estricta de Pareto. Se dice que una solución $\boldsymbol{x}$ *domina estrictamente* a una solución $\boldsymbol{y}$, denotado por $\boldsymbol{x} \prec_s \boldsymbol{y}$, si y sólo si $f_i(\boldsymbol{x}) < f_i(\boldsymbol{y}), \forall i \in \{1, \ldots, m\}$.

Para las siguientes definiciones, se asume que el símbolo $\lhd$ puede ser cualquier elemento del conjunto $\{\preceq, \prec, \prec_s\}$.

Definición 2.7. Incomparabilidad. Dos soluciones $\boldsymbol{x}$ y $\boldsymbol{y}$ son *incomparables*, denotado por $\boldsymbol{x} \,||\, \boldsymbol{y}$ con respecto a $\lhd$, si y sólo si $\boldsymbol{x} \not\lhd \boldsymbol{y} \wedge \boldsymbol{y} \not\lhd \boldsymbol{x}$.

Definición 2.8. Indiferencia. Dos soluciones $\boldsymbol{x}$ y $\boldsymbol{y}$ son *indiferentes*, denotado por $\boldsymbol{x} \sim \boldsymbol{y}$, si y sólo si $f_i(\boldsymbol{x}) = f_i(\boldsymbol{y}), \forall i \in \{1, \ldots, m\}$.

Relación	Notación	Interpretación en el espacio objetivo	Ejemplo
Dominancia estricta	$\boldsymbol{x} \prec_s \boldsymbol{y}$	$\boldsymbol{x}$ es mejor que $\boldsymbol{y}$ en todos los objetivos	$\boldsymbol{z}^* \prec_s c$
Dominancia	$\boldsymbol{x} \prec \boldsymbol{y}$	$\boldsymbol{x}$ no es peor $\boldsymbol{y}$ en todos los objetivos y es estrictamente mejor en al menos un objetivo	$c \prec d$
Dominancia débil	$\boldsymbol{x} \preceq \boldsymbol{y}$	$\boldsymbol{x}$ no es peor $\boldsymbol{y}$ en todos los objetivos	$b \preceq c,\ c \preceq e$
Incomparabilidad	$\boldsymbol{x} \parallel \boldsymbol{y}$	$\boldsymbol{x}$ y $\boldsymbol{y}$ son no dominados entre sí	$c \parallel f,\ c \parallel g$
Indiferencia	$\boldsymbol{x} \sim \boldsymbol{y}$	$\boldsymbol{x}$ y $\boldsymbol{y}$ tienen el mismo valor en cada objetivo	$d \sim \boldsymbol{z}^{nad}$

Tabla 2.1: Descripción de las diferentes relaciones de comparabilidad.

Ejemplo 2.2. La Figura 2.4 (derecha) resalta los vectores objetivo correspondientes a las soluciones $\{a, b, c, d, e, f, g\} = \{(5,5), (7,3), (7,7), (10,10), (7,11), (5,9), (9,5)\}$ del problema (2.2). El espacio objetivo se ha dividido en cuatro regiones, denotadas por I, II, III y IV, tomando a c como referencia. La región I contiene todas las soluciones dominadas por c. La región III contiene todas las soluciones que dominan a c. Por último, las regiones II y IV contienen soluciones incomparables respecto a c. La Tabla 2.1 describe con mayor claridad las relaciones de comparabilidad basadas en optimalidad de Pareto.

2.1.3 Vectores especiales

Para cada una de las m funciones objetivo, existe un valor óptimo individual, denotado por f_i^*, y generalmente es distinto para cada objetivo. El vector objetivo formado con estos valores óptimos constituye el *vector objetivo ideal* [39, 13].

Definición 2.9. Vector objetivo ideal. El vector objetivo ideal, denotado por $\boldsymbol{z}^*$, es el vector formado por los valores óptimos de cada una de las funciones objetivo. Cada una de sus componentes se define de la siguiente manera:

$$z_i^* = f_i^*, \quad i = 1, \ldots, m \tag{2.3}$$

Si el vector objetivo ideal fuera factible (es decir, $z^* \in Z$), entonces el conjunto óptimo de Pareto estaría formado por una única solución. Sin embargo, esto no es posible debido a que generalmente los objetivos se encuentran en conflicto. Aún cuando no sea un vector factible, puede ser considerado como un punto de referencia al cual se deben dirigir las soluciones potenciales. A partir del vector objetivo ideal z^* se obtienen los límites inferiores del conjunto de óptimos de Pareto en cada función objetivo [39, 13]. En la práctica, algunos métodos de optimización requieren de un punto de referencia que sea mejor que el vector objetivo ideal z^*, es decir, que domine estrictamente a cada óptimo de Pareto. Este vector se conoce como *vector objetivo utópico* [39, 13].

Definición 2.10. Vector objetivo utópico. Un vector objetivo utópico, denotado por z^{**}, es un vector infactible cuyos componentes se definen de la siguiente manera:

$$z_i^{**} = z_i^* - \epsilon, \quad i = 1, \ldots, m \tag{2.4}$$

donde z_i^* representa la i-ésima componente de z^* y $\epsilon > 0$. A diferencia del vector objetivo ideal, que representa el límite inferior de cada función objetivo, el *vector objetivo de nadir*, denotado por z^{nad}, representa el límite superior de cada función objetivo del conjunto óptimo de Pareto y no del espacio de búsqueda. El vector objetivo de nadir no debe ser confundido con el vector objetivo compuesto por los peores valores en cada función. La Figura 2.4 (derecha) muestra los vectores objetivo ideal z^*, utópico z^{**}, de nadir z^{nad} distribuidos en el espacio objetivo correspondiente al problema (2.2). Adicionalmente, se ilustra al vector objetivo compuesto por los peores valores en cada objetivo, denotado por w.

2.2 Técnicas clásicas

Las técnicas mencionadas en esta sección pertenecen al área de *investigación de operaciones* (*Operation Research*, OR). Estas técnicas, llamadas de aquí en adelante como *técnicas clásicas*

para distinguirlas de las técnicas evolutivas, no abordan directamente el problema de optimización multi-objetivo como tal, sino que realizan una escalarización del mismo.

Las técnicas clásicas se pueden clasificar de acuerdo a varios criterios, entre ellos, el instante en el cual el tomador de decisiones expresa sus preferencias durante el proceso de resolución del problema de optimización, dando lugar a cuatro clases de métodos: sin preferencias, *a priori*, *a posteriori*, e interactivos. En los métodos sin preferencias, la opinión del tomador de decisiones no es requerida, el problema de optimización multi-objetivo es resuelto usando algún método relativamente simple y la solución obtenida es presentada al tomador de decisiones, quien la acepta o la rechaza. En los métodos *a priori*, el tomador de decisiones debe especificar sus preferencias antes del proceso de resolución. Sin embargo, puede ser difícil ya que es posible que no conozca de antemano qué tan realistas sean sus preferencias. Los métodos *a posteriori* se enfocan en generar el conjunto de óptimos de Pareto (o parte del mismo), para luego presentarlo al tomador de decisiones, quien selecciona las soluciones más adecuadas. Por último, en los métodos interactivos, el tomador de decisiones trabaja en conjunto con un analista o un programa de computadora interactivo. Cada vez que el analista le presenta las soluciones encontradas, el tomador de decisiones puede especificar y corregir sus preferencias. De este modo, se logra una mayor confianza en las soluciones encontradas. En la Tabla 2.2 se muestran algunas técnicas asociadas a las clases descritas. Dos de las técnicas clásicas más empleadas son la suma ponderada y restricción-ϵ, las cuales se describen a continuación considerando la maximización de un problema multi-objetivo.

La suma ponderada permite escalarizar un conjunto de m objetivos en un único objetivo al multiplicar cada objetivo f_i, por un peso definido por el usuario, denotado por w_i, donde $i = 1, \ldots, m$. De este modo es posible transformar un problema de optimización multi-objetivo en uno mono-objetivo de la siguiente forma [39, 49]:

$$\begin{aligned} \text{Maximizar} \quad & \textstyle\sum_{i=1}^{m} w_i f_i(\boldsymbol{x}) \\ \text{sujeto a} \quad & \boldsymbol{x} \in X \end{aligned} \tag{2.5}$$

Clase	Técnicas
Sin preferencias	Criterio Global (*Global Criterion*)
	Conjunto Próximo Multiobjetivo (*Multiobjective Proximal Bundle*)
A priori	Orden Lexicográfico (*Lexicographic Ordering*)
	Programación de Metas (*Goal Programming*)
A posteriori	Suma Ponderada (*Weighted Sum*)
	Restricción-ϵ (ϵ-*Constraint*)
	Métricas ponderadas (*Weighted Metrics*)
Interactivos	Método de Tchebycheff (*Tchebycheff Method*)
	Método de Punto de Referencia (*Reference Point Method*)

Tabla 2.2: Ejemplos de técnicas clásicas.

donde los pesos $w_i \geq 0$ denotan la importancia relativa de los objetivos. Usualmente se asume que los pesos están normalizados:

$$\sum_{i=0}^{m} w_i = 1 \tag{2.6}$$

La principal desventaja de esta técnica es su sensibilidad al contorno del frente de Pareto. Por ejemplo, considere un problema de maximización con dos objetivos, tal como se ilustra en la Figura 2.5. Para pesos fijos, w_1, w_2, la solución $\boldsymbol{x}$ busca maximizar:

$$y = w_1 f_1(\boldsymbol{x}) + w_2 f_2(\boldsymbol{x}) \tag{2.7}$$

y el problema puede reformularse como:

$$f_2(\boldsymbol{x}) = -\frac{w_1}{w_2} f_1(\boldsymbol{x}) + \frac{y}{w_2} \tag{2.8}$$

La función f_2 define una línea con pendiente $-w_1/w_2$ e intersección en y/w_2 en el espacio objetivo (línea gruesa en la Figura 2.5). Gráficamente, el proceso de optimización corresponde a mover esta línea hacia arriba hasta que no exista ningún vector objetivo factible por encima de ella y se encuentre

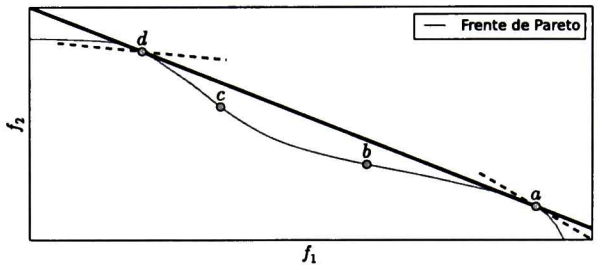


Figura 2.5: Interpretación gráfica de la suma ponderada.

al menos un vector objetivo factible sobre la misma (en este caso, los vectores a y d). Al cambiar los pesos w_1 y w_2 es posible cambiar la pendiente e intersección de la línea, pudiendo alcanzar otros puntos (por ejemplo, las líneas punteadas logran alcanzar los puntos a y d individualmente). Sin embargo, no es posible alcanzar los puntos b y c con este procedimiento, incluso si se modifican los pesos w_1 y w_2, siendo ésta la principal desventaja de este método.

La restricción-ϵ es una técnica que permite elegir arbitrariamente un objetivo a optimizar y convertir el resto de los objetivos en restricciones al definir un límite inferior[1] para cada uno [39]:

$$\begin{aligned} \text{Maximizar} \quad & f_i(\boldsymbol{x}) \\ \text{sujeto a} \quad & f_j(\boldsymbol{x}) \geq \epsilon_j \quad j = 1, \ldots, m, \quad j \neq i \\ & \boldsymbol{x} \in X \end{aligned} \tag{2.9}$$

donde $f_i(\boldsymbol{x})$ denota el objetivo seleccionado a maximizar, $f_j(\boldsymbol{x})$ denota un conjunto de restricciones de desigualdad y ϵ_j denota un conjunto de límites inferiores. La idea de este método es maximizar un objetivo a la vez, considerando a los otros objetivos como restricciones limitadas por ϵ_j. Es posible encontrar múltiples soluciones óptimas de Pareto al variar el límite ϵ_j.

Considere un problema de optimización de dos objetivos, como se detalla en la Figura 2.6. El objetivo $f_1(\boldsymbol{x})$ se desea maximizar, mientras que el objetivo $f_2(\boldsymbol{x})$ se ha convertido en una restricción limitada por $\epsilon_2 = r$. Con esta restricción, el punto b maximiza el objetivo $f_1(\boldsymbol{x})$ mientras que el punto

[1]Si se trata de un problema de minimización, el límite es superior.

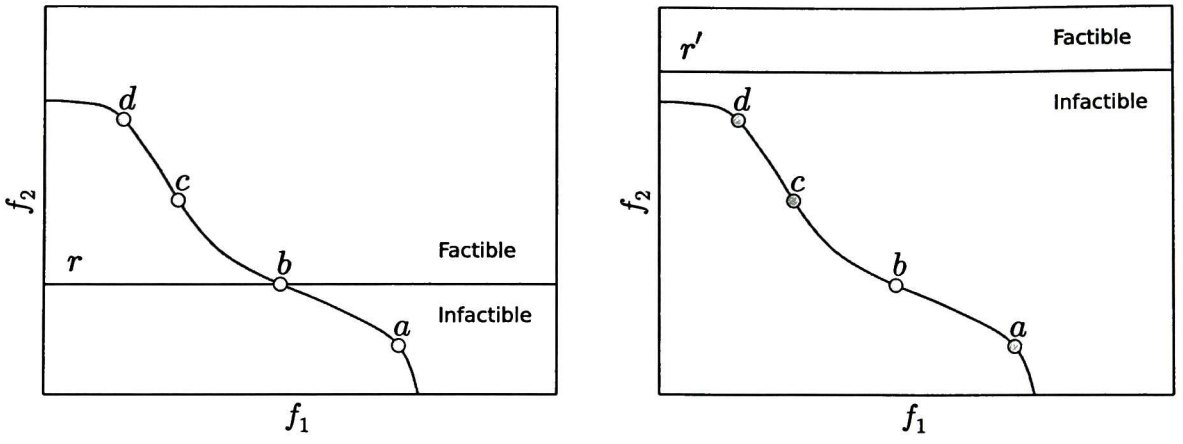


Figura 2.6: Interpretación gráfica de restricción-ϵ.

a, a pesar de ser mayor que b considerando el mismo objetivo, se convierte en una solución infactible. Por otra parte, si $\epsilon_2 = r'$ todo el espacio objetivo se vuelve infactible. Para evitar esta situación, un rango adecuado de valores para ϵ_j debe ser conocido previamente [49].

Los métodos descritos sugieren una manera para convertir un problema de optimización multi-objetivo en uno mono-objetivo en función de una serie de parámetros de conversión, tales como vectores de pesos o límites ϵ. La suma ponderada trata de optimizar la suma de objetivos, mientras que la restricción ϵ permite optimizar un único objetivo y convertir el resto en restricciones. Sin embargo, se pueden observar algunas dificultades para éstos y otros métodos [13]:

- Sólo se puede encontrar una solución óptima de Pareto por ejecución. Debido a esto, para obtener un mayor número de soluciones se debe transformar el problema mediante alguna técnica clásica y resolverlo, usando parámetros de conversión distintos en cada ejecución.

- Algunos métodos son sensibles al contorno del frente de Pareto. Anteriormente se describió gráficamente esta desventaja para la suma ponderada (Figura 2.5), indicando así que no importa qué vector de pesos se use, las soluciones óptimas de Pareto ubicadas en una región no convexa no pueden ser alcanzadas por este tipo de métodos.

- Todos los métodos requieren de algún tipo de conocimiento del problema, tal como vectores de pesos o los límites ϵ.

A pesar de estos inconvenientes, los métodos clásicos presentan ventajas por las cuales son usadas para resolver problemas de optimización multi-objetivo del mundo real [13]:

- Para algunos métodos, existen teoremas que demuestran que la solución obtenida a partir de la reformulación del problema original corresponde a un óptimo de Pareto [13].
- Son simples y fáciles de implementar.

La suma ponderada garantiza que cada solución óptima de Pareto corresponde a una solución óptima del problema simplificado para un problema multi-objetivo convexo. Por otro lado, el método de restricción-ϵ garantiza lo anterior para problemas convexos y no convexos.

2.3 Computación evolutiva

La computación evolutiva (*Evolutionary Computing*, EC) es un área de investigación dentro de las ciencias de la computación que comprende un extenso conjunto de algoritmos bioinspirados capaces de abordar problemas de optimización. Como su nombre sugiere, es un tipo especial de computación basado en el proceso de evolución natural, en donde una población de individuos es ubicada en un entorno con recursos limitados [17]. Debido a esto, los individuos compiten entre sí para sobrevivir y reproducirse. La *aptitud* define el grado en el que un individuo ha alcanzado sus logros. De este modo, una mayor aptitud favorece una mayor probabilidad para sobrevivir y multiplicarse. Al trasladar esta metáfora a la resolución de un problema, cada individuo representa una solución potencial, en donde su calidad está en función de su aptitud. La relación entre los elementos del cómputo evolutivo y la resolución de un problema de optimización se muestran en la Tabla 2.3. La metáfora de la computación evolutiva se basa en dos principios:

Evolución		Resolución
Entorno	$\longleftrightarrow$	Problema
Individuo	$\longleftrightarrow$	Solución potencial
Aptitud	$\longleftrightarrow$	Calidad

Tabla 2.3: Relación entre elementos de la metáfora del cómputo evolutivo y la resolución de un problema de optimización.

- **Supervivencia del más apto**. Dado que los individuos compiten entre sí, el proceso de selección natural favorece a aquellos que se ajusten o adapten mejor a las condiciones del entorno.

- **Variaciones genéticas**. Cada individuo se compone de un conjunto de elementos conocidos como genes, los cuales a su vez conforman un cromosoma. Estos genes definen la aptitud de un individuo, de modo que una adecuada secuencia de genes podría mejorar su aptitud. Dentro de la teoría evolutiva, durante la etapa de reproducción se generan variaciones pequeñas y aleatorias, conocidas como mutaciones. A través de estas variaciones es posible generar nuevos cromosomas y mejorar la aptitud de futuros individuos.

Tomar ideas de la naturaleza y aplicarlas al diseño de algoritmos de optimización no es algo nuevo. Durante la década de 1960 tres distintas propuestas fueron creadas a partir de la misma idea básica en diferentes lugares. En Estados Unidos, Fogel *et al.* desarrollaron la programación evolutiva (*Evolutionary Programming*, EP) [19], mientras que Holland definió al algoritmo genético (*Genetic Algorithm*, GA) [22]. Por otra parte, en Alemania, Rechenberg diseñó las estrategias evolutivas (*Evolutionary Strategies*, ES) [42]. Años más tarde, Koza desarrolló la programación genética (*Genetic Programming*) [33]. Todos estos métodos forman parte de la computación evolutiva y son globalmente llamados *algoritmos evolutivos* (*evolutionary algorithm*, EA). Cada uno de estos enfoques comparte la misma inspiración biológica y difieren únicamente en detalles técnicos, tal como la representación de los individuos. En un algoritmo genético, un individuo se representa como una cadena tomada de un alfabeto finito (generalmente binario), en una estrategia evolutiva se emplean

Algoritmo 1 Algoritmo Genético Simple

1: $P \leftarrow$ INITIALIZE(P)
2: **while** condición de paro **do**
3: $P' \leftarrow$ PARENTSELECTION(P)
4: $P'' \leftarrow$ GENETICOPERATIONS(P')
5: $P \leftarrow$ GENERATIONUPDATE($P \cup P''$)
6: **end while**

vectores con números reales, en programación evolutiva se utilizan máquinas de estado finito y árboles en programación genética. En la siguiente sección se describirá el funcionamiento y los componentes de un algoritmo genético por ser el algoritmo evolutivo más popular. Sin embargo, cabe mencionar que los mismos principios aplican para otros algoritmos.

2.3.1 Algoritmo genético

Los algoritmos genéticos son métodos de búsqueda estocástica diseñados para explorar espacios de problemas complejos con la finalidad de encontrar soluciones subóptimas usando información mínima sobre el problema [3]. A diferencia los métodos de optimización basados en gradiente, un algoritmo genético no requiere de este tipo de información para guiar la búsqueda. Este tipo de algoritmo trata de imitar el proceso de evolución natural al aplicar operadores de cruza, mutación y selección sobre una población de individuos, denotada por P. La calidad de cada individuo se define por medio de una función de aptitud, de modo que los más aptos tienen una mayor probabilidad de sobrevivir y multiplicarse. El Algoritmo 1 muestra la estructura general de un algoritmo genético simple. La primera etapa en un algoritmo genético es la selección de padres, la cual consiste en elegir a los individuos que participarán en la etapa de cruza. A partir de una población de padres, P', se genera una nueva población de hijos, P'', por medio de operadores genéticos. Por último, se realiza una segunda selección para elegir a los individuos que sobrevivirán a partir de ambas poblaciones, $P \cup P''$, y formarán parte de la siguiente generación. Este procedimiento se repite hasta alcanzar un criterio de paro, generalmente definido por un número de iteraciones llamadas *generaciones*.

El primer paso al definir un algoritmo evolutivo es enlazar el espacio del problema de optimización y el espacio donde la evolución se lleva a cabo. Los elementos que representan posibles soluciones en el contexto del problema de optimización son llamados *fenotipos*, mientras que su codificación, es decir, los individuos en el contexto de un algoritmo evolutivo, son llamados *genotipos*. La *representación* define la relación entre fenotipos y genotipos. Considere el siguiente ejemplo. Dado un problema de optimización (maximización) definido por $f(x) = x^2$, el *espacio fenotípico* equivale al espacio de decisión X, y cada solución $x \in X$ representa un fenotipo. Por otro lado, al utilizar una codificación binaria, cada genotipo se define por medio de una cadena de bits. De este modo, a partir de una solución hipotética, $x = 18$, se define su fenotipo, 18, y su genotipo, 10010. Al conjunto de todos los fenotipos se conoce como *espacio fenotípico*, mientras que el conjunto de todos los genotipos se conoce como *espacio genotípico* o *espacio de búsqueda*.

En algunas ocasiones los términos *espacio de decisión* y *espacio de búsqueda* son usados indistintamente, sin embargo, existe una diferencia entre ellos. El espacio de decisión forma parte de la definición formal del problema de optimización. El espacio de búsqueda es la región en donde opera el algoritmo evolutivo y puede ser distinto del espacio de decisión [49]. Siguiendo con el ejemplo anterior, el espacio de decisión está compuesto por números enteros, $X \in \mathbb{Z}$, mientras que el espacio de búsqueda está compuesto por cadenas binarias de n bits, $\{0,1\}^n$.

La *función de evaluación* representa la tarea a resolver en el contexto evolutivo. Esta función asigna una medida de calidad o *aptitud* a un genotipo a partir de su fenotipo correspondiente. De acuerdo al ejemplo anterior, la aptitud del genotipo 10010 podría ser definida como $f(x) = 18^2 = 324$. La función de evaluación es llamada generalmente *función de aptitud* en computación evolutiva.

La *población* de un algoritmo evolutivo representa el conjunto de individuos o genotipos. Dada una representación, definir una población puede ser tan simple como especificar el número de individuos que existe en el conjunto, es decir, el *tamaño de la población*, el cual generalmente permanece fijo. Sin embargo, existen poblaciones más sofisticadas en las cuales los individuos son separados de acuerdo a una estructura espacial. Se distinguen principalmente dos tipos de poblaciones:

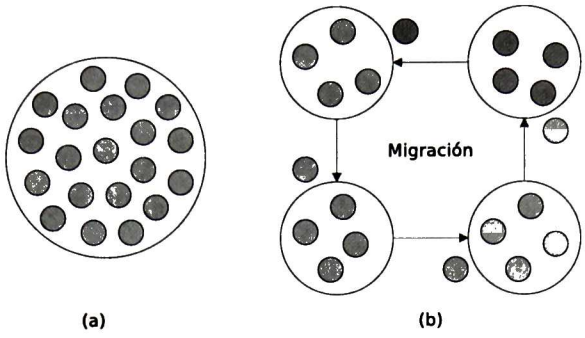


Figura 2.7: Diferentes tipos de poblaciones. (a) En una población panmíctica todos los individuos se encuentran en un mismo grupo. (b) En una población estructurada, los individuos son separados en distintos grupos.

- **Panmíctica**. La población es considerada como un solo conjunto, de modo que los operadores de selección y variación se aplican de manera global, es decir, cualquier individuo puede potencialmente recombinarse con cualquier otro.

- **Estructurada**. La población es organizada (dividida) de acuerdo a un modelo. Los algoritmos genéticos con este tipo de población son conocidos como algoritmos genéticos estructurados, entre los cuales se encuentra el algoritmo genético distribuido.

En la Figura 2.7 se ilustran los dos tipos de poblaciones. En (a) todos los individuos pertenecen a un único grupo, mientras que en (b) la población es dividida en cuatro diferentes grupos.

El *mecanismo de selección de padres* se encarga de elegir a los individuos, conocidos como padres, que participarán en la generación de nuevos individuos, conocidos como hijos. A pesar de que la selección está basada en la aptitud, generalmente se trata de un proceso probabilístico, en donde incluso los individuos con baja aptitud tienen cierta probabilidad de ser elegidos para formar parte del conjunto de padres. Si el proceso de selección sólo considera a los mejores individuos, entonces sería voraz y la población podría estancarse rápidamente en un óptimo local.

Los *operadores de variación* permiten generar nuevos genotipos a partir de individuos existentes. Estos operadores se dividen de acuerdo a su *aridad*, es decir, al número de individuos que requieren para generar nuevos individuos. Un operador de variación con aridad unitaria se conoce como *operador*

de mutación, el cual genera un nuevo genotipo al realizar cambios aleatorios sobre un genotipo existente. Un operador de variación con aridad binaria se conoce como *operador de cruza*, el cual genera uno o dos nuevos genotipos al intercambiar genes de dos padres. La finalidad de utilizar mutación es generar nuevos genes, mientras que el objetivo de la cruza es combinar características deseables de entre dos padres. Cabe mencionar que los operadores de variación dependen de la representación de los individuos. Así, por ejemplo, los operadores de variación que se emplean en un algoritmo genético simple son distintos a los que se usan en programación genética, ya que en el primero se usan cadenas binarias mientras que en el segundo se utilizan árboles.

A diferencia de la selección de padres, el *mecanismo de selección de sobrevivientes* se realiza luego de generar la población de hijos. Como se mencionó anteriormente, el tamaño de la población usualmente permanece fijo. Por este motivo, es necesario elegir a los individuos que formarán parte de la siguiente generación. En algunos casos, esta selección se realiza de manera determinista con la finalidad de elegir sólo a los mejores individuos. Junto con la selección de padres, la selección de sobrevivientes es responsable de mejorar la calidad de los individuos.

La *inicialización* define el método utilizado para generar la población inicial. Generalmente es tan simple como crear un conjunto de individuos de manera aleatoria. Sin embargo, es algunas ocasiones, es posible utilizar alguna heurística para crear una población inicial con mayor aptitud.

El *criterio de paro* define la condición de terminación del algoritmo evolutivo. Por ejemplo, es posible definir un umbral de aptitud. Si una individuo logra alcanzar dicho umbral, entonces el proceso de búsqueda termina. Sin embargo, en algunos casos este criterio puede no cumplirse. Por este motivo, existen otras condiciones, tales como:

- Número de evaluaciones de la función objetivo.

- Número de generaciones en las cuales la aptitud no se ha mejorado.

2.3.2 Algoritmo genético distribuido

A diferencia de un algoritmo genético panmíctico, donde la población es considerada como un solo conjunto, la población de un algoritmo genético distribuido es dividida en diferentes subpoblaciones, llamadas islas o demes. Cada isla se encuentra aislada de las demás y evoluciona independientemente. Sin embargo, existen intercambios de individuos entre islas cada determinado tiempo. Además, los operadores genéticos (selección, mutación y cruza) se realizan a nivel de isla, lo que implica que cada isla puede buscar en diferentes regiones del espacio de búsqueda.

En un algoritmo genético distribuido se requiere de una política de migración encargada de definir la topología del modelo y controlar la migración de individuos, así como la forma en la que los individuos son seleccionados y reemplazados durante cada intercambio. Los principales parámetros de la política de migración incluyen los siguientes:

- **Intervalo de migración** ρ. Dado que algoritmo genético distribuido generalmente realiza intercambios de individuos entre subpoblaciones, se debe definir el intervalo de tiempo en que estos intercambios ocurren. Este parámetro define el número de pasos o generaciones en las que cada isla evoluciona independientemente antes de realizar un intercambio.

- **Razón de migración** η. Determina el número de individuos que se intercambian por cada migración. Este valor puede ser dado como un porcentaje del tamaño de la población o como un valor absoluto.

- **Operadores de selección/reemplazo de migrantes**. Define cómo serán seleccionados los individuos de la isla de origen y cómo se realizará el reemplazo en la isla de destino.

- **Topología**. Define el vecindario de cada isla. Cada vecindario se compone de aquellas islas a las que una subpoblación determinada puede enviar (o desde las que puede recibir) individuos.

Algoritmo 2 NSGA-II

1: $R_t \leftarrow P_t \cup Q_t$
2: $(F_1, \ldots, F_k) \leftarrow$ FASTNONDOMINATEDSORT(R_t) ▷ Ordenamiento de R_t
3: $P_{t+1} \leftarrow \emptyset$, $i \leftarrow 1$
4: **while** $|P_{t+1}| + |F_i| \leq N$ **do**
5: CROWDINGDISTANCEASSIGMENT(F_i) ▷ Estimación de densidad
6: $P_{t+1} \leftarrow P_{t+1} \cup F_i$ ▷ Agregar el frente F_i a la nueva población
7: $i \leftarrow i + 1$
8: **end while**
9: SORT$(F_i, \prec_c)$ ▷ Ordenar descendentemente de acuerdo a $\prec_c$
10: $P_{t+1} \leftarrow P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$ ▷ Seleccionar los primeros $N - |P_{t+1}|$ elementos
11: $Q_{t+1} \leftarrow$ GENETICOPERATIONS(P_{t+1}) ▷ Aplicar selección, cruza y mutación

2.3.3 Algoritmo evolutivo multi-objetivo

Dado que un algoritmo evolutivo manipula una población de soluciones potenciales, representa un método de optimización adecuado para resolver problemas de optimización multi-objetivo. De este modo, un algoritmo evolutivo tiene el potencial de encontrar múltiples óptimos de Pareto en una sola ejecución [11, 49]. Además, a diferencia de otros métodos de optimización, los algoritmos evolutivos son menos susceptibles a la forma o continuidad del frente de Pareto, es decir, son capaces de abordar problemas con frentes de Pareto discontinuos o cóncavos [11]. Desde su inicio, los algoritmos evolutivos fueron diseñados para abordar problemas de optimización mono-objetivo. La adaptación de un algoritmo evolutivo para resolver problemas de optimización multi-objetivo se conoce como algoritmo evolutivo multi-objetivo (*Multi-Objective Evolutionary Algorithm*, MOEA) [44].

El algoritmo NSGA-II (*Non dominanted Sorting Genetic Algorithm* II) [14] es una versión mejorada de su predecesor, el NSGA, el cual reduce la complejidad computacional e incorpora elitismo. El Algoritmo 2 describe su funcionamiento. A partir de una población de padres, P_t, se genera una población de hijos, Q_t, durante la generación t, cada una de ellas con N individuos. La unión de ambas poblaciones, $R_t = P_t \cup Q_t$, es ordenada en un conjunto de frentes, $\{F_1, \ldots, F_k\}$, mediante un procedimiento llamado *ordenamiento rápido basado en no dominancia*. Cada uno de los frentes

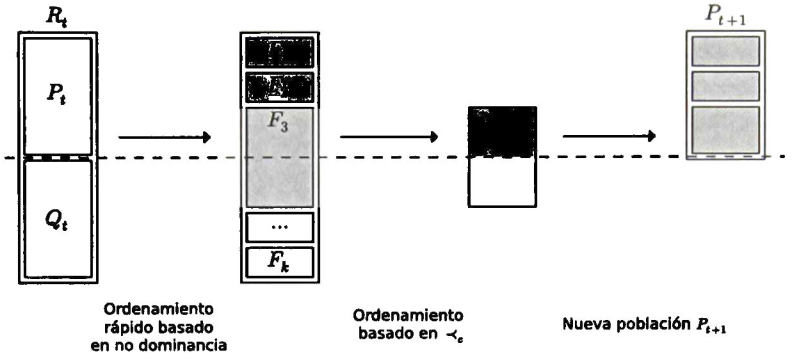


Figura 2.8: Procedimiento de NSGA-II. La unión de ambas poblaciones, $R_t = P_t \cup Q_t$, es ordenada en k frentes, $\{F_1, \ldots, F_k\}$. El frente F_3 es ordenado mediante el operador $\prec_c$ para seleccionar a los mejores individuos y así crear la población P_{t+1}.

contiene soluciones no dominadas entre sí a las cuales se les asigna un rango r. Es decir, todas las soluciones dentro de un mismo frente F_i comparten el mismo rango $r = i$, en donde $i = 1, \ldots, k$. Aquellas soluciones que se encuentren más próximas al frente de Pareto tendrán un menor rango r; es decir, entre menor sea r, mejor. Enseguida, se realiza una estimación de la densidad d a partir de la distancia entre las soluciones adyacentes a cada individuo. Entre mayor sea esta distancia, más asilada estará una solución de sus vecinos; esto es, entre mayor sea d, mejor. Con base en ambos atributos, r y d, se define el operador $\prec_c$ de la siguiente manera.

Definición 2.11. Operador $\prec_c$. Dadas dos soluciones $\{a, b\} \in X$, se dice que a es mejor que b si se cumple alguna de las siguientes condiciones:

- Si a tiene un menor rango que b, es decir, $r(a) < r(b)$.
- Si ambas soluciones tienen el mismo rango y la solución a se encuentra más aislada que b, esto es, $r(a) = r(b)$ y $d(a) > d(b)$.

La nueva población de padres, P_{t+1}, se crea a partir de los individuos con el mejor rango y densidad. Por último, la población de hijos, Q_{t+1}, se crea a partir de P_{t+1} mediante operadores genéticos. La Figura 2.8 ilustra el procedimiento descrito previamente.

3

Estado del arte

La revisión del estado del arte se ha realizado con base en las siguientes categorías:

- **Reformulación del problema original**. Diferentes algoritmos de optimización no abordan el problema original directamente, sino que resuelven una reformulación del mismo. En esta categoría, correspondiente a la Sección 3.1, se consideran específicamente dos tipos de reformulación: escalarización y multiobjetivización.

- **Uso simultáneo de diferentes formulaciones del problema original**. Actualmente existen algoritmos de optimización que reformulan un problema de optimización para luego resolverlo. Sin embargo, algunos de ellos no consideran el problema original durante el proceso de resolución. En esta categoría, correspondiente a la Sección 3.2, se describen algunas propuestas en la literatura que alternan entre ambas formulaciones: el problema original y alguna transformación del mismo.

- **Incorporación de mecanismos de selección**. Algunos algoritmos evolutivos integran mecanismos para adaptar componentes del mismo, tal como el operador de mutación a utilizar en una generación determinada. De este modo, el principal reto de un mecanismo de este tipo es elegir al mejor componente en cada etapa del proceso de búsqueda de acuerdo a un criterio de desempeño. La Sección 3.3 revisa trabajo previo referente a este tipo de mecanismos.

Finalmente, la Sección 3.4 reúne los principales aspectos del trabajo previo a modo de comparación.

3.1 Reformulación del problema original

Antes de abordar un problema de optimización, el usuario debe seleccionar un método de resolución de acuerdo a las características del problema, entre ellas, el número de objetivos, denotado por m. En la literatura existen algoritmos especializados para un número determinado de objetivos. A pesar de esto, es posible emplear métodos alternativos al problema original a través de una reformulación del mismo. A continuación se describe trabajo previo que resuelve un problema de optimización, ya sea mono-objetivo o multi-objetivo, por medio de su reformulación. Se distinguen dos tipos de reformulación:

- **Escalarización**. Esta conversión permite reformular un problema de optimización multi-objetivo como uno mono-objetivo. Consiste en combinar m funciones objetivo en uno solo empleando operaciones como suma o multiplicación. Al término de la conversión se obtiene una función escalar, la cual será resuelta mediante alguna técnica de optimización mono-objetivo.

- **Multiobjetivización**. Permite reformular un problema de optimización mono-objetivo como uno multi-objetivo. Esta conversión consiste en aumentar el número de objetivos originales, ya sea al agregar uno o varios objetivos adicionales o al descomponer el objetivo original en dos o

más objetivos. Al término de la conversión se obtiene un vector de funciones objetivo, el cual es abordado posteriormente con alguna técnica de optimización multi-objetivo.

Una de las formas más simples de transformar un problema de optimización multi-objetivo consiste en combinar todos los objetivos en uno solo a través de una función de escalarización. Murata e Ishibuchi propusieron en [40] (1995) un algoritmo genético multi-objetivo para encontrar un conjunto de soluciones no dominadas de un problema de optimización multi-objetivo por medio de una función de escalarización. Su propuesta contempla una función de aptitud, $f(\boldsymbol{x})$, generada a partir de una suma ponderada y de un vector de pesos, $\boldsymbol{w}$. La función de aptitud se define como:

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} w_i \, f_i(\boldsymbol{x}) \tag{3.1}$$

en donde el vector de pesos $\boldsymbol{w} = [w_1, \ldots, w_m]^T$ cumple con las siguientes restricciones:

$$w_i \geq 0, \; i = 1, \ldots, m \tag{3.2}$$

$$\sum_{i=1}^{m} w_i = 1 \tag{3.3}$$

De este modo se pueden formular funciones de aptitud distintas al cambiar únicamente $\boldsymbol{w}$. Gracias a esta característica es posible explorar en diferentes direcciones de búsqueda. Los autores proponen generar aleatoriamente un vector de pesos $\boldsymbol{w}$ por cada selección de padres para la cruza. Es decir, para seleccionar N parejas de padres, se requiere generar aleatoriamente N vectores $\boldsymbol{w}$ diferentes, lo cual permite explorar N direcciones de búsqueda diferentes por generación. Años más tarde, los autores extendieron su propuesta al incorporar búsqueda local [26] (1998).

En 2007, Zhang y Li [48] propusieron un algoritmo evolutivo multi-objetivo basado en descomposición, llamado MOEA/D. Su propuesta descompone un problema de optimización multi-objetivo en N subproblemas mediante un método de escalarización. Para este efecto, se define un conjunto de N vectores de pesos uniformemente espaciados, denotado por $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_N$, en donde

$\boldsymbol{\lambda}_i = [\lambda_1, \ldots, \lambda_m]^T, i \in 1, \ldots, N$. Por cada vector de pesos $\boldsymbol{\lambda}_i$ se define un subproblema escalar distinto. A partir de la distribución espacial de los vectores de pesos se establece una relación de vecindarios entre subproblemas, en donde cada vecindario se compone de T subproblemas. Cada subproblema es resuelto al emplear únicamente información de su vecindario. Los autores compararon su propuesta con respecto a MOGLS [26, 28] y NSGA-II [14] en diferentes problemas de optimización combinatoria y continua, respectivamente. A partir de su experimentación, encontraron que la calidad de las soluciones encontradas por MOEA/D fue mejor o similar a las soluciones encontradas por MOGLS y NSGA-II en la mayoría de los problemas de prueba.

Las propuestas anteriores están basadas en técnicas de escalarización, las cuales transforman un problema de optimización multi-objetivo como uno mono-objetivo. Por otro lado, es posible realizar una transformación opuesta al reformular un problema de optimización mono-objetivo como uno multi-objetivo mediante multiobjetivización. Este tipo de reformulación fue propuesta originalmente por Louis y Rawlins [36] (1993), sin embargo, el término *multiobjetivización* fue acuñado por Knowles *et al.* [32] (2001). En su trabajo seminal, Knowles *et al.* propusieron descomponer un problema de optimización mono-objetivo en subproblemas, los cuales debían estar relacionados con el problema original. De este modo, un problema de optimización mono-objetivo se convierte en uno multi-objetivo. Uno de los aspectos más atractivos de la técnica es su capacidad para remover óptimos locales, facilitando así la resolución del problema original.

A raíz de esta publicación surgieron propuestas adicionales. Jensen [29] (2004) propuso incorporar objetivos adicionales, conocidos como *helpers*, al problema original. Su experimentación reveló que el desempeño de un algoritmo genético tradicional puede ser mejorado al incorporar objetivos *helpers* parcialmente en conflicto con el problema original.

Brockhoff *et al.* [4] (2007) analizaron cómo afecta la inclusión de objetivos adicionales a un problema de optimización mono-objetivo al esfuerzo computacional requerido para generar el conjunto de soluciones óptimas de Pareto. Para su experimentación utilizaron un problema combinatorio

denominado $\text{PLATEAU}_1(\boldsymbol{x})$:

$$\text{PLATEAU}_1(\boldsymbol{x}) = \begin{cases} |\boldsymbol{x}|_0 & : \boldsymbol{x} \notin \{1^n 0^{n-i}\},\ 1 \leq i \leq n \\ n+1 & : \boldsymbol{x} \in \{1^n 0^{n-i}\},\ 1 \leq i \leq n \\ n+2 & : \boldsymbol{x} = 1^n \end{cases} \tag{3.4}$$

en donde $\boldsymbol{x}$ es una cadena de n bits, $\boldsymbol{x} \in \{0,1\}^n$, mientras que $|\boldsymbol{x}|_0$ denota el número bits igual a cero en $\boldsymbol{x}$. La solución óptima del problema corresponde a la cadena $\boldsymbol{x} = 1^n$. Los autores propusieron dos nuevos problemas a partir de la multiobjetivización de (3.4):

$$\text{PLOM}(\boldsymbol{x}) = [\text{PLATEAU}_1(\boldsymbol{x}), |\boldsymbol{x}|_1]^T \tag{3.5}$$

$$\text{PLZM}(\boldsymbol{x}) = [\text{PLATEAU}_1(\boldsymbol{x}), |\boldsymbol{x}|_0]^T \tag{3.6}$$

en donde $|\boldsymbol{x}|_0$ y $|\boldsymbol{x}|_1$ denotan el número de bits en $\boldsymbol{x}$ equivalentes a 0 y 1, respectivamente. El objetivo adicional $|\boldsymbol{x}|_1$ permite guiar la búsqueda en la *dirección correcta* hacia el óptimo $\boldsymbol{x} = 1^n$, ya que brinda mayor información sobre el problema, mientras que el objetivo adicional $|\boldsymbol{x}|_0$ guía la búsqueda en la *dirección contraria*. Es decir, la formulación (3.5) facilita la búsqueda, mientras que (3.6) no brinda beneficio alguno. Como parte de su estudio, los autores establecieron las cotas de tiempo de ejecución esperado (*expected runtime*) para los problemas original (3.4) y multiobjetivizado (3.5). Estas cotas se muestran en la Tabla 3.1, en donde se observa una reducción al utilizar la reformulación multiobjetivizada PLOM. De este modo se muestra experimental y analíticamente los beneficios que brinda la multiobjetivización cuando se realiza de forma apropiada (en este caso, al incorporar conocimiento del problema al proceso de búsqueda).

Los trabajos descritos anteriormente se han enfocado en multiobjetivizar problemas de optimización mono-objetivo. No obstante, Ishibuchi *et al.* [25] (2010) analizaron el uso de multiobjetivización para resolver un problema de dos objetivos a través de su reformulación como un

Problema	Tiempo de ejecución esperado
PLATEAU$_1$	$\Theta(n^3)$
PLOM	$O(n^2 \log n)$

Tabla 3.1: Cotas de tiempo de ejecución esperado.

problema de cuatro objetivos, lo cual representa el primer trabajo en experimentar los efectos de la multiobjetivización en un problema originalmente multi-objetivo. Su propuesta consiste en emplear objetivos *helper*, los cuales están correlacionados con los objetivos iniciales. El problema original está definido de la siguiente manera:

$$\begin{aligned} \text{Maximizar} \quad & \boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x})]^T \\ \text{sujeto a} \quad & \boldsymbol{x} \in X \end{aligned} \tag{3.7}$$

mientras que la multiobjetivización propuesta se define de la siguiente manera:

$$\begin{aligned} \text{Maximizar} \quad & \boldsymbol{g}(\boldsymbol{x}) = [g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), g_3(\boldsymbol{x}), g_4(\boldsymbol{x})]^T \\ \text{sujeto a} \quad & \boldsymbol{x} \in X \end{aligned} \tag{3.8}$$

donde los objetivos *helper* son definidos usando un $\alpha > 0$:

$$\begin{aligned} g_1(\boldsymbol{x}) &= f_1(\boldsymbol{x}) - \alpha f_2(\boldsymbol{x}) & g_3(\boldsymbol{x}) &= f_2(\boldsymbol{x}) + \alpha f_1(\boldsymbol{x}) \\ g_2(\boldsymbol{x}) &= f_1(\boldsymbol{x}) + \alpha f_2(\boldsymbol{x}) & g_4(\boldsymbol{x}) &= f_2(\boldsymbol{x}) - \alpha f_1(\boldsymbol{x}) \end{aligned} \tag{3.9}$$

Dado que α es un número real positivo pequeño, los objetivos *helper* $g_1(\boldsymbol{x})$ y $g_2(\boldsymbol{x})$ son similares al objetivo original $f_1(\boldsymbol{x})$, del mismo modo que lo son $g_3(\boldsymbol{x})$ y $g_4(\boldsymbol{x})$ con respecto a $f_2(\boldsymbol{x})$. Los cuatro objetivos generados son funciones de escalarización de los objetivos originales obtenidos a partir de vectores de pesos $(1, -\alpha), (1, \alpha), (\alpha, 1), (\alpha, -1)$. Cada vector de pesos puede ser visto como una dirección de búsqueda en el espacio objetivo definido por $f_1(\boldsymbol{x})$ y $f_2(\boldsymbol{x})$. La Figura 3.1 ilustra la relación entre los objetivos originales y los objetivos generados.

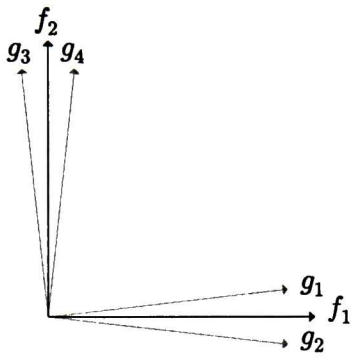


Figura 3.1: Relación entre los dos objetivos originales, f_1 y f_2, y los cuatro objetivos generados, g_1, g_2, g_3 y g_4.

Entre los resultados obtenidos al emplear (3.8), los autores encontraron que el parámetro α influye en la diversidad de las soluciones encontradas y en la convergencia hacia el frente de Pareto. Al aumentar el valor de α se incrementó la diversidad, mientras que la convergencia se deterioró ligeramente [25]. Adicionalmente, Ishibuchi *et al.* consideraron incorporar un cambio en la relación de dominancia (basado en la propuesta de Sato *et al.* [43]), y propusieron una segunda alternativa de multiobjetivización, definida de la siguiente manera:

$$\begin{aligned} \text{Maximizar} \quad & \boldsymbol{h}(\boldsymbol{x}) = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x})]^T \\ \text{sujeto a} \quad & \boldsymbol{x} \in X \end{aligned} \tag{3.10}$$

donde:

$$\begin{aligned} h_1(\boldsymbol{x}) &= f_1(\boldsymbol{x}) - \alpha f_2(\boldsymbol{x}) &= g_1(\boldsymbol{x}) \\ h_2(\boldsymbol{x}) &= f_2(\boldsymbol{x}) - \alpha f_1(\boldsymbol{x}) &= g_4(\boldsymbol{x}) \end{aligned} \tag{3.11}$$

La formulación definida en (3.10) mejora la convergencia con respecto a la propuesta original, al mismo tiempo que se cubre el frente de Pareto completo. Al igual que Brockhoff *et al.* [4], el trabajo presentado por Ishibuchi *et al.* muestra los beneficios que conlleva la multiobjetivización al elegir los objetivos adicionales adecuadamente.

El trabajo descrito anteriormente muestra que reformular el problema original, ya sea por escalarización o multiobjetivización, puede llegar a facilitar la resolución del mismo, en lugar de abordar directamente la formulación original.

3.2 Uso simultáneo de diferentes formulaciones del problema original

Algunas de las propuestas descritas anteriormente permiten resolver un problema de optimización al reformularlo mediante escalarización. Sin embargo, no abordan directamente la formulación original sino una versión diferente del mismo. Además, no se alterna entre ambas versiones del problema, original y escalarizada. A continuación se menciona trabajo previo en el cual se emplean ambas formulaciones durante el proceso evolutivo. Cabe mencionar que dicho trabajo previo es más cercano al objetivo general de nuestra investigación.

En 2006 Ishibuchi *et al.* presentaron dos artículos donde propusieron un esquema de hibridación entre un algoritmo evolutivo multi-objetivo y uno mono-objetivo. En el primero de ellos [27], describieron la idea y motivación de su esquema de hibridación, así como su evaluación en varias instancias de un problema multi-objetivo, mientras que en una segunda publicación [24], realizaron una experimentación más profunda con el fin de brindar un mayor soporte a su esquema.

La idea fundamental de su propuesta es la siguiente. Se definen dos formulaciones del problema de optimización multi-objetivo: la formulación original h_1, la cual representa al problema de optimización como tal y una formulación escalarizada h_2, la cual transforma el problema mediante suma de pesos. De este modo, se define un conjunto de formulaciones $\boldsymbol{h} = \{h_1, h_2\}$. Ambas formulaciones son utilizadas durante el proceso de optimización mediante NSGA-II. La selección de la formulación se realiza de manera probabilística. Cada vez que se desea evaluar la calidad de dos soluciones, se elige aleatoriamente una formulación $h \in \boldsymbol{h}$. Si se selecciona la formulación original, entonces se utiliza el método de comparación de NSGA-II (basado en dominancia de Pareto). Por otra parte, si

se selecciona la formulación escalarizada, entonces se utiliza la escalarización del vector objetivo de cada solución.

En [27], Ishibuchi *et al.* utilizaron el problema de la mochila (*knapsack*) [52] para $m \in \{2, 3, 4\}$ objetivos, y evaluaron dos enfoques de resolución: mono-objetivo y multi-objetivo. El enfoque de resolución mono-objetivo consistió en resolver el problema de optimización mediante suma de pesos al definir un vector de pesos $\boldsymbol{w}$ de tres formas distintas:

- SOGA-1: El vector de pesos se compone únicamente de coeficientes unitarios, de modo que $\boldsymbol{w} = [1, \ldots, 1]^T$.
- SOGA-2: Se emplea un vector $\boldsymbol{w}$ binario, de modo que existen $(2^m - 1)$ vectores, excluyendo el vector compuesto únicamente por 0. Por ejemplo, para $m = 2$, se emplean los siguientes vectores: $[0, 1]^T, [1, 0]^T, [1, 1]^T$.
- SOGA-3: El vector de pesos debe cumplir con la siguiente condición:

$$\sum_{i=1}^{m} w_i = d \tag{3.12}$$

  Para su experimentación, $d = 4$. Por ejemplo, para $m = 2$, se emplean los siguientes vectores: $[4, 0]^T, [3, 1]^T, [2, 2]^T, [1, 3]^T, [4, 0]^T$.

Por otra parte, el enfoque de resolución multi-objetivo consistió en utilizar NSGA-II. Para evaluar ambos enfoques de resolución, emplearon la suma de objetivos como criterio de comparación. A partir de su experimentación, los autores obtuvieron las siguientes observaciones en función del número de objetivos m:

- Cuando $m = 2$ el enfoque multi-objetivo supera al enfoque mono-objetivo.
- Cuando $m \in \{3, 4\}$ el enfoque mono-objetivo supera al enfoque multi-objetivo.

Es decir, ambos enfoques tienen ventajas y desventajas, lo que sugiere que la hibridación de ambos podría resultar prometedora. Como señalan Ishibuchi *et al.* en [27], la principal diferencia entre ambos enfoques radica en los mecanismos de evaluación de soluciones. Por un lado, un algoritmo evolutivo mono-objetivo emplea generalmente una función escalar para la asignación de aptitud, mientras que uno multi-objetivo (NSGA-II en este caso) usualmente se basa en dominancia de Pareto. Dado esto, el esquema de hibridación propuesto por Ishibuchi *et al.* consiste en utilizar ambos métodos de evaluación de manera probabilística. La hibridación propuesta contempla dos probabilidades, P_{PS} y P_{GU}, las cuales se aplican de la siguiente manera [1]:

- **Selección de padres**. Sea P_{PS} la probabilidad de emplear una función escalar durante la selección de padres. Cada vez que un par de padres son seleccionados a partir de la población actual, los mecanismos de evaluación mono-objetivo y multi-objetivo son utilizados con una probabilidad P_{PS} y $(1 - P_{PS})$, respectivamente.

- **Actualización de la población**. Sea P_{GU} la probabilidad de emplear una función escalar durante la actualización de la población. Cuando un individuo es seleccionado entre la población de padres e hijos para la actualización de la población, los mecanismos de evaluación mono-objetivo y multi-objetivo son utilizados con una probabilidad P_{GS} y $(1-P_{GS})$, respectivamente.

Para evaluar y comparar el desempeño del híbrido propuesto se utilizó el mismo problema de prueba para $m \in \{2, 3, 4\}$, empleando la suma de objetivos como criterio de comparación. Los resultados revelaron que su híbrido fue capaz de mejorar el desempeño de los enfoques mono-objetivo y multi-objetivo al obtener una mayor suma de objetivos, lo que indica que este tipo de hibridación es prometedora para problemas de optimización multi-objetivo. Para dar mayor soporte a su idea, Ishibuchi *et al.* realizaron un estudio posterior [24] donde se evaluó el impacto de los parámetros P_{PS} y P_{GU}. En dicho estudio obtuvieron conclusiones similares, mostrando así los beneficios de su hibridación.

[1] (*Parent Selection*, PS), (*Generation Update*, GU)

La idea propuesta por Ishibuchi *et al.* representa una base para diferentes tipos de hibridación, permitiendo explorar otras alternativas bajo el mismo enfoque.

3.3 Incorporación de mecanismos de selección

El trabajo de Ishibuchi *et al.* [24, 27] es el más cercano a nuestra investigación al utilizar probabilísticamente diferentes formulaciones de un mismo problema de optimización, denotadas por $\boldsymbol{h} = \{h_1, h_2\}$, durante el proceso búsqueda. Sin embargo, la selección de $h \in \boldsymbol{h}$ puede ser planteada como un problema de selección adaptativa con el fin de elegir al elemento más adecuado a partir de un conjunto dado. A continuación se describe trabajo previo relacionado con la selección adaptativa de operadores de mutación y funciones auxiliares.

En 2010, Lardeux y Goëffon [34] propusieron DIM (*Dynamic Islands Model*), en el cual utilizan un algoritmo evolutivo distribuido basado en islas. En su modelo propuesto se define un conjunto de n islas, denotadas por $I = \{I_1, \cdots, I_n\}$, en donde cada isla I_i se compone de una población P_i y de un operador de mutación m_i. Cada isla evoluciona independientemente empleando un operador de mutación distinto. Además, existen intercambios de individuos (migraciones) entre islas con cierta frecuencia. El control de la migración se define a partir de una matriz de transición M, en donde $M(i,j)$ representa la probabilidad que tiene un individuo de migrar de la isla i a la isla j. Entre mayor sea $M(i,j)$, habrá mayor probabilidad de migrar hacia j, lo cual implicaría que la población de la isla j se podría ver incrementada. Con esto en mente, Lardeux y Göeffon propusieron incorporar al modelo un esquema de aprendizaje por refuerzo, en donde se favoreciera (con un incremento en la matriz de transición M) a aquellos operadores (islas) que presenten el mejor desempeño durante diferente etapas del proceso de búsqueda. El desempeño de un operador m_i se ve reflejado en el tamaño de la población P_i. Entre mejor sea m_i, mayor probabilidad habrá de migrar hacia la isla I_i, lo que podría incrementar el tamaño de P_i.

Para evaluar el desempeño del modelo propuesto, Candan y Goëffon [9] emplearon un problema

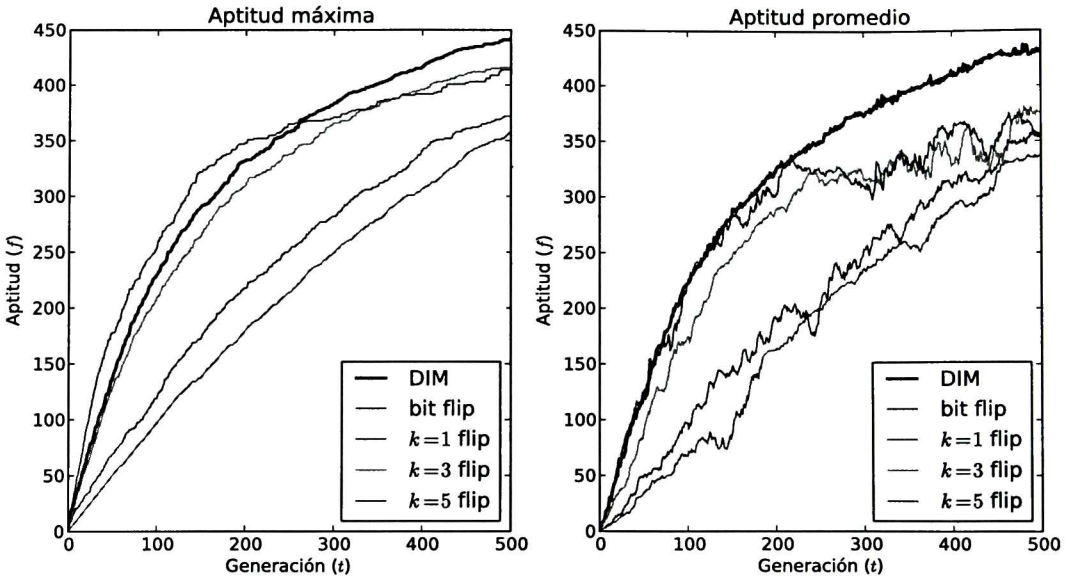


Figura 3.2: Comparación de DIM.

de optimización llamado One-Max. Este problema considera cadenas binarias de longitud n. La aptitud de una cadena $\boldsymbol{x}$, denotada como $|\boldsymbol{x}|_1$, corresponde al número de bits igual a 1. La solución óptima $\boldsymbol{x}$ corresponde a una cadena binaria de longitud n en donde $|\boldsymbol{x}|_1 = n$. A partir de los resultados obtenidos concluyen que es posible mejorar el desempeño de un algoritmo evolutivo mediante diferentes operadores de mutación en un modelo de islas.

La Figura 3.2 compara el desempeño obtenido por DIM utilizando cuatro operadores de mutación distintos con respecto a diferentes algoritmos evolutivos, cada uno con un operador de mutación fijo. Esta comparación replica parte de la experimentación realizada por Candan y Goëffon [9]. En este caso se utilizó el mismo problema de prueba One-Max con cadenas binarias de $n = 100$ bits y se usaron los mismos operadores de mutación (bit flip y k flip, $k \in \{1, 3, 5\}$). Como se puede observar, DIM es capaz de alcanzar una mayor aptitud durante el proceso evolutivo al emplear diferentes operadores de mutación en lugar de utilizar sólo uno de ellos.

En 2011, McClymont y Keedwell [38] propusieron un método de selección adaptativa basado en aprendizaje por refuerzo y cadenas de Markov, llamado MCHH (*Markov Chain Hyper-Heuristic*),

con la finalidad de elegir el mejor operador de mutación m_i de un conjunto $\boldsymbol{m} = \{m_1, \ldots, m_n\}$. Su enfoque considera una cadena de Markov completamente conectada, en donde cada estado representa un operador de mutación distinto.

Un algoritmo evolutivo (en este caso, una estrategia evolutiva) utiliza cada operador $m_i \in \boldsymbol{m}$ durante un *episodio* de $\epsilon = 5$ generaciones y calcula su desempeño $p(m_i)$. Si $p(m_i)$ es superior a un umbral γ, se beneficia al operador m_i al incrementar el peso de su transición en la cadena, de otro modo, es penalizado. Para demostrar los beneficios de su enfoque, los autores utilizaron 5 operadores de mutación distintos, donde uno de ellos es intencionalmente ineficiente y no brinda ninguna aportación al proceso de búsqueda. Con base en su experimentación, se encontró que el método propuesto fue capaz de penalizar efectivamente a aquellos operadores deficientes y favoreció sólo a los que presentaron un buen desempeño.

En 2012, Afanasyeva y Buzdalov [2] propusieron un algoritmo evolutivo controlado por aprendizaje por refuerzo, llamado EA+RL, con la finalidad de incrementar la eficiencia al resolver un problema de optimización mono-objetivo, denotado por f. Definen un conjunto de formulaciones (funciones escalares), $\boldsymbol{g} = \{g_1, \ldots, g_n\}$, en donde la formulación g_1 representa al problema de optimización mono-objetivo como tal, $g_1 = f$. mientras que el resto de las funciones g_i son *funciones auxiliares* correlacionadas con f, y podrían facilitar el proceso de optimización en diferentes etapas, aunque no existe conocimiento *a priori* sobre cuál es la mejor función en cada etapa. De este modo, un algoritmo evolutivo puede ser capaz de seleccionar la formulación $g \in \boldsymbol{g}$ más adecuada mediante aprendizaje por refuerzo. Los autores evaluaron distintos métodos de aprendizaje por refuerzo, entre ellos, aprendizaje Q con una estrategia voraz ϵ. El objetivo de utilizar aprendizaje Q en el método propuesto fue valorar el desempeño de una formulación, $g \in \boldsymbol{g}$, al asignar una recompensa, definida como $r \in \{0, 0.5, 1\}$. Si la formulación g_i evaluada en la generación t presenta un desempeño superior al de la formulación g_j en la generación previa $t - 1$, entonces es recompensada. Por otra parte, el objetivo de la estrategia voraz ϵ fue seleccionar la formulación que sería usada en la siguiente generación. Esta selección se realiza de manera voraz con una probabilidad $1 - \epsilon$. De tal forma que

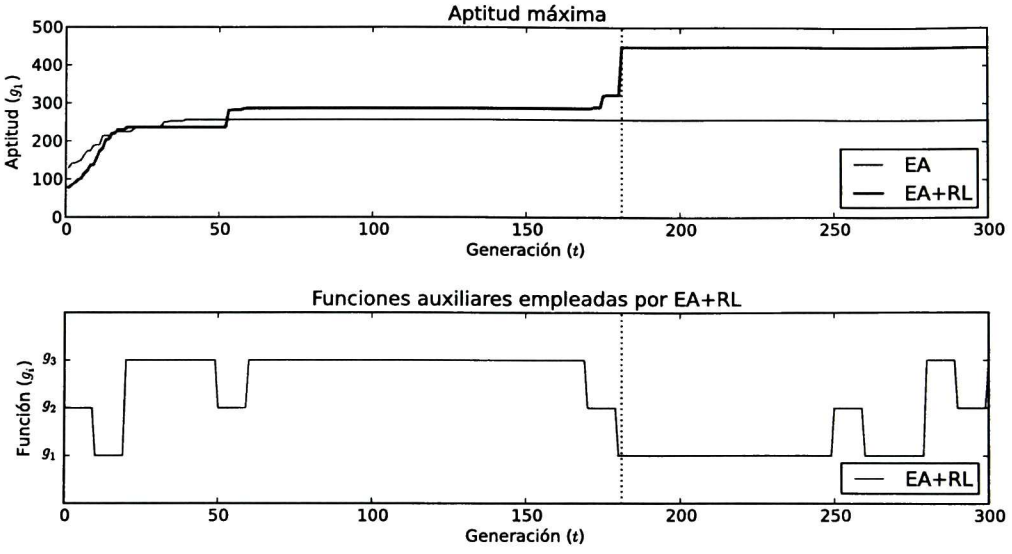


Figura 3.3: Comparación de EA+RL.

sea posible elegir a formulaciones no tan adecuadas en diferentes etapas de la búsqueda con una probabilidad ϵ.

Para evaluar el desempeño de su método, utilizaron un problema de optimización llamado H-IFF (*Hierarchical-if-and-only-if*), el cual utiliza cadenas binaras de longitud l, $\boldsymbol{x} = (x_1, \ldots, x_l)$. Este problema cuenta con dos soluciones óptimas definidas como $\boldsymbol{x}_0 = (0, \ldots, 0)$ y $\boldsymbol{x}_1 = (1, \ldots, 1)$. A partir de los resultados obtenidos, concluyen que el uso de aprendizaje por refuerzo aplicado a la selección de diferentes formulaciones permite incrementar la calidad de las soluciones encontradas durante un número fijo de generaciones.

La Figura 3.3 compara el desempeño de un algoritmo evolutivo simple (EA) con respecto a su método propuesto (EA+RL) al replicar parte de su experimentación [2]. En este caso se utilizó el problema de prueba original, H-IFF, con cadenas binarias de longitud $l = 64$ bits, cuya solución óptima $\boldsymbol{x}$ tiene una aptitud $g_1(\boldsymbol{x}) = 448$. Además, la selección de g se realizó cada 5 generaciones. Como se observa en la parte superior, EA+RL logró alcanzar la solución óptima en la generación $t = 181$ (línea punteada). En la parte inferior se muestra cómo el método EA+RL utilizó las tres funciones en

Referencia	Trabajo	Reformulación del problema original	Uso simultáneo de diferentes formulaciones	Incorporación de mecanismos de selección
[26]	Ishibuchi, Murata (1998)	✓		
[48]	Zhang, Li (2007)	✓		
[32]	Knowles *et al.* (2001)	✓		
[29]	Jensen (2004)	✓		
[25]	Ishibuchi *et al.* (2010)	✓		
[24, 27]	Ishibuchi *et al.* (2006)	✓	✓	*
[34]	Lardeux y Goëffon (2010)			✓
[38]	McClymont y Keedwell (2011)			✓
[2]	Afanasyeva y Buzdalov (2012)			✓

Tabla 3.2: Comparación de trabajo previo.

diferentes etapas del proceso de búsqueda. Es interesante notar que, antes de la generación $t = 181$, la función auxiliar g_3 fue empleada por más tiempo, incluso más que la misma función g_1, la cual representa al problema de optimización original.

3.4 Discusión

La revisión del estado del arte ha sido abordada alrededor de las siguientes tres categorías: reformulación del problema original, uso de diferentes formulaciones del problema original y la incorporación de un mecanismo de selección. Con base en esta clasificación, la Tabla 3.2 resume las principales contribuciones encontradas en el trabajo previo, en donde ✓denota que el trabajo realiza una aportación en la categoría correspondiente, mientras que $*$ denota una característica que se puede mejorar. La mayoría de los artículos descritos realizan una reformulación del problema antes de resolverlo, como se muestra en la tercera columna. Además, existe relativamente poco trabajo que aborde al menos dos formulaciones del problema de optimización. Más aún, el trabajo existente no incorpora un mecanismo de selección que permita realizar una mejor elección, motivo por el cual se marca como una característica deseable a mejorar.

La Tabla 3.3 concentra el trabajo previo descrito relacionado a algunos mecanismos de selección. Como se puede ver, la mayoría de ellos emplea algún indicador de desempeño con la finalidad de realizar una mejor selección. En el modelo dinámico de islas se utiliza la aptitud de cada isla (es decir, de cada operador de mutación $m \in \boldsymbol{m}$). Si la aptitud presenta un incremento luego de aplicar

Tipo de mecanismo	Conjunto de selección	Indicador de desempeño
Probabilístico	Formulaciones, $\boldsymbol{h}$	Ninguno
Modelo dinámico de islas	Operadores de mutación, $\boldsymbol{m}$	Aptitud de cada isla
Aprendizaje Q + voraz ϵ	Funciones escalares, $\boldsymbol{g}$	Aptitud
Cadenas de Markov	Operadores de mutación, $\boldsymbol{m}$	Dominancia entre hijos y padres

Tabla 3.3: Revisión de trabajo previo de acuerdo al mecanismo de selección empleado.

un operador de mutación, se incrementa la probabilidad de migración hacia dicha isla, aumentado así el uso de este operador. En EA+RL, si la formulación g_i evaluada en la generación t presenta un desempeño superior al de la formulación g_j en la generación previa $t-1$, entonces es recompensada. Por último, MCHH mide el desempeño mediante una métrica basada en dominancia entre padres e hijos. La revisión de los artículos mostrados en la Tabla 3.3 propició la identificación de la siguientes áreas de oportunidad:

- Explorar si el modelo de islas dinámico se puede orientar hacia la selección adaptativa de una formulación $h \in \boldsymbol{h}$. Además, evaluar la posibilidad de incorporar alguna etapa de filtrado y reemplazo entre la población de cada isla, con la finalidad de filtrar sólo las mejores soluciones y enviarlas a todas las islas por igual.

- Evaluar la posibilidad de utilizar aprendizaje por refuerzo para guiar la selección de la formulación más adecuada en un problema de optimización multi-objetivo.

- Analizar diferentes métricas de desempeño que permita determinar cuál es la formulación más adecuada en cada etapa de la búsqueda en un problema de optimización multi-objetivo.

4

Algoritmo propuesto basado en un modelo de islas

Este capítulo presenta el algoritmo propuesto basado en un modelo de islas, capaz de evaluar diferentes formulaciones de un problema de optimización multi-objetivo. La Sección 4.1 describe la estructura y división de la población en diferentes islas. La Sección 4.2 detalla el intercambio de individuos entre subpoblaciones mediante no dominancia. Cada isla evoluciona mediante una formulación h_i distinta. La Sección 4.3 describe el procedimiento empleado para inferir el desempeño de h_i y así identificar a la formulación más adecuada durante el proceso de búsqueda. La Sección 4.4 describe la selección entre islas, fomentando la elección de aquellas islas que brinden una mejora a la búsqueda. Una vez definidos los elementos de nuestro método, el algoritmo propuesto se formaliza en la Sección 4.5.

4.1 Población estructurada

La población total P es dividida en N diferentes grupos o islas, denotado por $P = \{P_1, \ldots, P_N\}$. El conjunto de islas es denotado por $I = \{I_1, \ldots, I_N\}$. Cada isla se compone de una subpoblación P_i con μ individuos y de una formulación h_i; por lo tanto, cada isla se define como una tupla $I_i = (P_i, h_i)$, donde $i = 1, \ldots, N$. El conjunto de formulaciones $\boldsymbol{h} = \{h_1, \ldots, h_N\}$ define las diferentes formulaciones del problema de optimización multi-objetivo que serán empleadas en cada isla durante el proceso evolutivo. De este modo, $\boldsymbol{h}$ se define de la siguiente manera:

$$\begin{aligned} h_1 &= \boldsymbol{f} \\ h_i &= g_i, \quad i = 2, \ldots, N \end{aligned} \tag{4.1}$$

donde $\boldsymbol{f}$ denota la formulación original del problema y g_i denota una formulación alternativa del mismo problema. De acuerdo con la definición (4.1), la isla I_1 empleará la formulación h_1 durante el proceso evolutivo, mientras que las islas I_i emplearán las formulaciones alternativas g_i, respectivamente.

4.2 Intercambio de individuos

El intercambio de individuos se realiza en dos etapas: filtrado y reemplazo. En la primera etapa, la población P_{best} es creada a partir de los η mejores individuos luego de ordenar a la población total P mediante el ordenamiento rápido basado en no dominancia [14], en donde $\eta \leq \mu$. En la segunda etapa, se eligen η individuos aleatoriamente en cada isla y se reemplazan con los individuos en P_{best}. De este modo, cada isla se beneficia del esfuerzo realizado por las demás islas al introducir a los mejores individuos encontrados.

Considere el siguiente ejemplo. La Figura 4.1 ilustra la etapa de filtrado sobre un conjunto de $N = 4$ islas, $\mu = 4$ individuos por isla y $\eta = 2$ individuos por intercambio. En este ejemplo, la

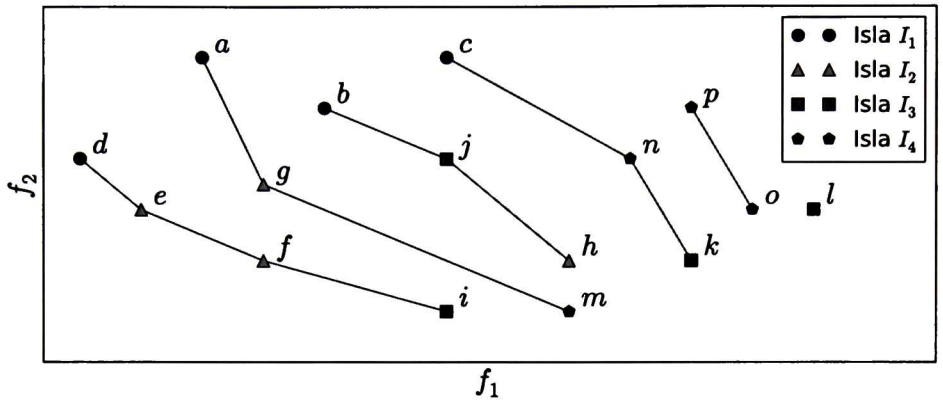


Figura 4.1: Etapa de filtrado. La población total es ordenada en frentes. La población P_{best} se crea a partir de los mejores η individuos. En este caso, $\mu = 4$, $\eta = 2$ y $P_{best} = \{d, i\}$.

población P es ordenada como un conjunto de frentes, $\{F_1, \ldots, F_6\}$, definidos por (4.2), donde F_1 y F_6 contienen a los mejores y a los peores individuos de P, respectivamente. La población P_{best} es creada a partir de los mejores $\eta = 2$ individuos de F_1, en este caso, las soluciones d y e. Por último, se eligen aleatoriamente η individuos en cada isla y se reemplazan con P_{best}.

$$\begin{array}{lllllllll} F_1 & = & \{d,e,f,i\} & F_3 & = & \{b,j,h\} & F_5 & = & \{p,o\} \\ F_2 & = & \{a,g,m\} & F_4 & = & \{c,n,k\} & F_6 & = & \{l\} \end{array} \tag{4.2}$$

4.3 Evaluación del desempeño de cada isla

Con el fin de identificar la formulación h_i más adecuada en diferentes etapas del proceso de búsqueda, se propone un indicador de desempeño r_i basado en dos criterios: contribución c_i y número de generaciones s_i, en donde c_i denota el número total de soluciones en el mejor frente F_1 que pertenecen a la isla I_i luego de evolucionar por s_i generaciones:

$$c_i = |\{ \boldsymbol{x} : \boldsymbol{x} \in I_i \wedge \boldsymbol{x} \in F_1 \}| \quad i = 1, \ldots, N \tag{4.3}$$

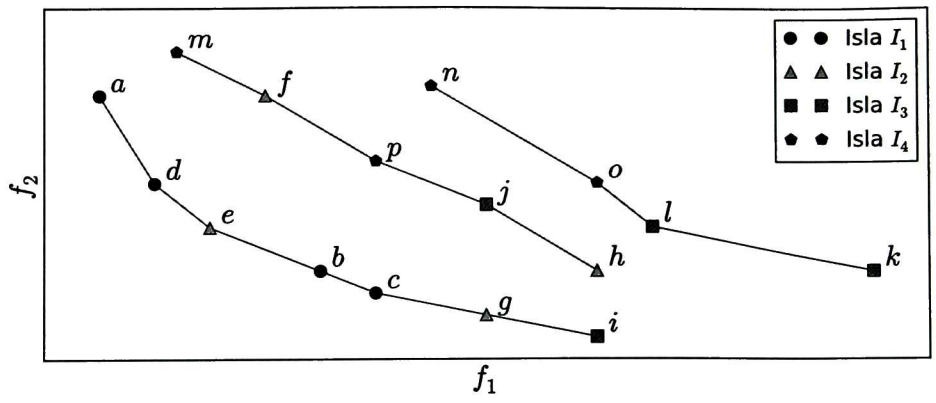


Figura 4.2: Cada isla aporta un número distinto de individuos al mejor frente. Su contribución se define por $\boldsymbol{c} = (4, 2, 1, 0)$.

La Figura 4.2 muestra un conjunto de $N = 4$ islas y $\mu = 4$ individuos por isla. Cada isla ha evolucionado independientemente durante 5 generaciones. El mejor frente F_1 está compuesto por $4, 2, 1$ y 0 individuos pertenecientes a las islas I_1, I_2, I_3 e I_4, respectivamente. Note que cada isla aporta una contribución distinta c_i, es decir, un número diferente de individuos al mejor frente. Por lo tanto, se define un vector de contribución $\boldsymbol{c} = (c_1, \ldots, c_N)$, donde c_i representa el número total de soluciones en F_1 que pertenecen a la isla I_i. De la misma forma, se define un vector de generaciones $\boldsymbol{s} = (s_1, \ldots, s_N)$, donde s_i representa el número total de generaciones usadas por la isla I_i durante el proceso evolutivo. En este ejemplo, dichos vectores se definen como $\boldsymbol{c} = (4, 2, 1, 0)$ y $\boldsymbol{s} = (5, 5, 5, 5)$. Finalmente, es posible obtener una medida aproximada del desempeño de la isla I_i y, por lo tanto, de la formulación h_i, utilizando la siguiente razón de desempeño: $r_i = c_i/s_i$. En este ejemplo, se definen las siguientes relaciones:

$$r_1 = 4/5 \quad r_2 = 2/5 \quad r_3 = 1/5 \quad r_4 = 0/5 \tag{4.4}$$

Por lo tanto, se puede concluir que la formulación h_1 usada por la isla I_1 es la mejor formulación debido a que provee la mejor razón de desempeño r_1, es decir, aporta un mayor número de individuos al mejor frente usando el mismo número de generaciones que el resto de las islas.

Algoritmo 3 Actualización de $\boldsymbol{p}$

```
1: function UPDATE(c, s)
2:     α ← 1, β ← ρN
3:     p ← 0, ∀p ∈ p
4:     for c_i ∈ c do
5:         if c_i = 0 then                ▷ La isla I_i no provee soluciones al mejor frente
6:             c_i ← α                    ▷ Valor por defecto para c_i
7:         end if
8:     end for
9:     for s_i ∈ s do
10:        if s_i = 0 then                ▷ La isla I_i no ha sido evaluada
11:            s_i ← β                    ▷ Valor por defecto para s_i
12:        end if
13:    end for
14:    for r_i ∈ r do
15:        r_i ← c_i/s_i                  ▷ Razón de desempeño r_i
16:        r_i ← r_i/∑_{r∈r} r            ▷ Normalización de r_i
17:    end for
18:    p_i ← r_i, i = 1, ..., N           ▷ Actualización de p_i
19:    return p
20: end function
```

4.4 Selección de islas

En nuestra propuesta, una isla I_i es seleccionada aleatoriamente para evolucionar por ρ generaciones con una probabilidad p_i, $i = 1, \ldots, N$. El valor de p_i es calculado a partir la razón de desempeño r_i, tal como se describe en el Algoritmo 3. Dado una razón de desempeño r_i, existen dos posibles escenarios:

- $c_i = 0$: la isla I_i no ha contribuido con alguna solución al mejor frente, por lo que dicha isla no tendrá probabilidad de ser seleccionada en el futuro. Para evitar esta posibilidad, un valor por defecto α es asignado a c_i. De esta manera, dicha isla aún tiene oportunidad de ser seleccionada.

- $s_i = 0$: la isla I_i no ha evolucionado. Dado que la selección entre islas se realiza

probabilísticamente, existe la posibilidad de que alguna isla no tenga la oportunidad de evolucionar. Para evitar una operación no válida, $r_i = c_i/0$, un valor por defecto β es asignado a s_i.

Una vez que ambas condiciones se han verificado (líneas 5 y 10), el vector $\boldsymbol{r}$ es normalizado. Finalmente, el valor de p_i es actualizado a partir de r_i.

4.5 Algoritmo propuesto

Nuestra propuesta, descrita en el Algoritmo 4, funciona de la siguiente manera. Se crea una subpoblación inicial con μ individuos, denotada por P_{base}. La población total P se define a partir de P_{base}, de esta forma, cada isla parte de la misma subpoblación inicial. El proceso de búsqueda se divide en T diferentes etapas (línea 5), en donde $t \in \{1, \ldots, T\}$. Por cada etapa t, N islas son seleccionadas aleatoriamente para evolucionar. En la primera etapa, $t = 1$, se realiza una selección determinista (línea 9), de modo que todas las islas son evaluadas al menos una vez por la función EVOLVE. Durante las etapas siguientes, $t > 1$, se lleva a cabo una selección aleatoria (línea 11) basada en ruleta [17] en función de $\boldsymbol{p}$. Cada valor $p \in \boldsymbol{p}$ define la probabilidad de seleccionar a la isla I_i para evolucionar. El proceso de optimización es realizado por la función EVOLVE, la cual evoluciona una subpoblación P_i por medio de un algoritmo evolutivo A utilizando la formulación h_i por ρ generaciones. La subpoblación P_i es actualizada mediante la población P_i' y el contador s_i es incrementado ρ generaciones al terminar el proceso de optimización. Este procedimiento es realizado N veces por cada etapa t. El vector $\boldsymbol{c}$ es calculado por la función CONTRIBUTION con base en las soluciones que aporta cada isla al mejor frente (Sección 4.3). El vector de probabilidades $\boldsymbol{p}$ es actualizado a partir de los vectores $\boldsymbol{c}$ y $\boldsymbol{s}$ por la función UPDATE (Algoritmo 3). Finalmente, cada subpoblación P_i es actualizada de acuerdo a las etapas de filtrado y reemplazo (Sección 4.2).

Algoritmo 4 Algoritmo propuesto basado en un modelo dinámico de islas

1: $P_{base} \leftarrow \text{INITPOP}(\mu)$
2: $P \leftarrow \{P_{base}, \ldots, P_{base}\}$
3: $p \leftarrow 1/N,\ \forall p \in \boldsymbol{p}$ ▷ Vector de probabilidad de selección
4: $c \leftarrow 0,\ \forall c \in \boldsymbol{c}$ ▷ Vector de contribución
5: **for** $t \in \{1, \ldots, T\}$ **do**
6: $s \leftarrow 0,\ \forall s \in \boldsymbol{s}$ ▷ Contador de generaciones
7: **for** $n \in \{1, \ldots, N\}$ **do**
8: **if** $t = 1$ **then**
9: $i \leftarrow k$ ▷ Selección determinista
10: **else**
11: $i \leftarrow \text{ROULLETE}(\boldsymbol{p})$ ▷ Selección aleatoria
12: **end if**
13: $P_i' \leftarrow \text{EVOLVE}(A, P_i, h_i, \rho)$ ▷ Evaluación de la formulación h_i
14: $P_i \leftarrow P_i'$
15: $s_i \leftarrow s_i + \rho$
16: **end for**
17: $\boldsymbol{c} \leftarrow \text{CONTRIBUTION}(P)$ ▷ Contribución de cada isla
18: $\boldsymbol{p} \leftarrow \text{UPDATE}(\boldsymbol{c}, \boldsymbol{s})$
19: $P_{best} \leftarrow \text{FILTER}(P, \eta)$ ▷ Etapa de filtrado
20: $P_i \leftarrow \text{REPLACE}(P_i, P_{best}),\ i = 1, \ldots, N$ ▷ Etapa de reemplazo
21: **end for**

5

Algoritmo propuesto basado en aprendizaje por refuerzo

Este capítulo presenta el algoritmo propuesto basado en aprendizaje por refuerzo, capaz de evaluar diferentes formulaciones mediante un esquema de recompensa. De este modo, si una formulación h_i brinda una mejora al proceso de búsqueda, se otorga una recompensa positiva; en otro caso, se penaliza. La Sección 5.1 describe conceptos propios del aprendizaje por refuerzo, así como el algoritmo tomado como base para nuestra propuesta: aprendizaje Q y estrategia voraz ϵ. La Sección 5.2 define el método propuesto, en donde una formulación h_i es empleada durante ρ generaciones antes de evaluar su desempeño. Finalmente, la Sección 5.3 muestra los tres esquemas de recompensa empleados para valorar el beneficio de una formulación determinada.

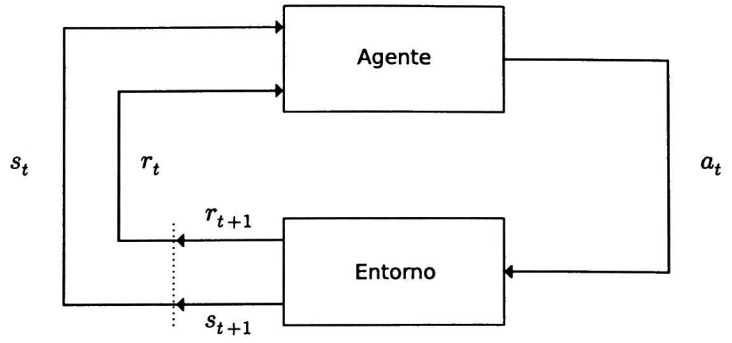


Figura 5.1: Interacción entre agente y entorno en aprendizaje por refuerzo.

5.1 Aprendizaje por refuerzo

El aprendizaje por refuerzo es una rama del aprendizaje máquina y permite que un *agente* aprenda a comportarse en un *entorno*, en donde el único medio de retroalimentación es una *recompensa*. El objetivo del agente es realizar acciones que permitan maximizar la recompensa obtenida a largo plazo [46].

El agente interactúa con el entorno en intervalos discretos de tiempo, $t = 1, 2, \ldots, T$ En cada intervalo de tiempo t, el agente recibe una representación del *estado* del entorno, $s_t \in S$, donde S es el conjunto de estados posibles. Con base en el estado s_t, el agente selecciona una *acción* $a_t \in A(s)$, donde $A(s)$ es el conjunto de acciones disponibles en el estado s_t. En el siguiente intervalo de tiempo, $t + 1$, el agente recibe una recompensa escalar, r_{t+1}, luego de efectuar la acción a. Por último, el agente se traslada a un nuevo estado s_{t+1}. La Figura 5.1 muestra la interacción entre agente y entorno.

En cada intervalo de tiempo t, el agente emplea una *política*, denotada por $\pi_t : S \rightarrow A$, para seleccionar una acción a_t a partir del estado s_t, en donde $a_t = \pi_t(s_t)$. Asimismo, el cálculo de la recompensa r_{t+1} se realiza mediante una función de recompensa, $R : S \times A \rightarrow \mathbb{R}$, en donde $r_{t+1} = R(s_t, a_t)$. Los métodos de aprendizaje por refuerzo especifican cómo el agente cambia su política como resultado de su experiencia con el fin de maximizar su recompensa a largo plazo [45].

Algoritmo 5 Política voraz ϵ

```
1: function ε-GREEDY(s_t, Q, ε)
2:     if FLIP(ε) then
3:         Seleccionar a_t aleatoriamente a partir de A(s_t)        ▷ Exploración
4:     else
5:         a_t ← π(s_t)                                             ▷ Explotación
6:     end if
7:     return a_t
8: end function
```

El uso de métodos de aprendizaje por refuerzo para mejorar el desempeño de un algoritmo evolutivo ha sido explorado previamente en la literatura [18, 1, 2, 7, 6, 30]. En 2012, Afanasyeva y Buzdalov [2] estudiaron un problema similar al abordado en esta tesis, pero diseñado para optimización mono-objetivo; en lugar de utilizar un conjunto con diferentes formulaciones de un problema multi-objetivo $\boldsymbol{f}$, los autores emplearon un conjunto de k *funciones auxiliares* de un problema de optimización mono-objetivo f, denotado por $\boldsymbol{g} = \{g_1, \ldots, g_k\}$, en donde $g_1 = f$. A partir del conjunto $\boldsymbol{g}$, utilizaron un método de aprendizaje por refuerzo, conocido como aprendizaje Q con una política voraz ϵ, con el fin de seleccionar la mejor función $g \in \boldsymbol{g}$ durante cada etapa del proceso de búsqueda. Dada la similitud con el problema abordado en esta tesis, se ha adoptado este método de aprendizaje, descrito a continuación.

El aprendizaje Q emplea una matriz Q que almacena una aproximación de la recompensa obtenida al realizar una acción a_t desde un estado s_t, denotada por $Q(s_t, a_t)$. Un agente puede optar por realizar sólo acciones óptimas de acuerdo con la matriz Q, es decir, aquellas acciones con la máxima recompensa. Sin embargo, es importante que sea capaz de explorar diferentes acciones que no sean necesariamente las mejores. La política voraz ϵ, descrita en el Algoritmo 5, brinda un compromiso entre explotación y exploración, en donde un agente realiza la mejor acción $a_t^* = \pi(s_t)$ a partir de un estado s_t con una probabilidad $1 - \epsilon$:

$$\pi(s_t) = \arg\max_{a^*} Q(s_t, a^*) \tag{5.1}$$

Algoritmo 6 Aprendizaje Q

1: Inicializar matriz $Q(s,a) \leftarrow 0$, $\forall s \in S, a \in A(s)$
2: **for** cada episodio **do**
3: Definir estado inicial s_t aleatoriamente
4: **while** s_t no sea el estado final **do**
5: Seleccionar una acción a_t a partir de s_t mediante la política voraz ϵ usando $Q(s_t, a_t)$
6: Realizar la acción a_t, calcular recompensa $R(s_t, a_t)$ y definir el siguiente estado $s_{t+1} \leftarrow a$
7: Actualizar $Q(s_t, a_t)$ mediante la siguiente regla:
8: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right]$
9: $s_t \leftarrow s_{t+1}$
10: **end while**
11: **end for**

o bien, puede realizar una acción a_t seleccionada de manera aleatoria con una probabilidad ϵ. De esta manera, el agente es capaz de explorar otras acciones que, de otro modo, no conocería.

El funcionamiento del aprendizaje Q se muestra en el Algoritmo 6. La actualización de la matriz Q se realiza por *episodios*. En cada episodio se elige aleatoriamente un estado inicial s_t. Enseguida, se elige una acción a_t mediante la política voraz ϵ; de este modo, es posible realizar una selección aleatoria o una selección voraz con una probabilidad ϵ y $1 - \epsilon$, respectivamente. El siguiente estado, s_{t+1}, se define a partir de a_t, de modo que $s_{t+1} = a_t$. Luego de definir un estado s_t y una acción a_t, se actualiza la matriz Q mediante la regla definida por (5.2) [45]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right] \tag{5.2}$$

donde α es un factor de aprendizaje y γ es un factor de descuento. La actualización de $Q(s_t, a_t)$ se basa en la recompensa inmediata y en la recompensa que se puede adquirir en un estado futuro. La recompensa inmediata está definida por $R(s_t, a_t)$ y representa el beneficio que se otorga al agente al realizar la acción a_t desde el estado s_t. Por otro lado, la recompensa futura considera la máxima ganancia que se podría alcanzar al visitar los posibles estados a' a partir de s_{t+1}, y se define por el término $\max_{a'} Q(s_{t+1}, a')$ en (5.2). De este modo, el aprendizaje Q evalúa tanto la recompensa inmediata (matriz R) como el beneficio futuro basado en experiencia previa (matriz Q).

Algoritmo 7 Algoritmo propuesto basado en aprendizaje por refuerzo

1: Inicializar matrices $Q(s,a) \leftarrow 0$ y $R(s,a) \leftarrow 0$, $\forall s \in S, a \in A(s)$
2: Definir población inicial P_t, $t \leftarrow 0$
3: Definir estado inicial s_t aleatoriamente
4: **for** cada episodio **do**
5: $a_t \leftarrow \epsilon$-GREEDY(s_t, Q, ϵ)
6: $s_{t+1} \leftarrow a_t$
7: $h \leftarrow \boldsymbol{h}_{s_t}$
8: $P_{t+1} \leftarrow$ EVOLVE(A, P_t, h, ρ) ▷ Evaluación de la formulación h_t
9: $R(s_t, a_t) \leftarrow$ GETREWARD(P_t, P_{t+1}) ▷ Actualización de R
10: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[R(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right]$ ▷ Actualización de Q
11: $P_t \leftarrow P_{t+1}$
12: $s_t \leftarrow s_{t+1}$
13: **end for**

5.2 Algoritmo propuesto

El aprendizaje Q le indica a un agente cómo comportarse dentro de un entorno, de manera que las acciones que realice maximicen la recompensa a largo plazo. Es posible trasladar esta idea a un dominio distinto, en donde cada estado represente una formulación distinta y el entorno sea un algoritmo evolutivo. De este modo, sería posible identificar la mejor formulación en cada etapa del proceso de búsqueda mediante aprendizaje Q. El algoritmo propuesto permite evaluar diferentes formulaciones de un problema de optimización multi-objetivo mediante un método de aprendizaje por refuerzo. La selección entre distintas formulaciones se realiza bajo un enfoque de recompensa. Si una formulación $h \in \boldsymbol{h}$ brinda un beneficio a la búsqueda, se recompensa; en otro caso, se penaliza. El proceso de búsqueda es divido en episodios de ρ generaciones. En cada episodio se elige una formulación diferente. Cada estado s representa una formulación distinta $h \in \boldsymbol{h}$, mientras que tomar una acción a significa seleccionar una formulación del conjunto $\boldsymbol{h}$. Una vez seleccionada una formulación, se evalúa por un algoritmo evolutivo durante ρ generaciones.

El método propuesto se describe en el Algoritmo 7. Inicialmente, se definen Q y R como matrices vacías y se crea una población P_t. El estado inicial s_t se elige aleatoriamente; de este modo, la primera

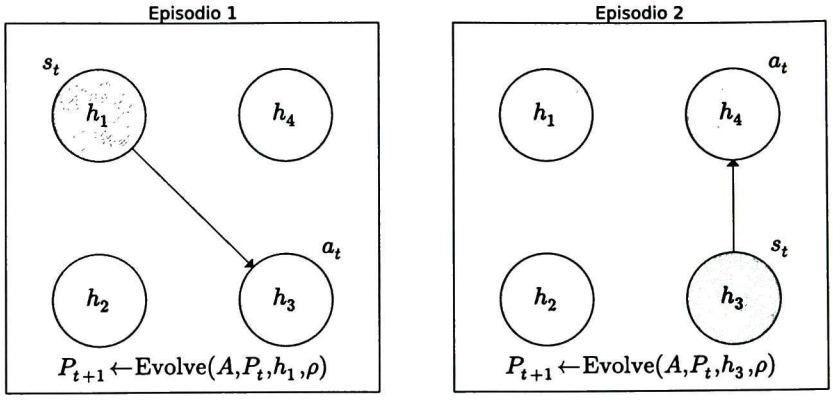


Figura 5.2: Selección de h. (Izquierda) La formulación h es evaluada en el primer episodio; la formulación h_3 es elegida como la siguiente acción a realizar. (Derecha) La formulación h_3 es evaluada en el segundo episodio y se elige h_4 como la siguiente acción.

formulación h empleada por el método propuesto se elige al azar entre el conjunto de formulaciones posibles, $\boldsymbol{h}$ (línea 7), a partir de s_t. Enseguida, se elige una acción a_t por medio de ϵ-GREEDY a partir de s_t y Q. La población P_t evoluciona durante ρ generaciones empleando la formulación h mediante un algoritmo evolutivo A. Así, P_t y P_{t+1} representan a la población antes y después de evaluar la formulación h_t (línea 8). La recompensa $R(s_t, a_t)$ se calcula a partir de ambas poblaciones, P_t y P_{t+1}, mediante el método GETREWARD, descrito más adelante, y se actualiza $Q(s_t, a_t)$ a partir de la regla definida por (5.2) [45]. Por último, se actualiza la población P_t y el estado s_t a partir de P_{t+1} y s_{t+1}, respectivamente. El procedimiento anterior se repite hasta cumplir la condición de paro, en este caso, un número de episodios predefinido. Los estados y acciones son representados mediante enteros positivos. Así, si un agente se encuentra en el estado $s_t = 1$, se evaluará entonces la formulación h_1.

La Figura 5.2 ilustra cómo se efectúa la selección de h durante dos episodios. En el primer episodio, el estado s_t es elegido aleatoriamente, $s_t = 1$, mientras que la acción a_t es elegida mediante ϵ-GREEDY, $a_t = 3$. El siguiente estado, s_{t+1}, se define a partir de a_t, por lo que la formulación a evaluar en el segundo episodio sera h_3. Enseguida, se emplea h_1 durante el proceso evolutivo para generar P_{t+1}. Finalmente, se actualizan las matrices Q y R. El segundo episodio inicia en $s_t = 3$,

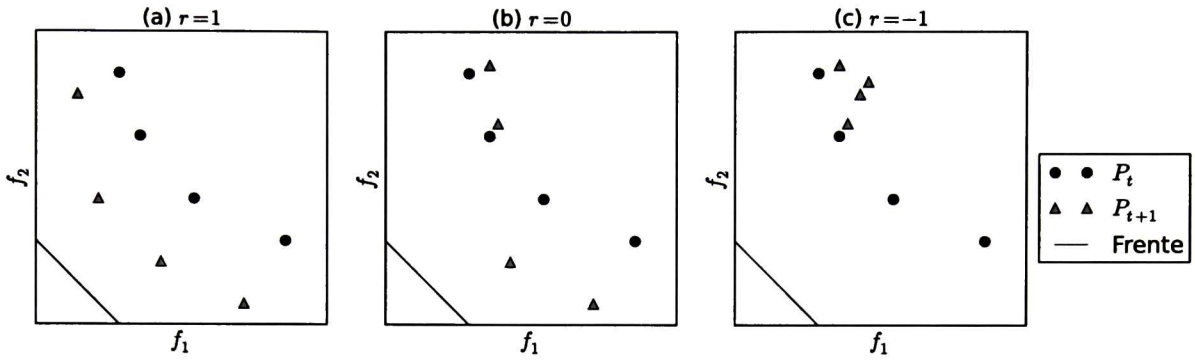


Figura 5.3: Asignación de la recompensa r de acuerdo al desempeño de la formulación h. Los conjuntos P_t y P_{t+1} representan a la población antes y después de evaluar h, respectivamente. (a) h es adecuada. (b) h es relativamente adecuada. (c) h es inadecuada.

mientras que la acción a_t es elegida mediante ϵ-GREEDY, $a_t = 4$. Enseguida, se emplea h_3 durante el proceso evolutivo para generar P_{t+1}. Por último, se actualizan Q y R, y se define el siguiente estado, $s_t = 4$.

5.3 Evaluación del desempeño de cada formulación

Un aspecto importante en un método de aprendizaje por refuerzo es la definición de la recompensa otorgada a un agente al realizar una acción determinada. En el método propuesto, la recompensa r se define respecto a dos poblaciones, P_t y P_{t+1}, las cuales representan a la población antes y después de evaluar la formulación h. La recompensa se restringe a $r \in \{-1, 0, +1\}$, de manera similar a [2].

La Figura 5.3 ilustra la definición de r en tres casos hipotéticos: a, b y c. En cada uno se muestra la distribución de las poblaciones P_t y P_{t+1} en el espacio objetivo. En el primer caso (a), la formulación h es adecuada, ya que la población P_{t+1} se encuentra más cerca del frente de Pareto que P_t, por lo que se otorga una recompensa positiva. En el segundo caso (b), la formulación h es relativamente adecuada, ya que, a pesar de que algunos individuos en P_{t+1} se alejan del frente de Pareto, otros se aproximan hacia el mismo, por lo que se otorga una recompensa neutral. En el último caso (c), la formulación h es inadecuada, debido a que la población P_{t+1} se encuentra más lejos del frente de

Algoritmo 8 Asignación de recompensa basado en la métrica C

```
1: function SetCoverageReward(P_t, P_{t+1})
2:     c_t ← C(P_t, P_{t+1})
3:     c_{t+1} ← C(P_{t+1}, P_t)
4:     if β · c_{t+1} ≥ c_t then
5:         r ← 1                                  ▷ Recompensa positiva
6:     else
7:         if c_{t+1} ≥ c_t then
8:             r ← 0                              ▷ Recompensa neutral
9:         else
10:            r ← −1                             ▷ Recompensa negativa
11:        end if
12:    end if
13:    return r
14: end function
```

Pareto que en un inicio, es decir, antes de evaluar h, por lo que se otorga una recompensa negativa. A continuación se describen tres métodos para la asignación de recompensa. Cada método se divide en tres secciones, una por cada tipo de recompensa: positiva, neutral y negativa.

El primer método, descrito en el Algoritmo 8, está basado en la métrica C [11] con el fin de promover la convergencia hacia el frente de Pareto. Dadas dos aproximaciones del frente de Pareto denotadas por A y B, $C(A, B)$ define el porcentaje de soluciones en B que son dominadas por al menos una solución en A:

$$C(A, B) = \frac{|\{v \in B \mid \exists u \in A : u \text{ domina a } v\}|}{|B|} \tag{5.3}$$

$C(A, B)$ no es necesariamente igual a $1 - C(B, A)$. $C(A, B) = 1$ indica que todas las soluciones en B son dominadas por algunas soluciones en A, mientras que $C(A, B) = 0$ implica que no existe solución en B que sea dominada por alguna solución en A [48]. Es decir, $C(A, B) > C(B, A)$ indica que A se encuentra más próxima al frente de Pareto que B. De acuerdo al Algoritmo 8, las variables $\{c_t, c_{t+1}\} \in [0, 1]$ denotan el valor de la métrica C con respecto a las poblaciones P_t y P_{t+1}, es decir, $C(P_t, P_{t+1})$ y $C(P_{t+1}, P_t)$, respectivamente. Dado que c_t y c_{t+1} son valores reales, es poco

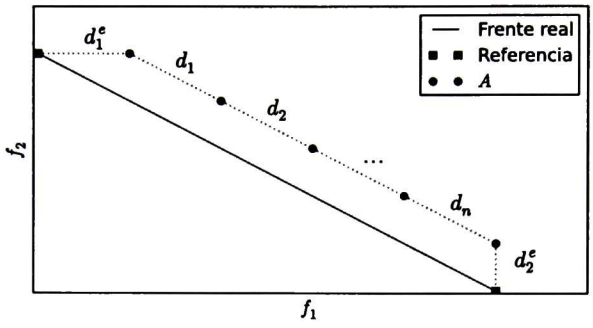


Figura 5.4: Parámetros de la métrica Δ.

probable que c_{t+1} sea igual a c_t, por tal motivo, se emplea un factor β en el cálculo de la recompensa r y así dividir el método en tres casos. En el primer caso, si $\beta \cdot c_{t+1} \geq c_t$, entonces se asigna una recompensa positiva, dado que la población P_{t+1} se encuentra más próxima al frente de Pareto que P_t, de acuerdo a la métrica C, lo cual implica que la formulación h favorece el proceso de búsqueda. En el segundo caso, si $c_{t+1} \geq c_t$, entonces se asigna una recompensa neutral, ya que no se mejora significativamente la proximidad de la población P_{t+1} con respecto a P_t hacia el frente de Pareto. En el tercer caso, se asigna una recompensa negativa dado que $c_{t+1} \leq c_t$, indicando que la formulación h no favorece al proceso de búsqueda.

El segundo método, descrito en el Algoritmo 9, está basado en las métricas C y Δ [13], con el fin de promover tanto convergencia como diversidad. Dada una aproximación del frente de Pareto denotada por A, $\Delta(A)$ denota la extensión alcanzada por las n soluciones en A:

$$\Delta(A) = \frac{\sum_{i=1}^{m} d_i^e + \sum_{i=1}^{n} |d_i - \overline{d}|}{\sum_{i=1}^{m} d_i^e + N \cdot \overline{d}} \tag{5.4}$$

en donde d_i denota la distancia entre dos soluciones consecutivas, $\overline{d}$ representa el promedio de las distancias d_i, $i = 1, \ldots, n$, y d_e representa la distancia entre las soluciones extremas de A y soluciones de referencia. La Figura 5.4 ilustra los parámetros de Δ. Entre menor sea la métrica, mejor será la distribución de las soluciones en A. De acuerdo al Algoritmo 9, las variables Δ_t y Δ_{t+1} denotan el valor de la métrica Δ con respecto a las poblaciones P_t y P_{t+1}, respectivamente. Las variables

Algoritmo 9 Asignación de recompensa basado en las métrica C y Δ

```
1: function SetCoverageDeltaReward(P_t, P_{t+1})
2:     c_t ← C(P_t, P_{t+1})
3:     c_{t+1} ← C(P_{t+1}, P_t)
4:     Δ_t ← Δ(P_t)
5:     Δ_{t+1} ← Δ(P_{t+1})
6:     if c_{t+1} ≥ c_t and Δ_{t+1} ≤ Δ_t then
7:         r ← 1                                  ▷ Recompensa positiva
8:     else
9:         if c_{t+1} ≥ c_t or Δ_{t+1} ≤ Δ_t then
10:            r ← 0                              ▷ Recompensa neutral
11:        else
12:            r ← −1                             ▷ Recompensa negativa
13:        end if
14:    end if
15:    return r
16: end function
```

c_t y c_{t+1} se definen de la misma manera que en el Algoritmo 8. En el primer caso, si $c_{t+1} \geq c_t$ y $\Delta_{t+1} \leq \Delta_t$, es decir, si se logra mejorar tanto convergencia como diversidad, entonces se otorga una recompensa positiva. En el segundo caso, si se mejora sólo una de las dos métricas, entonces se asigna una recompensa neutral. En el tercer caso, se otorga una recompensa negativa dado que la formulación h deteriora ambas métricas.

El tercer método, descrito en el Algoritmo 10, está basado en la contribución que aporta cada población al mejor frente encontrado. Las variables c_t y c_{t+1} denotan el número de soluciones que aportan las poblaciones P_t y P_{t+1} al mejor frente F_1. En el primer caso, $c_{t+1} > c_t$ implica que la formulación h es adecuada, por lo que se otorga una recompensa positiva. En el segundo caso, si ambas poblaciones aportan la misma contribución, se asigna una recompensa neutral. En el tercer caso, se devuelve una recompensa negativa, dado que la formulación h es inadecuada.

Algoritmo 10 Asignación de recompensa basado en contribución

```
1: function ContributionReward(P_t, P_{t+1})
2:     (F_1, F_2, ...) ← FastNonDominatedSort(P_t ∪ P_{t+1})
3:     c_t ← | { x : x ∈ P_t ∧ x ∈ F_1 } |
4:     c_{t+1} ← | { x : x ∈ P_{t+1} ∧ x ∈ F_1 } |
5:     if c_{t+1} > c_t then
6:         r ← 1                                   ▷ Recompensa positiva
7:     else
8:         if c_{t+1} = c_t then
9:             r ← 0                               ▷ Recompensa neutral
10:        else
11:            r ← −1                              ▷ Recompensa negativa
12:        end if
13:    end if
14:    return r
15: end function
```

6

Experimentación

Este capítulo presenta la evaluación de tres esquemas de hibridación capaces de abordar diferentes formulaciones de un problema de optimización multi-objetivo: basado en selección aleatoria, basado en un modelo de islas y basado en aprendizaje por refuerzo. Cada esquema se ha evaluado empleando diferentes problemas de prueba tomados de la literatura con $m \in \{2, 3, 5\}$ objetivos por 150 generaciones. El desempeño de cada esquema se ha analizado estadísticamente en las generaciones $t \in \{50, 100, 150\}$. La Sección 6.1 describe los algoritmos evaluados. La Sección 6.2 define el método de escalarización empleado en la experimentación. La Sección 6.3 establece los problemas de prueba y la definición del indicador de desempeño. La Sección 6.4 describe el análisis estadístico efectuado. La experimentación se ha llevado a cabo en dos etapas. En la primera etapa, correspondiente a la Sección 6.5, se ha realizado una comparación preliminar entre los diferentes esquemas de hibridación con el fin de determinar ventajas y áreas de mejora. En la segunda etapa, correspondiente a la Sección 6.6, se describen las modificaciones realizadas al esquema basado en islas con base en las conclusiones obtenidas en la comparación preliminar.

6.1 Descripción de los algoritmos evaluados

El uso de una formulación alternativa durante el proceso de búsqueda podría facilitar la resolución de un problema de optimización multi-objetivo. A partir de esta idea, Ishibuchi *et al.* [24, 27] propusieron tres modelos híbridos basados en NSGA-II, denotados en esta tesis por NSGA-II V1, NSGA-II V2 y NSGA-II V3, con el fin de agregar una formulación alternativa durante el proceso de búsqueda mediante un método de escalarización. De esta forma, es posible comparar la calidad entre un par de individuos mediante el método de escalarización o mediante el método de comparación de NSGA-II, basado en dominancia de Pareto. La elección entre un método u otro se realiza probabilísticamente en cada comparación de individuos durante la selección de padres y la selección de sobrevivientes en cada generación. Durante la selección de padres, cada vez que se comparan dos individuos se elige entre el método de escalarización, con una probabilidad P_{PS}, o el método de comparación de NSGA-II, con una probabilidad $P_{PS} - 1$. De manera similar, durante la selección de sobrevivientes, cada vez que se comparan dos individuos se elige entre el método de escalarización, con una probabilidad P_{GU}, o el método de comparación de NSGA-II, con una probabilidad $P_{GU} - 1$. La diferencia entre las tres versiones de NSGA-II propuestas por Ishibuchi *et al.* [24, 27] radica en el modo en que define cada vector de pesos, denotado por $\boldsymbol{w} = [w_1, \ldots, w_m]^T$, empleado por el método de escalarización, para m objetivos. NSGA-II V1 emplea un único vector de pesos durante el proceso evolutivo, definido como $\boldsymbol{v} = [1, 1, \ldots, 1]^T$. NSGA-II V2 selecciona el vector de pesos $\boldsymbol{w}$ aleatoriamente a partir de un conjunto W con $(2^m - 1)$ vectores binarios, excluyendo al vector $[0, 0, \ldots, 0]^T$; si $m = 2$, entonces $W = \{[1, 1]^T, [1, 0]^T, [0, 1]^T\}$. Por último, NSGA-II V3 selecciona el vector de pesos $\boldsymbol{w}$ aleatoriamente a partir de un conjunto W con vectores discretos, en donde cada vector satisface la relación (6.1):

$$\sum_{i=0}^{m} w_i = d \quad w_i > 0 \tag{6.1}$$

Modelo	$m = 2$			$m = 3$			$m = 5$		
	N	μ	η	N	μ	η	N	μ	η
DIM-A	6	40	40	6	36	36	6	35	35
DIM-B	6	40	20	6	36	18	6	35	18
DIM-C	2	120	120	2	120	120	2	126	126
DIM-D	2	120	60	2	120	60	2	126	63

Tabla 6.1: Modelos del algoritmo propuesto basado en islas. Se muestra el tamaño de la población usado durante la experimentación con $m \in \{2, 3, 5\}$ objetivos.

en donde d es un entero predefinido. Por ejemplo, si $d = 4$ y $m = 2$, entonces $W = \{[4, 0]^T, [3, 1]^T, [2, 2]^T, [1, 3]^T, [0, 4]^T\}$. En este trabajo de tesis se utilizaron los parámetros empleados por los autores originales [24], $P_{PS} = 0.5$, $P_{GU} = 0.5$ y $d = 4$. Adicionalmente, se incluye a NSGA-II en este estudio debido que los tres modelos híbridos anteriores y los dos algoritmos híbridos propuestos en esta tesis están basados en NSGA-II. El primer algoritmo propuesto en esta tesis está basado en un modelo dinámico de islas que permite evaluar diferentes formulaciones de un mismo problema de optimización multi-objetivo, de modo que cada isla aborda una formulación distinta. Se han evaluado cuatro diferentes modelos de este algoritmo, denotadas por DIM-A, DIM-B, DIM-C y DIM-D (ver Tabla 6.1). Cada una de ellos varía en el número de islas N, el número de individuos por isla μ y el número de individuos intercambiados en cada migración η. El tamaño de la población total se calcula como $N\mu$. En todos los modelos, una isla evalúa la formulación original del problema mientras que el resto emplea una formulación escalarizada.

El segundo algoritmo propuesto está basado en un método de aprendizaje por refuerzo que permite evaluar diferentes formulaciones de un mismo problema. La selección entre distintas formulaciones se realiza bajo un esquema de recompensa; si una formulación brinda un beneficio a la búsqueda durante cierta etapa, se recompensa, en otro caso, se penaliza. El método de aprendizaje por refuerzo utilizado es aprendizaje Q con una política voraz ϵ, tal como se emplea en [2]. Se evaluaron tres modelos de este algoritmo, denotadas por RL-MOEA1, RL-MOEA2 y RL-MOEA3 (ver Tabla 6.2). Cada modelo utiliza un método de asignación de recompensa distinto. La diferencia entre cada método es la métrica que se emplea para evaluar a la población luego de usar una formulación. Se

Modelo	Método de asignación de recompensa
RL-MOEA1	Métrica C
RL-MOEA2	Métrica C + Métrica Δ
RL-MOEA3	Contribución

Tabla 6.2: Modelos del algoritmo propuesto basado en aprendizaje por refuerzo.

Selección	Algoritmo	Tamaño de la población		
		$m = 2$	$m = 3$	$m = 5$
Ninguna	NSGA-II	240	210	210
Aleatoria	NSGA-II V1	240	210	210
	NSGA-II V2	240	210	210
	NSGA-II V3	240	210	210
Islas	DIM-A	240	216	180
	DIM-B	240	216	180
	DIM-C	240	240	252
	DIM-D	240	240	252
Aprendizaje	RL-MOEA1	240	210	210
	RL-MOEA2	240	210	210
	RL-MOEA3	240	210	210

Tabla 6.3: Clasificación de los algoritmos evaluados de acuerdo al modo de selección entre formulaciones. Se muestra el tamaño de la población usado durante la experimentación con $m \in \{2, 3, 5\}$ objetivos.

utilizaron tres indicadores: métrica C [50], métrica Δ [13] y contribución, propuesto en esta tesis. La Tabla 6.3 muestra la clasificación de los algoritmos evaluados de acuerdo al modo en que seleccionan una formulación del problema de optimización. Como operadores de variación se ha empleado cruza binaria simulada (*Simulated Binary Crossover*, SBX) y mutación polinomial (*Polynomial Mutation*) con una probabilidad de cruza y de mutación de $P_c = 0.8$ y $P_m = 0.1$, y un índice de distribución de cruza y de mutación $\eta_c = 10$ y $\eta_m = 20$, respectivamente.

6.2 Método de escalarización

En [24, 27], Ishibuchi *et al.* emplearon suma de pesos como método de escalarización para evaluar su híbrido debido a su simplicidad. Sin embargo, esta técnica de escalarización es sensible a la geometría del frente de Pareto ya que es incapaz de encontrar ciertas soluciones en el caso de problemas no convexos [13, 48]. Por tal motivo, se ha modificado el método de escalarización original con el fin de abordar problemas no convexos mediante una técnica conocida como Tchebycheff [13]:

$$\text{Minimizar } u_{te}(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{z}^*) = \max_{1 \leq i \leq m} \{w_i | f_i(\boldsymbol{x}) - z_i^*|\} \tag{6.2}$$

donde $\boldsymbol{f} = [f_1, \ldots, f_m]^T$ es un vector de m funciones objetivo, $\boldsymbol{x} \in X$ es un vector de diseño, $\boldsymbol{w} = [w_1, \ldots, w_m]^T$ es un vector de pesos y $\boldsymbol{z}^* = [z_i, \ldots, z_m]^T$ es el vector ideal, en donde $z_i = \{\min f_i(\boldsymbol{x}) \,|\, \boldsymbol{x} \in X\}$. Por último, se ha empleado un conjunto de vectores de pesos con la finalidad de emplear diferentes direcciones de búsqueda, el cual se ha definido de la siguiente manera: por cada individuo $\boldsymbol{x}_i$ en la población P se ha definido un vector de pesos $\boldsymbol{w}_i$ distinto, $i = 1, \ldots, |P|$. De este modo, se define una tupla $(\boldsymbol{x}_i, \boldsymbol{w}_i)$ por cada individuo en la población. Cada uno de los vectores $\boldsymbol{w}$ se encuentra uniformemente espaciado.

6.3 Problemas e indicador de desempeño

Se han elegido dos conjuntos de problemas de prueba de la literatura especializada: ZDT [50] y DTLZ [15]. El indicador hipervolumen [51] se ha utilizado con la finalidad de medir el desempeño de cada algoritmo. El hipervolumen se define como el volumen total delimitado por una aproximación del frente de Pareto, denotada por A, y un punto de referencia, $\boldsymbol{r} = [r_1, \ldots, r_m]^T$ [11]:

$$\text{HV}(A) = \left\{ \bigcup \text{volumen}(a) : a \in A \right\} \tag{6.3}$$

Función	Objetivo	Modalidad	Geometría	Punto de referencia
DTLZ1	$f_{1:m}$	M	Lineal	$(1,\ldots,1)$
DTLZ2	$f_{1:m}$	U	Cóncavo	$(1,\ldots 1)$
DTLZ3	$f_{1:m}$	M	Cóncavo	$(25,\ldots,25)$
DTLZ4	$f_{1:m}$	U	Cóncavo	$(1,\ldots,1)$
DTLZ5	$f_{1:m}$	U	-	$(4,\ldots,4)$
ZDT1	f_1, f_2	U, U	Convexo	$(2,\ldots,2)$
ZDT2	f_1, f_2	U, U	Cóncavo	$(2,\ldots,2)$
ZDT3	f_1, f_2	U, M	Inconexo	$(2,\ldots,2)$
ZDT4	f_1, f_2	U, M	Convexo	$(2,\ldots,2)$
ZDT6	f_1, f_2	M, M	Cóncavo	$(2,\ldots,2)$

Tabla 6.4: Características de las funciones de prueba, en donde U y M denotan unimodalidad y multimodalidad, respectivamente.

La Tabla 6.4 muestra algunas características de las funciones de prueba, así como el punto de referencia $\boldsymbol{r}$ utilizado para el cálculo del hipervolumen [23]. El vector $\boldsymbol{r}$ se ha calculado experimentalmente, tomando como base los valores de [21]. Cabe mencionar que, en el caso de DTLZ3, el vector de referencia se encuentra muy alejado del frente de Pareto a comparación con el resto de los problemas de prueba (ver Figura 6.1). Esto se debe a que, durante la experimentación, la gran mayoría de los métodos analizados no se aproximaban lo suficiente hacia el frente real y, por lo tanto, no mostraban valor alguno de acuerdo al hipervolumen. Por este motivo, se decidió alejar el punto de referencia en este problema lo suficiente como para reportar un valor de hipervolumen en las generaciones $t \in \{50, 100, 150\}$ y así poder realizar el análisis estadístico.

6.4 Análisis estadístico

Para validar el desempeño de cada algoritmo, se compararon los resultados obtenidos en cada uno de los problemas de prueba durante tres etapas del proceso de búsqueda: inicial ($t = 50$), media ($t = 100$) y final ($t = 150$), donde t hace referencia a la generación. De este modo, es posible identificar al algoritmo más adecuado en diferentes etapas.

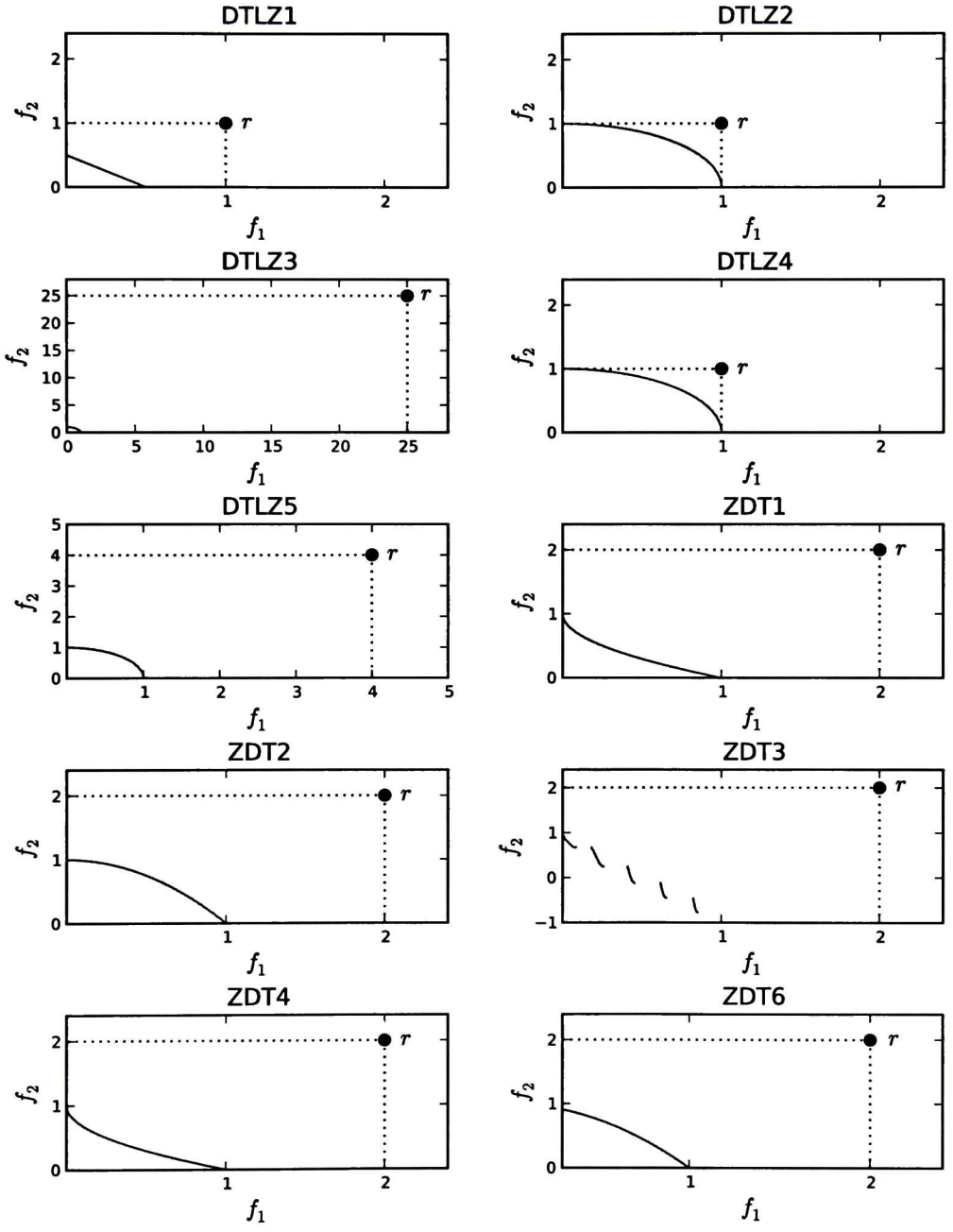


Figura 6.1: Frente de Pareto y punto de referencia r por cada problema de prueba.

El análisis estadístico se llevó a cabo como en [20, 37, 16] de la siguiente manera: En primer lugar, se probó la normalidad de los datos mediante la prueba de Shapiro-Wilk con $\alpha = 0.05$. Luego de comprobar la falta de normalidad, se utilizó la prueba no paramétrica de Friedman para determinar si existía diferencia estadística entre el desempeño de los algoritmos evaluados. Enseguida, se identificaron a aquellos algoritmos que presentaron un desempeño superior. Para el proceso de identificación, se empleó un procedimiento *post hoc*, conocido como Bonferroni, en donde se identificó aquel algoritmo que haya logrado el mejor desempeño en cada uno de los problemas, conocido como método de control. Por último, se identificaron aquellos algoritmos que fueron similares al método de control y aquellos con resultados inferiores.

6.5 Comparación preliminar

A continuación se muestran los resultados obtenidos al término de la primera etapa de la experimentación. Se han evaluado 3 esquemas de hibridación mediante 10 métodos en problemas con $m \in \{2, 3, 5\}$ objetivos (Tabla 6.3). Cada método se ha analizado al término de 50 ejecuciones independientes por cada problema de prueba. La motivación de esta primera etapa es, por cada uno de los métodos, identificar los problemas en los que presentan un buen desempeño, evaluar su rendimiento al incrementar el número de objetivos y encontrar ventajas y áreas de mejora.

6.5.1 Experimento 1: $m = 2$ objetivos

La Tabla 6.5 muestra el análisis estadístico realizado al evaluar la métrica hipervolumen en DTLZ y ZDT con $m = 2$ objetivos en las generaciones $t \in \{50, 100, 150\}$. A partir de esta tabla se puede observar que NSGA-II presentó el mejor desempeño en los problemas DTLZ1, DTLZ2, DTLZ4, DTLZ5 y en la etapa final de ZDT4. Por otro lado, los modelos basados en islas presentaron un buen desempeño en los problemas DTLZ3, ZDT1, ZDT2, ZDT3, ZDT6 y resultaron ser competitivos con respecto a NSGA-II en los problemas DTLZ1 y ZDT4. Por último, cabe mencionar que tanto los

modelos basados en selección aleatoria y como en aprendizaje por refuerzo no lograron sobresalir en este experimento.

Las Figuras 6.2 y 6.3 muestran la mediana del hipervolumen al evaluar DTLZ y ZDT, respectivamente. En cada figura se muestra el método de control en las generaciones $t \in \{50, 100, 150\}$. A partir de la Figura 6.2, se puede observar que los algoritmos basados en islas presentaron una convergencia más rápida en los problemas DTLZ1 y DTLZ3, especialmente durante las primeras etapas de la búsqueda. Por ejemplo, DIM-D logró alcanzar una aproximación al frente de Pareto real en la generación $t = 50$ similar a la aproximación alcanzada por NSGA-II hasta la generación $t = 75$ en DTLZ1. La Figura 6.4 compara las aproximaciones del frente de Pareto encontradas por DIM-D y NSGA-II en la ejecución 25 del experimento; se ha seleccionado esta ejecución por estar ubicada en la mediana de 50 ejecuciones independientes. Se puede observar que el algoritmo basado en islas logró una mayor cercanía al frente de Pareto real utilizando un número menor de generaciones. Del mismo modo, DIM-A y DIM-B lograron una mejor aproximación que NSGA-II en DTLZ3, tal como se muestra en la Figura 6.5. Por otro lado, NSGA-II es superior al resto de los métodos en las funciones DTLZ2, DTLZ4 y DTLZ5. Cabe mencionar que, como se mostrará en los experimentos posteriores, estos tres problemas tienden a ser resueltos con mayor facilidad mediante NSGA-II sin ninguna hibridación, esto es, empleando únicamente dominancia de Pareto. Por esta razón, algunos de los métodos híbridos no destacan sino después de aumentar el número de objetivos, tal como se muestra en las Tablas 6.6 y 6.7. En cuanto a los métodos basados en selección aleatoria, se puede concluir que NSGA-II V1 es superior a NSGA-II V2 y NSGA-II V3 en los problemas DTLZ1 y DTLZ3, lo cual podría indicar que el modo en que NSGA-II V2 y NSGA-II V3 definen los vectores de pesos no es adecuado para este tipo de problemas (ver Figura 6.2). Por último, a partir de la Figura 6.2, se puede observar que los métodos basados en aprendizaje por refuerzo tienden a deteriorar su desempeño en los problemas DTLZ2, DTLZ4 y DTLZ5. Las Figuras 6.6, 6.7 y 6.8 permiten explicar dicho deterioro. La Figura 6.6 muestra las formulaciones empleadas por RL-MOEA1 en DTLZ2 durante el proceso de búsqueda, así como el espacio objetivo durante

las generaciones $t \in \{75, 80, 85, 90, 95\}$. En la generación $t = 75$, el método logró alcanzar el frente de Pareto al emplear la formulación original. Durante las siguientes generaciones, $t \in \{80, 85, 90\}$, RL-MOEA1 utilizó la formulación escalarizada, provocando que las soluciones se agruparan en dos regiones, deteriorando la calidad de la aproximación encontrada. Por último, en la generación $t = 95$, se utilizó la formulación original, permitiendo lograr una mejor aproximación. Las Figuras 6.7 y 6.8 muestran un deterioro similar en los problemas DTLZ4 y DTLZ5, respectivamente.

De acuerdo a la Figura 6.3, los métodos basados en islas fueron superiores a NSGA-II en la mayoría de las funciones, a excepción de ZDT4 en la generación $t = 150$. Las Figuras 6.9, 6.10, 6.11, 6.12 y 6.13 comparan las aproximaciones del frente de Pareto encontradas por DIM-B y NSGA-II al evaluar las funciones ZDT1, ZDT2, ZDT3, ZDT4 y ZDT6, respectivamente, en la ejecución ubicada en la mediana durante el experimento. De manera general, DIM-B logró, en un menos tiempo, una mejor aproximación hacia el frente de Pareto que la alcanzada por NSGA-II, en especial en las funciones ZDT1, ZDT2, ZDT4 y ZDT6. Por ejemplo, la Figura 6.13 muestra cómo DIM-B fue capaz de encontrar el frente de Pareto antes que NSGA-II al evaluar ZDT6.

Problema	t	Tipo de selección										
		Ninguna	Aleatoria			Islas				Aprendizaje		
		NSGA-II	V1	V2	V3	DIM-A	DIM-B	DIM-C	DIM-D	RL-MOEA1	RL-MOEA2	RL-MOEA3
DTLZ1	50					✓	✓	✓	✓*			
	100	✓*						✓	✓			
	150	✓*						✓				
DTLZ2	50	✓*										
	100	✓*										
	150	✓*						✓				
DTLZ3	50					✓	✓*	✓	✓			
	100					✓	✓*		✓			
	150					✓*	✓	✓	✓			
DTLZ4	50	✓*										
	100	✓*										
	150	✓*										
DTLZ5	50	✓*										
	100	✓*						✓	✓			
	150	✓*										
ZDT1	50					✓	✓*	✓	✓			
	100							✓	✓*			
	150	✓						✓	✓*			
ZDT2	50	✓				✓	✓	✓*	✓			
	100					✓	✓	✓*	✓			
	150							✓	✓*			
ZDT3	50					✓	✓*					
	100					✓	✓*	✓	✓			
	150						✓	✓	✓*			
ZDT4	50		✓			✓*	✓					
	100					✓*	✓	✓	✓			
	150	✓*						✓	✓			
ZDT6	50					✓*	✓		✓			
	100					✓	✓*					
	150					✓	✓*		✓			

Tabla 6.5: Análisis estadístico realizado al evaluar la métrica hipervolumen en DTLZ y ZDT con $m = 2$ objetivos en las generaciones $t \in \{50, 100, 150\}$. Por cada fila se denota al método de control mediante * y se identifican a aquellos algoritmos similares al método de control mediante ✓.

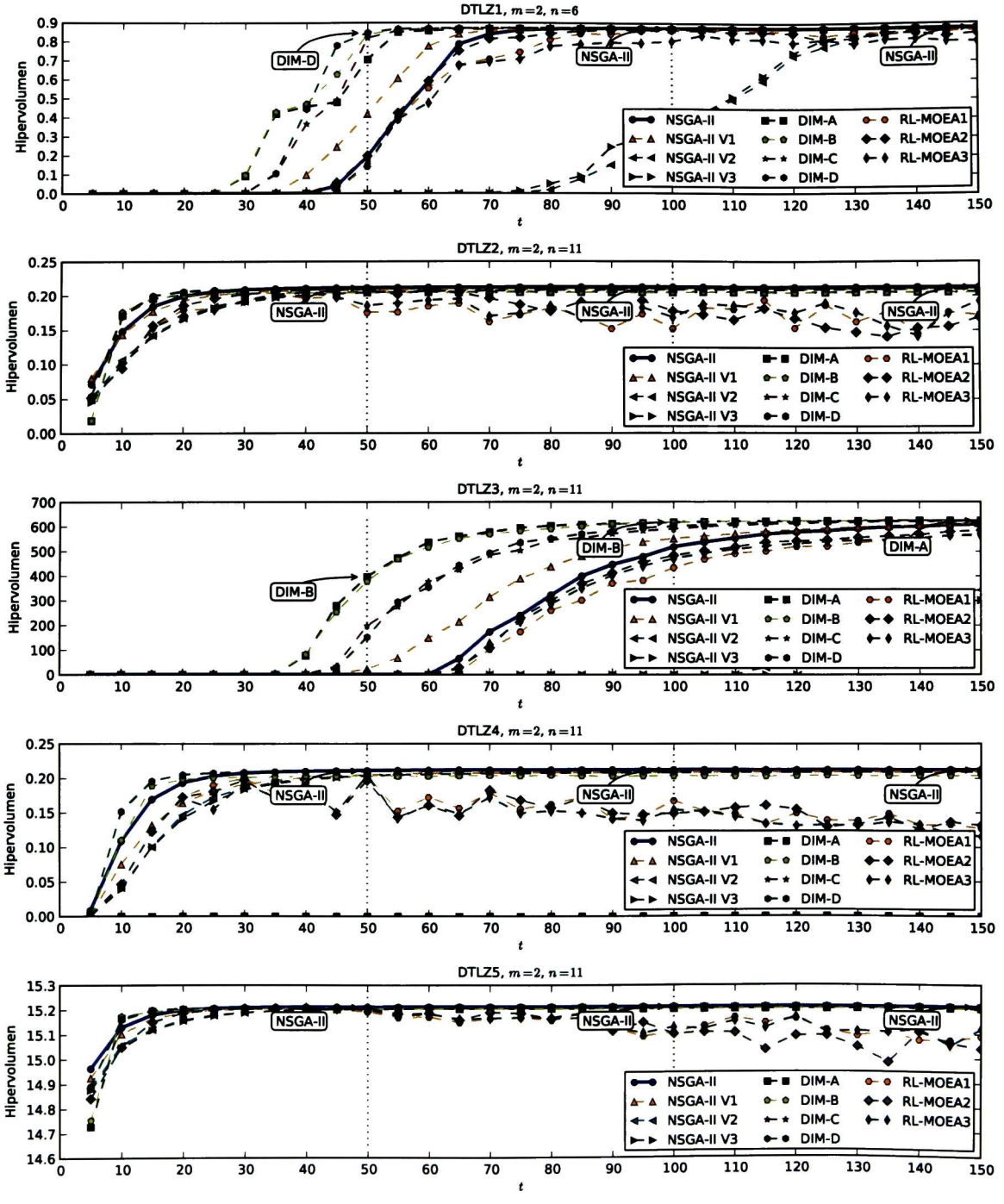


Figura 6.2: Mediana de la métrica hipervolumen obtenida al evaluar las funciones DTLZ con $m = 2$ objetivos. Se muestra el método de control en las generaciones $t \in \{50, 100, 150\}$.

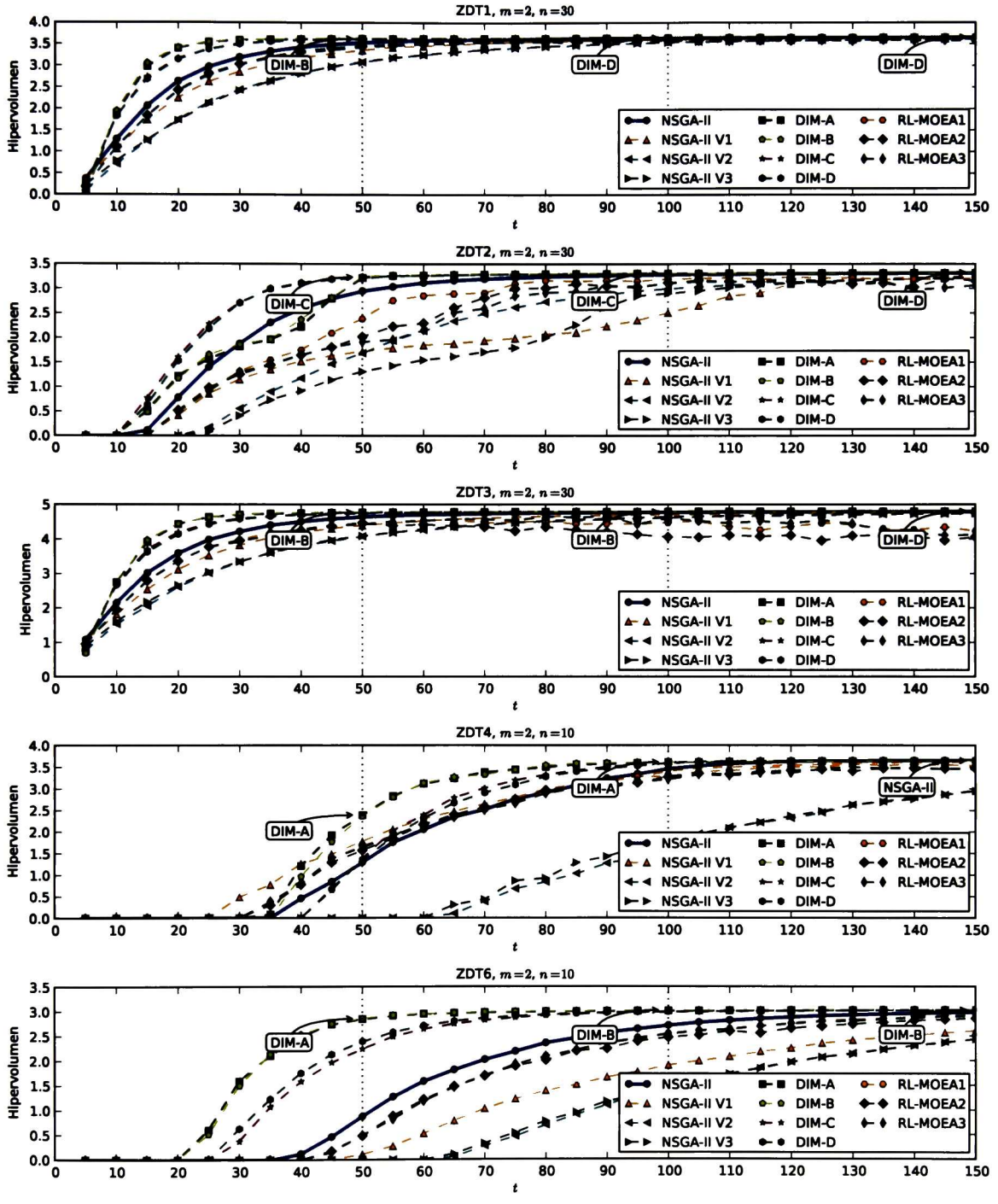


Figura 6.3: Mediana de la métrica hipervolumen obtenida al evaluar las funciones ZDT con $m = 2$ objetivos. Se muestra el método de control en las generaciones $t \in \{50, 100, 150\}$.

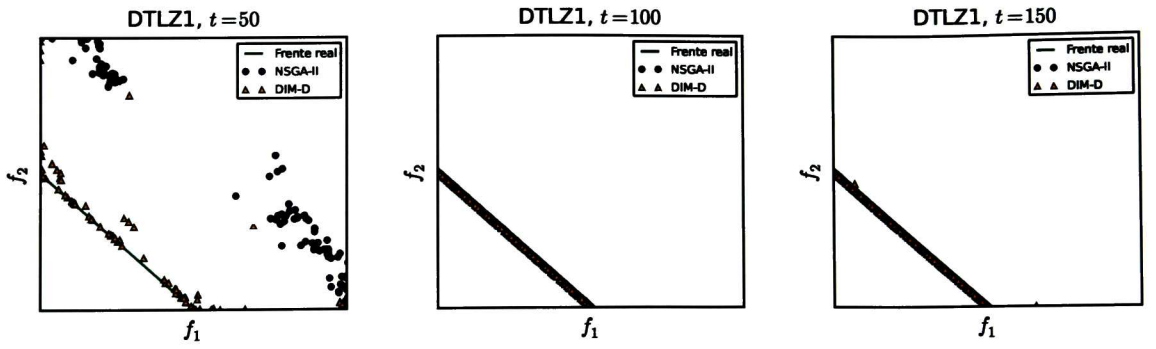


Figura 6.4: Comparación entre DIM-D y NSGA-II al evaluar la función DTLZ1.

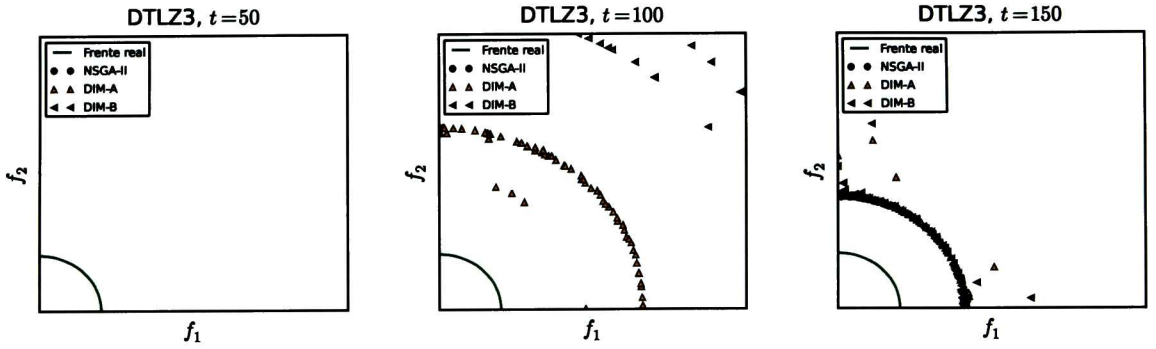


Figura 6.5: Comparación entre DIM-A, DIM-B y NSGA-II al evaluar la función DTLZ3.

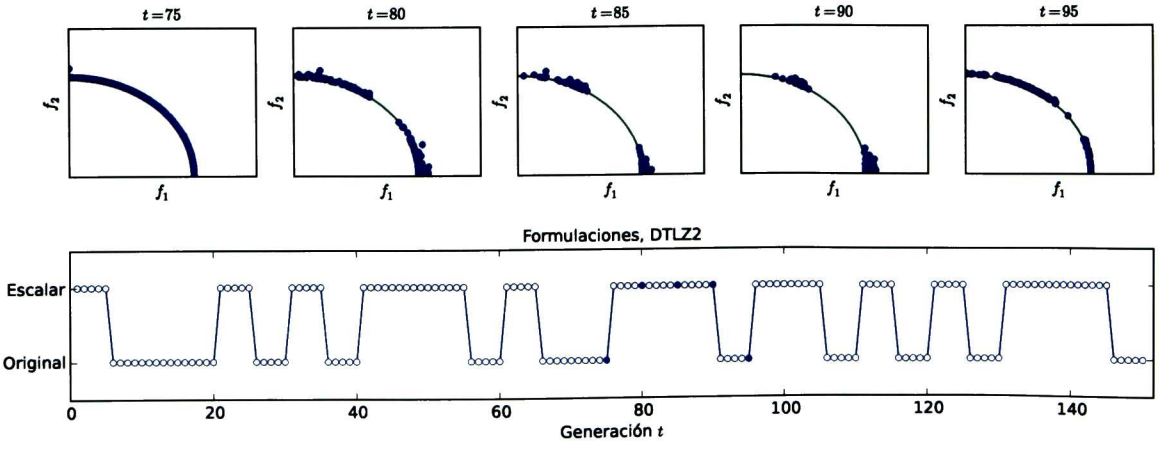


Figura 6.6: Deterioro del desempeño de RL-MOEA1 al evaluar la función DTLZ2 con $m = 2$ objetivos. (Arriba) Espacio objetivo en las generaciones $t \in \{75, 80, 85, 90, 95\}$. (Abajo) Formulaciones empleadas durante el proceso de búsqueda.

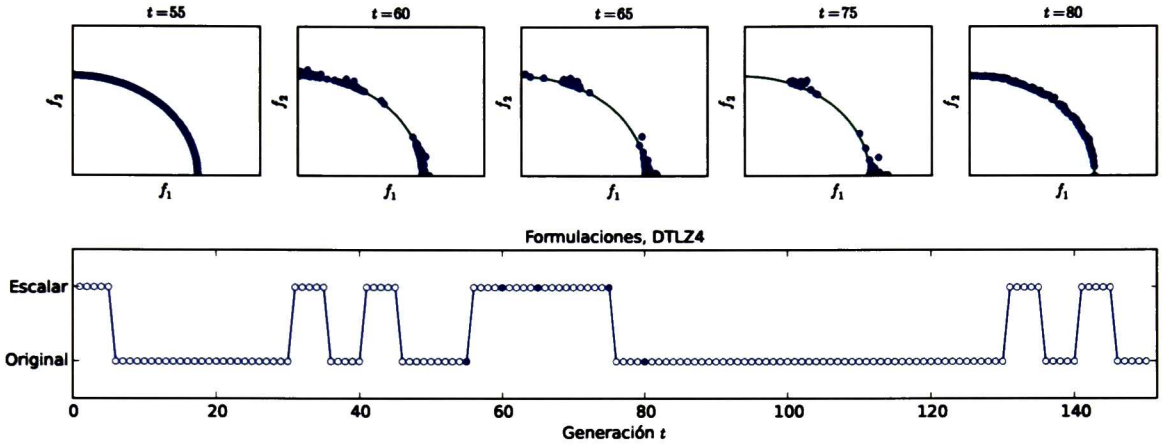


Figura 6.7: Deterioro del desempeño de RL-MOEA1 al evaluar la función DTLZ4 con $m = 2$ objetivos. (Arriba) Espacio objetivo en las generaciones $t \in \{55, 60, 65, 75, 80\}$. (Abajo) Formulaciones empleadas durante el proceso de búsqueda.

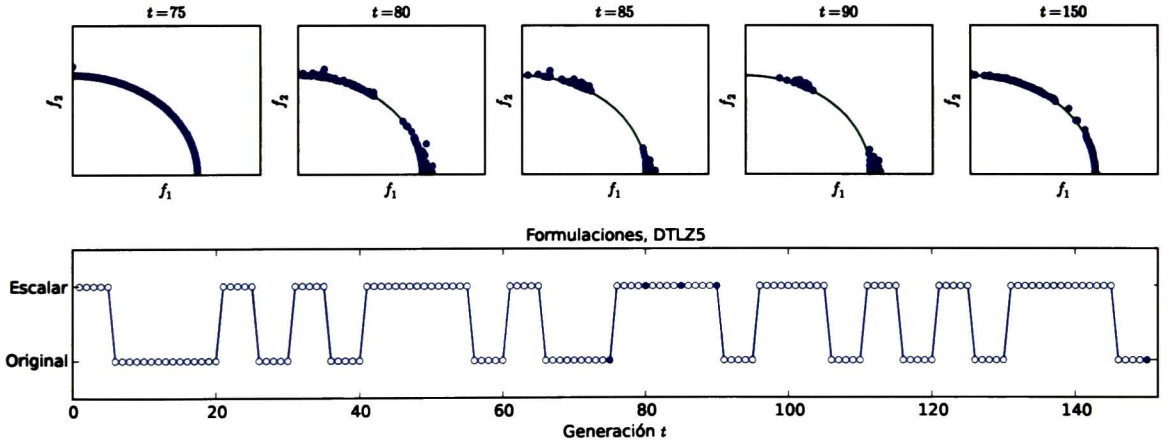


Figura 6.8: Deterioro del desempeño de RL-MOEA1 al evaluar la función DTLZ5 con $m = 2$ objetivos. (Arriba) Espacio objetivo en las generaciones $t \in \{75, 80, 85, 90, 150\}$. (Abajo) Formulaciones empleadas durante el proceso de búsqueda.

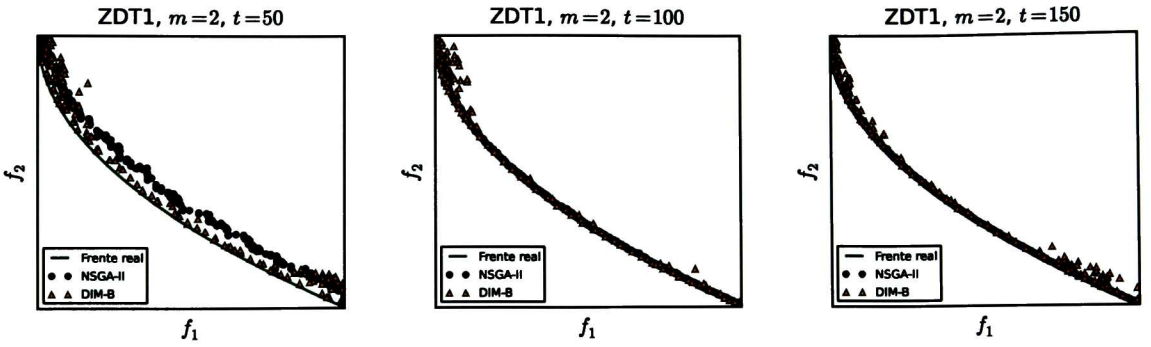


Figura 6.9: Comparación entre DIM-B y NSGA-II al evaluar la función ZDT1.

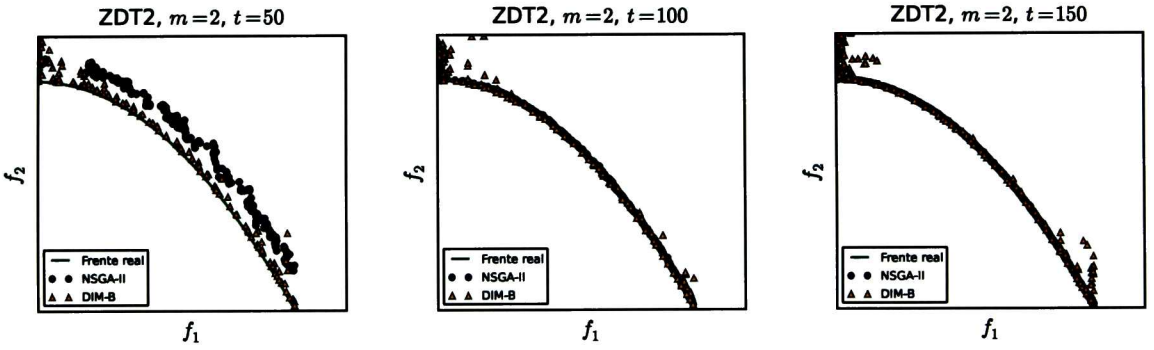


Figura 6.10: Comparación entre DIM-B y NSGA-II al evaluar la función ZDT2.

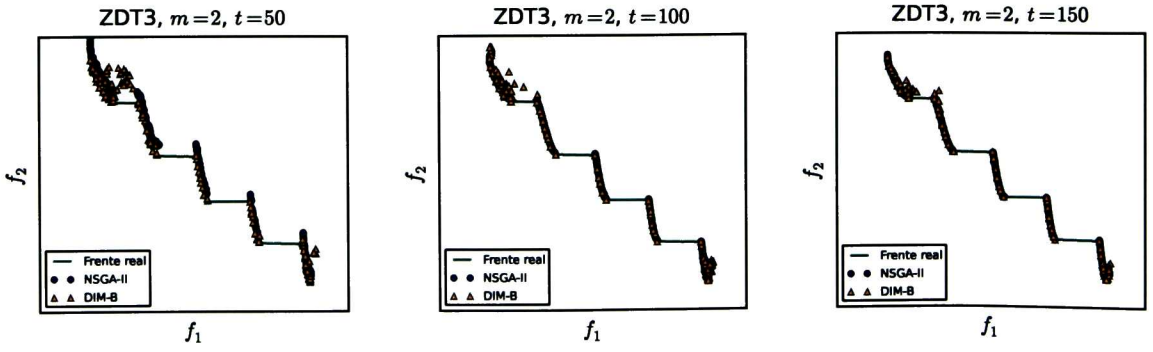


Figura 6.11: Comparación entre DIM-B y NSGA-II al evaluar la función ZDT3.

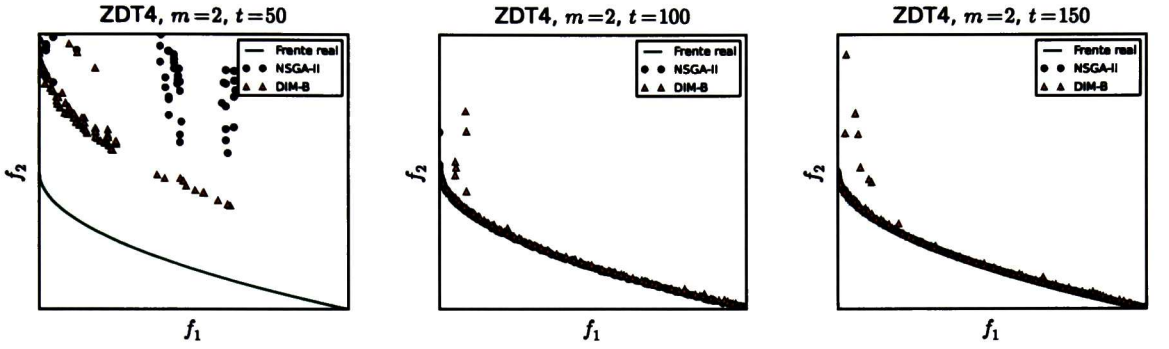


Figura 6.12: Comparación entre DIM-B y NSGA-II al evaluar la función ZDT4.

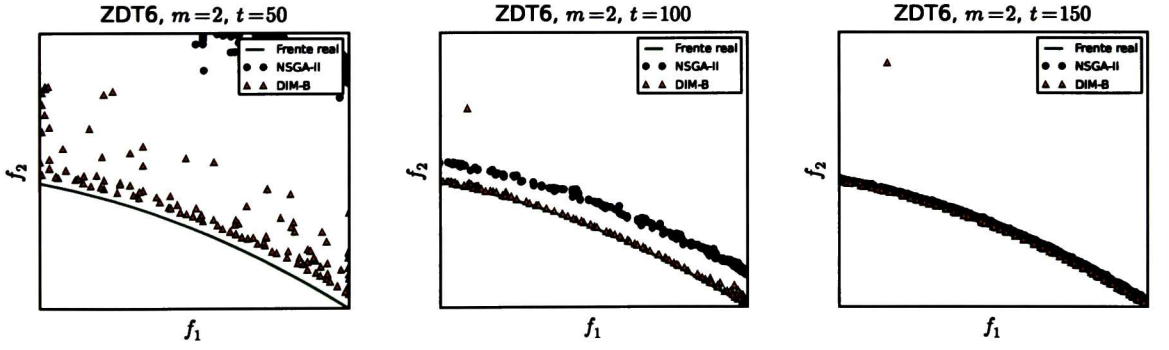


Figura 6.13: Comparación entre DIM-B y NSGA-II al evaluar la función ZDT6.

6.5.2 Experimento 2: $m = 3$ objetivos

La Tabla 6.6 muestra el análisis estadístico realizado al evaluar la métrica hipervolumen en DLTZ con $m = 3$ objetivos en las generaciones $t \in \{50, 100, 150\}$. A partir de esta tabla se puede observar que NSGA-II presenta el mejor desempeño en los problemas DTLZ2, DTLZ4, DTLZ5 y en la etapa final de DTLZ1. Por otro lado, de los cuatro métodos basados en islas, sólo DIM-C y DIM-D son competitivos con respecto a NSGA-II. Al comparar la mediana del hipervolumen obtenida por los algoritmos basados en islas en las funciones DTLZ1 y DTLZ3 con $m \in \{2, 3\}$ (Figuras 6.2 y 6.14), se puede observar que DIM-A y DIM-B son adecuados con $m = 2$ objetivos mientras que DIM-C y DIM-D lo son para $m = 3$. Además, al incrementar el número de objetivos, los métodos basados en selección aleatoria y en aprendizaje por refuerzo muestran una mejora en su desempeño, en especial en las funciones DTLZ1 y DTLZ3. Sin embargo, al analizar sólo los métodos basados en selección aleatoria, únicamente NSGA-II V1 sobresale en este grupo, lo cual refuerza la observación realizada en el experimento anterior, en la cual los vectores de pesos generados por NSGA-II V2 y NSGA-II V3 no son adecuados para este tipo de problemas. Las Figuras 6.15, 6.16, 6.17, 6.18 y 6.19 comparan el desempeño de NSGA-II V1 con respecto a NSGA-II en la cada una de las funciones evaluadas durante la ejecución 25 del experimento; se ha seleccionado esta ejecución por estar ubicada en la mediana de 50 ejecuciones independientes. Se puede observar en la Figura 6.17 que NSGA-II V1 es adecuado para abordar DTLZ3 ya que logra aproximarse más hacia el frente de Pareto que NSGA-II. Por otro lado, al observar la Figura 6.18 se puede apreciar que ambos métodos logran acercarse al frente de Pareto. Por último, al igual que en el experimento anterior, los métodos basados en aprendizaje por refuerzo tienden a deteriorar su desempeño luego de cierto tiempo. Esta situación se aprecia mejor en las Figuras 6.20 y 6.21 correspondientes a la evaluación de DTLZ4 y DTLZ5, respectivamente, mediante RL-MOEA2 durante las generaciones $t \in \{135, 140, 145\}$. En ambas figuras, RL-MOEA2 logra aproximarse al frente de Pareto en $t = 135$, pero tiende a concentrarse en una región en las generaciones posteriores, de manera similar al experimento anterior.

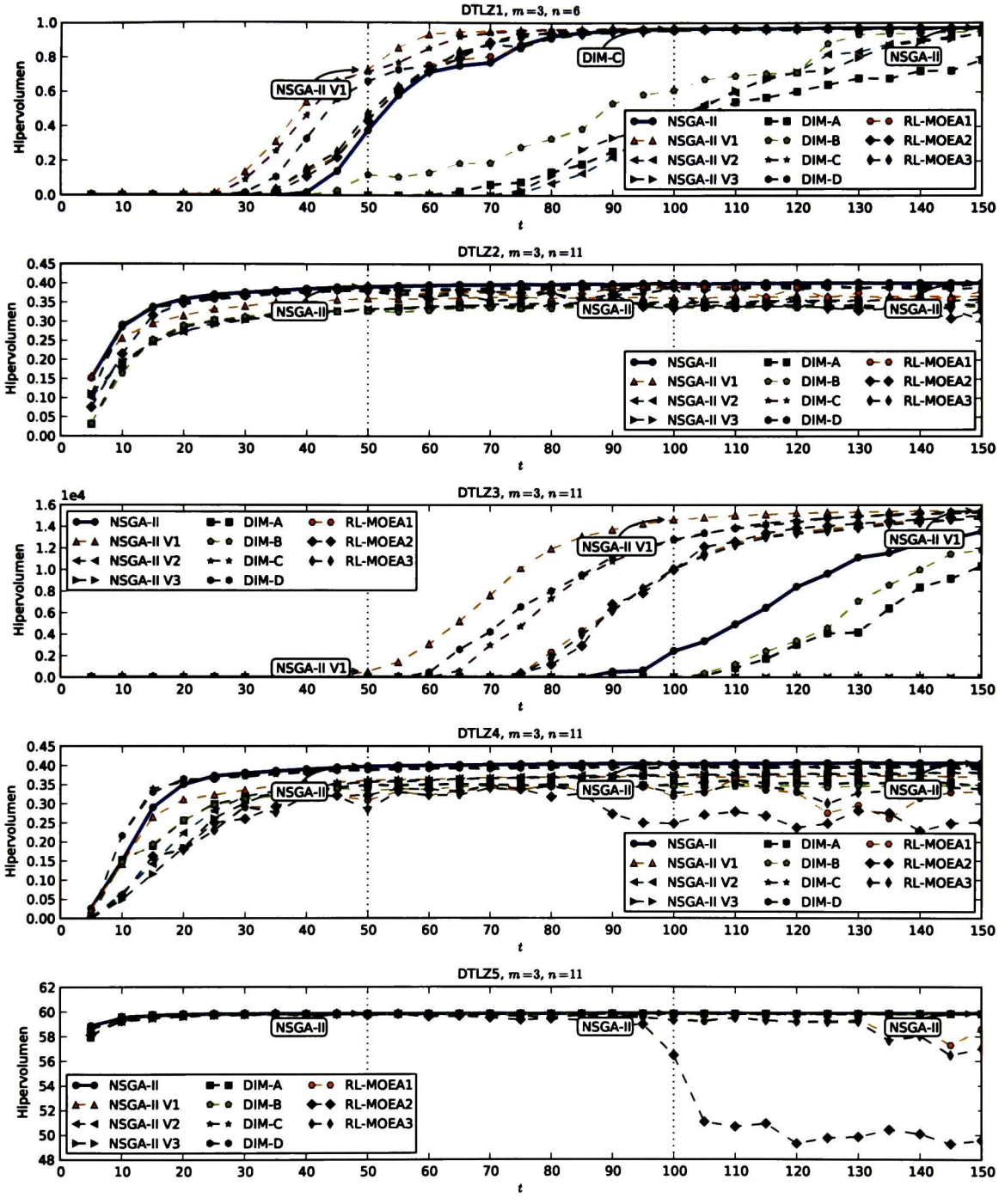


Figura 6.14: Mediana de la métrica hipervolumen obtenida al evaluar las funciones DTLZ con $m = 3$ objetivos. Se muestra el método de control en las generaciones $t \in \{50, 100, 150\}$.

Problema	t	Tipo de selección											
		Ninguna	Aleatoria			Islas				Aprendizaje			
		NSGA-II	V1	V2	V3	DIM-A	DIM-B	DIM-C	DIM-D	RL-MOEA1	RL-MOEA2	RL-MOEA3	
DTLZ1	50		✓*					✓	✓			✓	
	100	✓	✓					✓*	✓	✓	✓	✓	
	150	✓*						✓	✓				
DTLZ2	50	✓*						✓	✓				
	100	✓*						✓					
	150	✓*						✓					
DTLZ3	50		✓*										
	100		✓*					✓	✓				
	150		✓*					✓	✓				
DTLZ4	50	✓*						✓	✓				
	100	✓*						✓	✓				
	150	✓*						✓					
DTLZ5	50	✓*						✓	✓				
	100	✓*						✓					
	150	✓*						✓	✓				

Tabla 6.6: Análisis estadístico realizado al evaluar la métrica hipervolumen en DTLZ con $m = 3$ objetivos en las generaciones $t \in \{50, 100, 150\}$. Por cada fila se denota al método de control mediante * y se identifican a aquellos algoritmos similares al método de control mediante ✓.

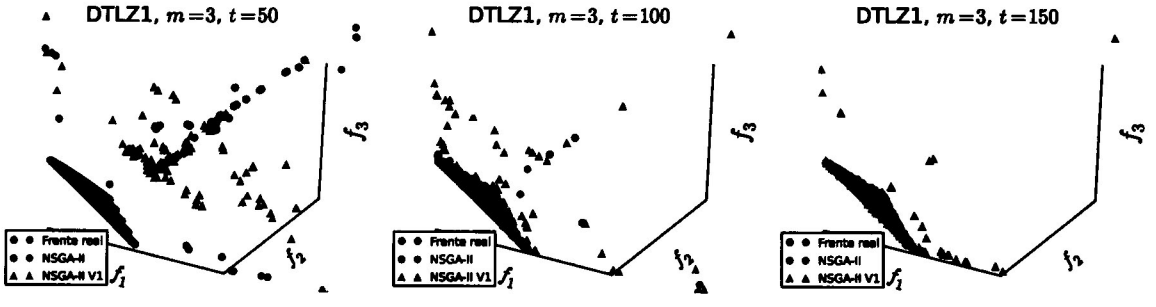


Figura 6.15: Comparación entre NSGA-II V1 y NSGA-II al evaluar la función DTLZ1.

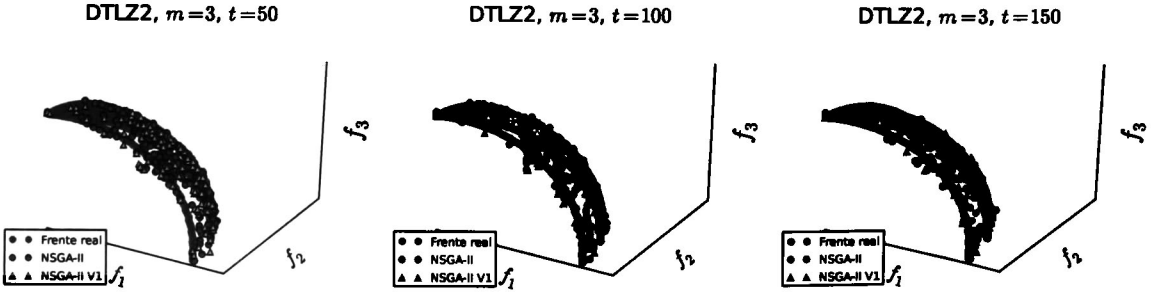


Figura 6.16: Comparación entre NSGA-II V1 y NSGA-II al evaluar la función DTLZ2.

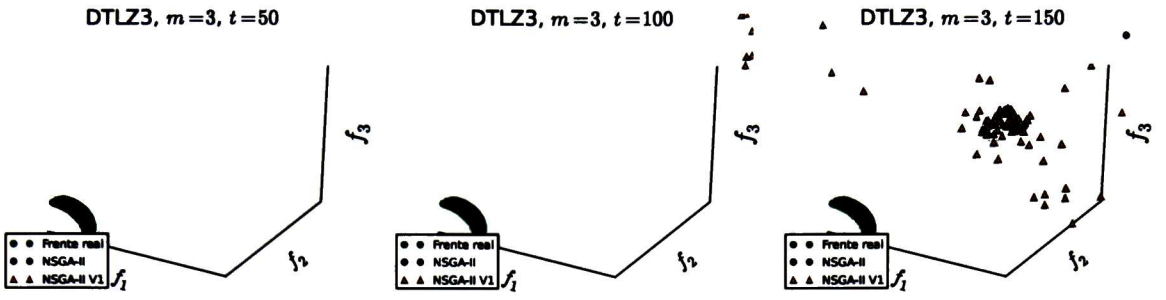


Figura 6.17: Comparación entre NSGA-II V1 y NSGA-II al evaluar la función DTLZ3.

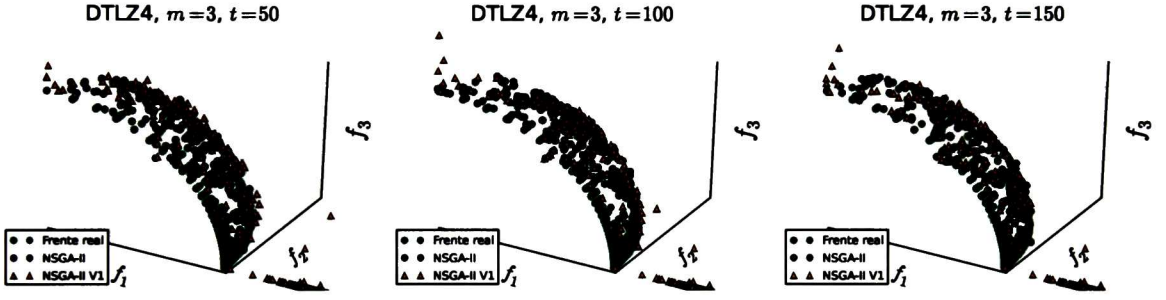


Figura 6.18: Comparación entre NSGA-II V1 y NSGA-II al evaluar la función DTLZ4.

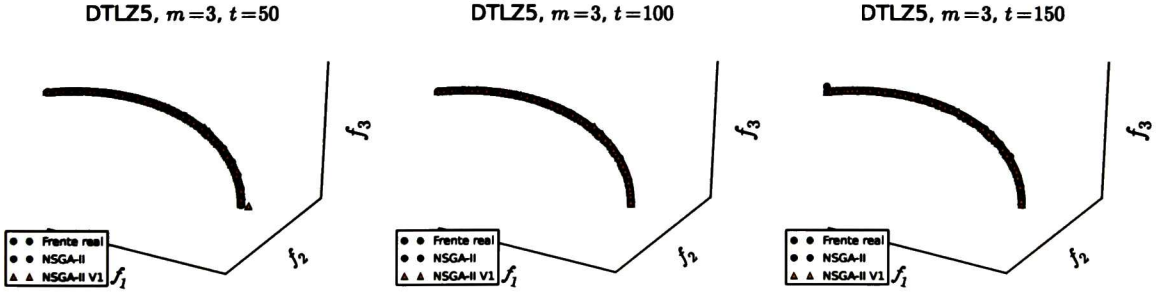


Figura 6.19: Comparación entre NSGA-II V1 y NSGA-II al evaluar la función DTLZ5.

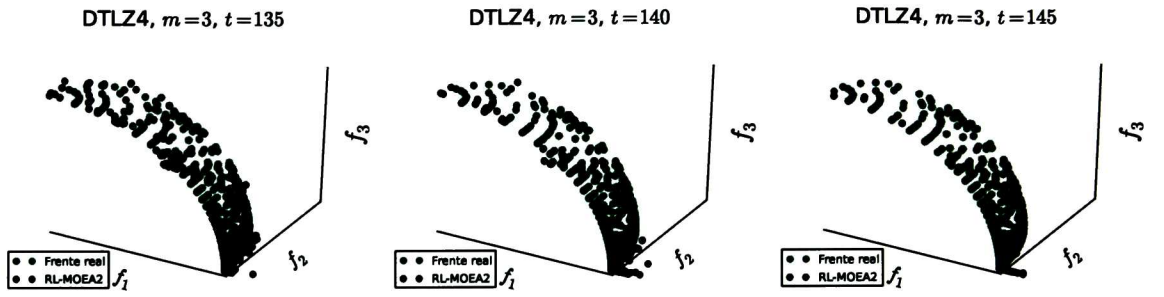


Figura 6.20: Degradación del desempeño de RL-MOEA2 al evaluar la función DTLZ4 con $m = 3$ objetivos durante las generaciones $t \in \{135, 140, 145\}$.

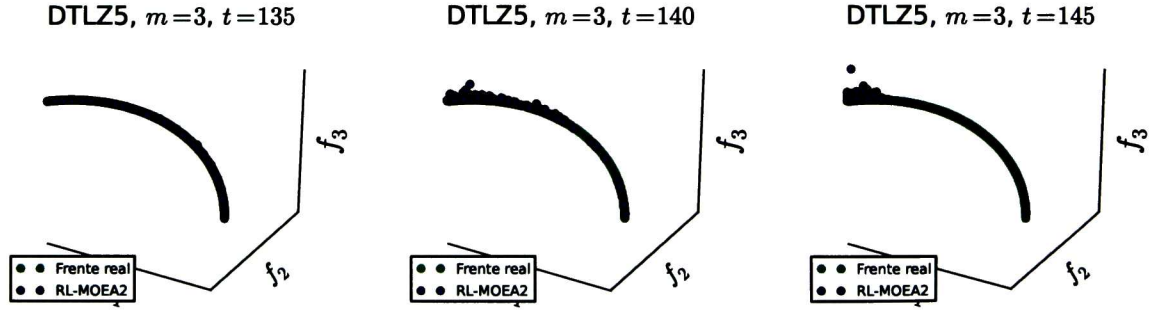


Figura 6.21: Degradación del desempeño de RL-MOEA2 al evaluar la función DTLZ5 con $m = 3$ objetivos durante las generaciones $t \in \{135, 140, 145\}$.

Problema	t	Tipo de selección										
		Ninguna	Aleatoria			Islas				Aprendizaje		
		NSGA-II	V1	V2	V3	DIM-A	DIM-B	DIM-C	DIM-D	RL-MOEA1	RL-MOEA2	RL-MOEA3
DTLZ1	50		✓*									
	100		✓*									
	150		✓*									
DTLZ2	50		✓*								✓	✓
	100		✓*									
	150		✓*									
DTLZ3	50											
	100		✓*									
	150		✓*									
DTLZ4	50		✓*								✓	
	100		✓*								✓	✓
	150		✓*									
DTLZ5	50	✓*	✓					✓				
	100	✓*	✓					✓				
	150	✓*	✓									

Tabla 6.7: Análisis estadístico realizado al evaluar la métrica hipervolumen en DTLZ con $m = 5$ objetivos en las generaciones $t \in \{50, 100, 150\}$. Por cada fila se denota al método de control mediante * y se identifican a aquellos algoritmos similares al método de control mediante ✓.

6.5.3 Experimento 3: $m = 5$ objetivos

La Tabla 6.7 muestra el análisis estadístico realizado al evaluar la métrica hipervolumen en DTLZ con $m = 5$ objetivos en las generaciones $t \in \{50, 100, 150\}$. La Figura 6.22 muestra la mediana del hipervolumen obtenida al evaluar las funciones DTLZ. A partir de la Tabla 6.7 se puede observar que NSGA-II presentó el mejor desempeño únicamente en DTLZ5, mientras que NSGA-II V1 resultó ser el mejor método en los problemas DTLZ1, DTLZ2, DTL3, DTLZ4, y fue similar a NSGA-II en DTLZ5. Además, se puede apreciar que tanto el método basado en islas como el método basado en aprendizaje por refuerzo no presentaron una mejora al incrementar el número de objetivos. Al observar la Figura 6.22 se puede notar cómo NSGA-II V1 superó al resto de los métodos en la mayoría de las funciones. A diferencia del experimento anterior, en donde NSGA-II V1 fue el mejor método sólo para DTLZ3, en esta ocasión lo fue para un mayor número de funciones. Los métodos basados en aprendizaje presentaron un mejor desempeño que NSGA-II en DTLZ2 y DTLZ4. Sin embargo, al igual que en los experimentos anteriores, suelen deteriorar su desempeño en DTLZ5. Este mismo comportamiento ocurre con NSGA-II al evaluar DTLZ4.

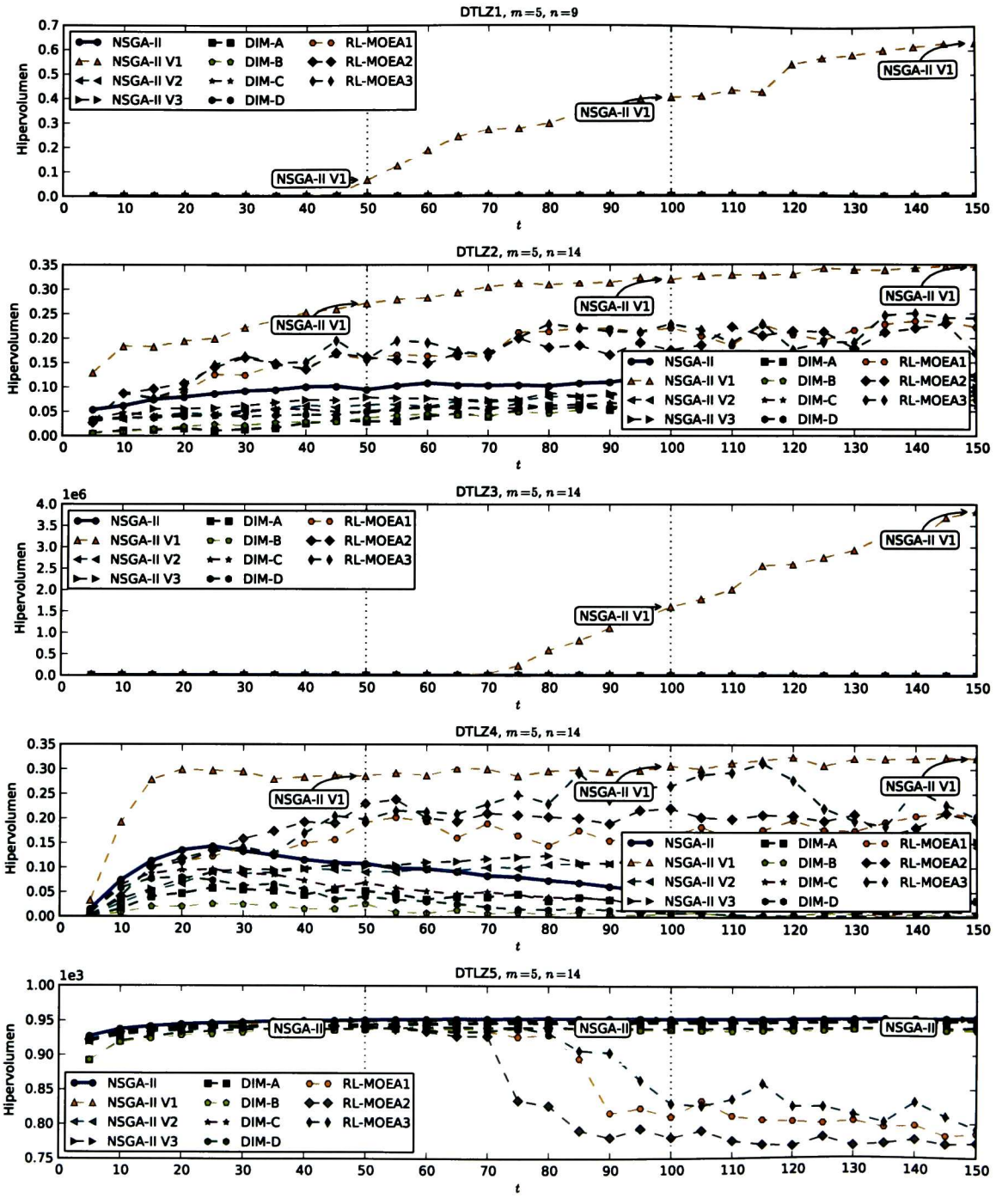


Figura 6.22: Mediana de la métrica hipervolumen obtenida al evaluar las funciones DTLZ con $m = 5$ objetivos. Se muestra el método de control en las generaciones $t \in \{50, 100, 150\}$.

6.5.4 Sumario de la experimentación

Se han evaluado tres diferentes esquemas de hibridación de NSGA-II con la finalidad de incorporar distintas formulaciones de un mismo problema de optimización multi-objetivo. Para evaluar el rendimiento de cada esquema, se utilizaron distintos problemas de prueba con $m \in \{2, 3, 5\}$ objetivos y se analizó el desempeño de cada algoritmo en tres etapas de la búsqueda correspondientes a las generaciones $t \in \{50, 100, 150\}$. Con base en la experimentación, se ha encontrado que al evaluar diferentes formulaciones es posible mejorar el desempeño promedio de NSGA-II. Sin embargo, parte de dicha mejora se debe al diseño de la hibridación y no únicamente a la incorporación de distintas formulaciones, ya que de sólo dos de tres esquemas de hibridación mostraron mejor desempeño que NSGA-II. Es decir, tanto el diseño algorítmico como la incorporación de diferentes formulaciones son factores que impactan en el desempeño de una hibridación. A continuación se muestran las observaciones realizadas en cada uno de los esquemas obtenidas al término de la experimentación.

El esquema basado en selección aleatoria es simple, debido a que no utiliza ningún indicador de desempeño para definir qué formulación usar en cada etapa del proceso evolutivo. Además, ha resultado ser un esquema efectivo en los problemas evaluados con $m \in \{3, 5\}$ objetivos. Sin embargo, no es competitivo al evaluar problemas con $m = 2$ objetivos.

El esquema basado en un modelo dinámico de islas es competitivo con respecto a NSGA-II en los problemas con $m = 2$ objetivos, especialmente durante las primeras etapas de la búsqueda. Este comportamiento se destaca en los problemas multimodales DTLZ1, DTLZ3, ZDT4 y ZDT6. Por otra parte, debido a que el intercambio de individuos se realiza mediante el ordenamiento de NSGA-II, basado en dominancia de Pareto, hereda los inconvenientes de este operador al abordar problemas con un mayor número de objetivos durante la etapa de filtrado. Por este motivo, el desempeño de este método se podría mejorar al modificar la etapa de filtrado.

El esquema basado en aprendizaje por refuerzo representa un modelo más general que el esquema basado en selección aleatoria al incorporar una medida de desempeño. Sin embargo, no fue capaz de

seleccionar la mejor formulación durante diferentes etapas de la búsqueda, lo cual se podría deber a que el método de aprendizaje seleccionado (aprendizaje Q y política voraz ϵ) no logró adaptarse a las características del problema. Por tal motivo, el uso de algún otro método de aprendizaje que sea capaz de adaptarse de manera más rápida podría mejorar el funcionamiento de este esquema.

El esquema basado en aprendizaje por refuerzo no logró desempeñarse adecuadamente en la mayor parte de la experimentación. Una de las diferencias entre los tres modelos es el uso de la población al evaluar diferentes formulaciones. En un modelo basado en aprendizaje, cada formulación es evaluada empleando la población completa (i.e., a nivel de población); en un modelo basado en islas, cada formulación es evaluada en una isla distinta (i.e., a nivel de subpoblación); en un modelo aleatorio, una formulación distinta es utilizada por cada comparación entre individuos (i.e., a nivel de individuo). Es decir, se trata de una diferencia de granularidad. Como se observó durante la experimentación con $m = 2$ objetivos (Figuras 6.6, 6.7 y 6.8), la calidad de la aproximación al frente de Pareto encontrada por este esquema de hibridación se deterioró al cambiar de una formulación a otra, lo cual se atribuye a su nivel de granularidad.

6.6 Modificaciones realizadas al esquema de islas

El modelo de islas propuesto es competitivo con respecto a NSGA-II en los problemas evaluados con $m \in \{2, 3\}$ objetivos (Tablas 6.5 y 6.6). Sin embargo, el desempeño de este modelo disminuye a medida que se incrementa el número de objetivos, tal como se observó en los problemas con $m = 5$ objetivos (Tabla 6.7). Por tal motivo, en esta sección se describen dos aspectos del modelo de islas que se pueden modificar para mejorar su desempeño en problemas con $m > 5$ objetivos: el intercambio de individuos y la definición de los vectores de pesos.

6.6.1 Intercambio de individuos

Una de las características del modelo de islas es el intercambio de individuos. En el modelo propuesto, el intercambio se realiza en dos etapas: filtrado y reemplazo. En el filtrado se crea la población P_{best} a partir de los mejores individuos de la población total luego de ordenar a los individuos mediante no dominancia de Pareto. El reemplazo consiste en sustituir individuos aleatoriamente en cada isla con P_{best}. Como se comentó previamente, el desempeño del método propuesto se deterioró al abordar problemas con $m = 5$ objetivos, lo cual se atribuye al uso de dominancia de Pareto durante la etapa de filtrado. Uno de los principales inconvenientes de un algoritmo basado en no dominancia se presenta al abordar problemas con $m > 3$ objetivos, debido a que los individuos se convierten rápidamente en no dominados entre sí al incrementar el número de objetivos, reduciendo la presión de selección y dificultando el guiado de la búsqueda [47]. Por lo tanto, surge la siguiente pregunta ¿el uso de escalarización durante la etapa de filtrado podría mejorar el desempeño del esquema basado en islas?

6.6.2 Definición de los vectores de pesos

En los experimentos anteriores con $m = 5$ objetivos, NSGA-II V1 mostró el mejor desempeño en la mayoría de las funciones de prueba. A diferencia de NSGA-II V2 y NSGA-II V3, los cuales utilizan un conjunto distinto de vectores de pesos, NSGA-II V1 sólo emplea un único vector de pesos definido por $\boldsymbol{v} = [1, 1, 1, 1, 1]^T$ para $m = 5$ objetivos. Con base en los resultados encontrados, NSGA-II V1 es superior a NSGA-II V2 y NSGA-II V3. Dado que la principal diferencia entre las tres hibridaciones es el modo en que se generan los conjuntos de vectores de pesos, se podría sugerir que la definición del dicho conjunto tiene un impacto en su desempeño. Por otro lado, los modelos basados en islas utilizan vectores de pesos normalizados. Como se mencionó en la Sección 6.2, en la página 69, estos modelos emplean un vector de pesos $\boldsymbol{w}_i$ por cada individuo $\boldsymbol{x}_i$. De este modo, se define una tupla $(\boldsymbol{x}_i, \boldsymbol{w}_i)$ por cada individuo en cada isla. Sin embargo, al observar los resultados obtenidos por los

Algoritmo 11 Filtrado basado en no dominancia y escalarización

```
1: n ← |P|, P_best ← {}
2: (i_1, ..., i_n) ← FastNonDominatedSort(Q)
3: (j_1, ..., j_n) ← ScalarizationSort(Q)
4: for k ∈ {1, ..., η} do
5:     if Flip( ) then
6:         P_best ← P_best ∪ {P[i_k]}          ▷ Ordenamiento basado en no dominancia
7:     else
8:         P_best ← P_best ∪ {P[j_k]}          ▷ Ordenamiento basado en escalarización
9:     end if
10: end for
11: for n ∈ {1, ..., N} do
12:     Reemplazar aleatoriamente η individuos en P_n con P_best
13: end for
```

modelos de islas en los problemas con $m = 5$ objetivos, se encontró un deterioro en su desempeño. Por lo tanto, surge la pregunta, ¿el uso de un único vector de pesos $\boldsymbol{v} = [1, 1, 1, 1, 1]^T$ podría mejorar el funcionamiento de un modelo basado en islas en problemas con $m = 5$ objetivos?

6.6.3 Métodos de intercambio de individuos

El primer método de intercambio emplea dos esquemas de ordenamiento de manera aleatoria: basado en no dominancia y en escalarización. El Algoritmo 11 muestra el método propuesto. Inicialmente, la población total Q se jerarquiza mediante el ordenamiento basado en no dominancia y mediante el valor escalar de cada individuo, generando un conjunto de índices (líneas 2 y 3). Enseguida, se emplean ambos esquemas de ordenamiento aleatoriamente para crear la población P_{best}. Por último, se reemplazan aleatoriamente η individuos en la población P_n de cada isla con los individuos de P_{best}, en donde $n = 1, \ldots, N$. La Figura 6.23 (izquierda y centro) ilustra el procedimiento anterior, en donde la población total $Q = \{a, \ldots, h\}$ es jerarquizada mediante ambos esquemas de ordenamiento. Los mejores dos individuos en Q de acuerdo al ordenamiento basado en no dominancia son a y b, por estar localizados en los extremos del frente, mientras que los mejores dos individuos de acuerdo al ordenamiento basado en escalarización son d y e.

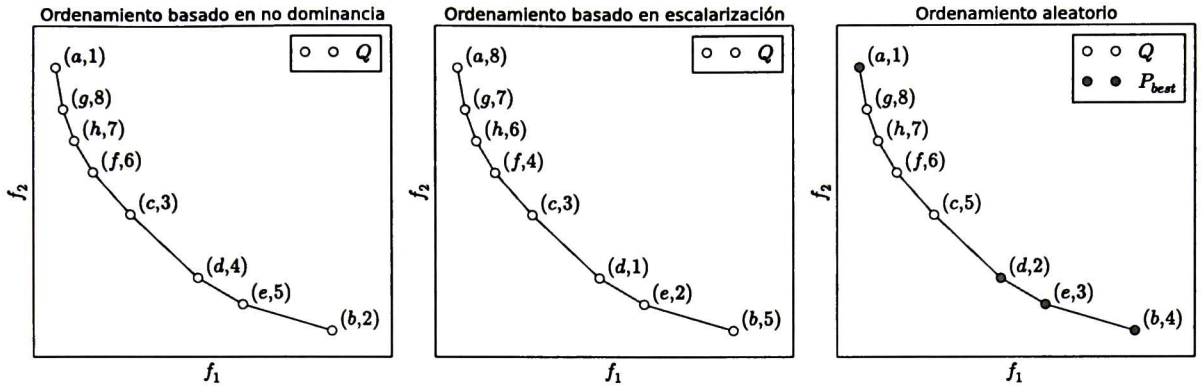


Figura 6.23: Métodos de ordenamiento: (izquierda) basado en no dominancia, (centro) basado en escalarización y (derecha) basado en no dominancia y escalarización aleatoriamente. El número entre paréntesis indica la jerarquía de cada solución.

A partir de Q se crea la población P_{best} con los mejores $\eta = 4$ individuos de acuerdo a ambos esquemas empleados aleatoriamente, en donde $P_{best} = \{a, d, e, b\}$ (derecha). El segundo método de intercambio de individuos está basado en una técnica de descomposición [35] capaz de dividir un problema de optimización multi-objetivo en múltiples subproblemas. Para este fin, se define un conjunto de k vectores uniformemente distribuidos en el espacio objetivo $Z \in \mathbb{R}^m$, denotado por $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k\}$. Enseguida, se divide Z en k subregiones, denotadas por $\Omega_1, \ldots, \Omega_k$, de la siguiente manera:

$$\Omega_i = \{\boldsymbol{u} \in Z \mid \langle \boldsymbol{u}, \boldsymbol{w}_i \rangle \leq \langle \boldsymbol{u}, \boldsymbol{w}_j \rangle \text{ para cualquier } j = 1, \ldots, k\} \tag{6.4}$$

en donde $\langle \boldsymbol{u}, \boldsymbol{w}_i \rangle$ denota el ángulo entre los vectores $\boldsymbol{u}$ y $\boldsymbol{w}_i$. En otras palabras, $\boldsymbol{u}$ pertenece a la región Ω_i si el ángulo formado entre $\boldsymbol{u}$ y $\boldsymbol{w}_i$ es el menor respecto a los demás vectores de dirección.

La Figura 6.24 (izquierda) muestra la distribución de la población $Q = \{a, \ldots, l\}$ en tres diferentes subregiones, Ω_1, Ω_2 y Ω_3, definidas por los vectores $\boldsymbol{w}_1, \boldsymbol{w}_2$ y $\boldsymbol{w}_3$, respectivamente:

$$\Omega_1 = \{j, k, l, i, g, h\} \qquad \Omega_2 = \{b, c\} \qquad \Omega_3 = \{a, d, e, f\} \tag{6.5}$$

Algoritmo 12 Intercambio de individuos basado en subregiones

```
1: for n ∈ {1, ..., N} do
2:     Inicializar P_n a partir de las soluciones en Q que se encuentran en Ω_n
3:     if |P_n| < μ then
4:         Agregar μ − |P_n| individuos a P_n a partir de Q seleccionadas aleatoriamente
5:     end if
6:     if |P_n| > μ then
7:         Ordenar las soluciones en P_n mediante el ordenamiento rápido basado en no dominancia
8:         Remover las peores |P_n| − μ soluciones en P_n
9:     end if
10: end for
```

Esta técnica de descomposición se emplea durante el intercambio de individuos del modelo de islas, de manera que la población P_n de cada isla se actualice de acuerdo a los individuos más próximos a la subregión Ω_n, $n = 1, \ldots, N$. De este modo, cada una de las N islas evoluciona independientemente al concentrarse en una subregión distinta. El Algoritmo 12 muestra el esquema propuesto para actualizar cada población P_n con μ individuos por isla. En un inicio, se define la población P_n a partir de los individuos en la población total Q que se encuentran en la subregión Ω_n. Si $|P_n| < \mu$, es decir, el número de individuos es insuficiente, entonces se agregan $\mu - |P_n|$ individuos de la población total Q seleccionados de manera aleatoria. Por otro lado, si $|P_n| > \mu$, entonces los individuos en P_n son ordenados y los peores $|P_n| - \mu$ son removidos. La Figura 6.24 (derecha) muestra la composición de tres poblaciones con $\mu = 4$ individuos por isla luego de emplear el método de descomposición descrito previamente:

$$\begin{aligned} P_1 &= \{j, k, l, i\} \\ P_2 &= \{b, c, a, d\} \\ P_3 &= \{f, a, d, e\} \end{aligned} \tag{6.6}$$

La población P_1 de define a partir de Ω_1. Dado que el número de individuos en esta región excede el tamaño de la isla, $|\Omega_1| > \mu$, P_1 se compone de los mejores μ individuos en Ω_1, descartando a los individuos h y g. La población P_2 se define a partir de Ω_2. Dado que el número de individuos en

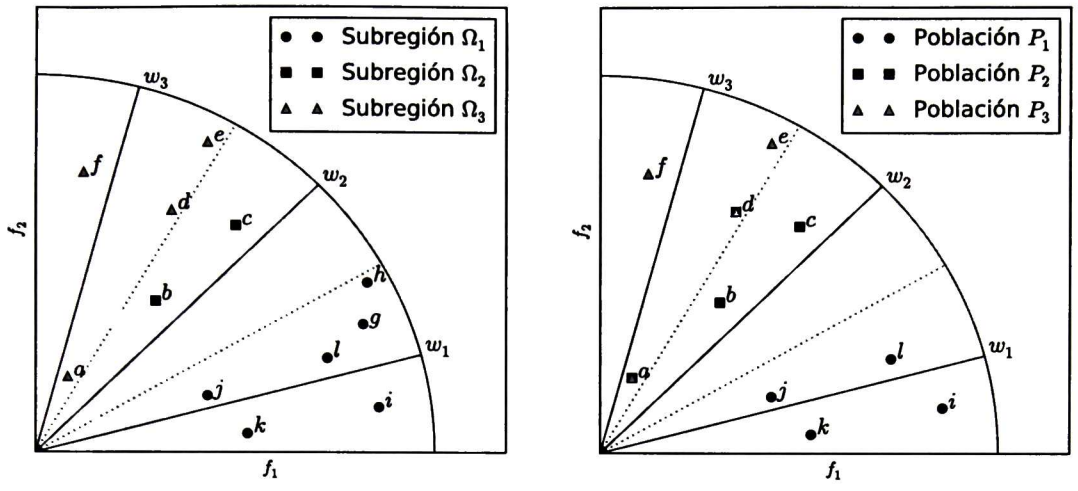


Figura 6.24: División del espacio objetivo en tres subregiones. (Izquierda) Composición de las tres subregiones, Ω_1, Ω_2 y Ω_3, antes del intercambio de individuos. (Derecha) Composición de las tres poblaciones, P_1, P_2 y P_3, luego del intercambio de individuos.

esta región es insuficiente, $|\Omega_2| < \mu$, las soluciones a y d son seleccionadas aleatoriamente a partir de Q y agregadas a P_2. Por último, la población P_3 es definida a partir de Ω_3. Dado que el número de individuos en esta región equivale al tamaño de la isla, $|\Omega_3| = \mu$, entonces $P_3 = \Omega_3$.

El tercer método de intercambio de individuos es una modificación del método anterior. En el Algoritmo 12, el ordenamiento rápido basado en no dominancia es empleado si el tamaño de la población P_n sobrepasa el número de individuos permitidos por isla, μ (línea 6). Dado que este

Algoritmo 13 Intercambio de individuos basado en subregiones y escalarización

1: **for** $n \in \{1, \ldots, N\}$ **do**
2: Inicializar P_n a partir de las soluciones en Q que se encuentran en Ω_n
3: **if** $|P_n| < \mu$ **then**
4: Agregar $\mu - |P_n|$ individuos a P_n a partir de Q seleccionadas aleatoriamente
5: **end if**
6: **if** $|P_n| > \mu$ **then**
7: Ordenar las soluciones en P_n mediante no dominancia y escalarización de manera aleatoria.
8: Remover las peores $|P_n| - \mu$ soluciones en P_n
9: **end if**
10: **end for**

Modelo DIM	Vector de pesos por individuo x_i		Método de intercambio de individuos			
	Real (x_i, w_i)	Discreto (x_i, v)	No dominancia	No dominancia/Escalarización	Subregiones	Subregiones/Escalarización
DIM-E	✓		✓			
DIM-F	✓			✓		
DIM-G	✓				✓	
DIM-H	✓					✓
DIM-I		✓	✓			
DIM-J		✓		✓		
DIM-K		✓			✓	
DIM-L		✓				✓

Tabla 6.8: Modelos evaluados del esquema de islas.

ordenamiento está basado en no dominancia, podría deteriorar su desempeño al incrementarse el número de objetivos, por lo cual se plantea emplear escalarización de manera aleatoria en dicho caso. El Algoritmo 13 muestra el método propuesto.

6.6.4 Evaluación y resultados

Para validar el desempeño de los tres métodos de intercambio de individuos se han evaluado diferentes modelos del esquema de islas mostrados en la Tabla 6.8. Cada modelo emplea un tipo de vector de pesos por individuo x_i y un método de intercambio distinto. En el caso de los vectores de pesos, se han evaluado dos tipos de vectores: real y discreto. Los modelos que emplean vectores reales definen un vector de pesos w_i por cada individuo x_i, en donde cada vector real w_i se encuentra normalizado. Por otro lado, el resto de los modelos emplean el mismo vector discreto $v = [1, 1, 1, 1, 1]^T$ empleado por NSGA-II V1 por cada individuo x_i. En el caso de los métodos de intercambio, se han evaluado tres métodos distintos: basado en no dominancia y escalarización (Algoritmo 11), basado en subregiones (Algoritmo 12) y basado en subregiones y escalarización (Algoritmo 13). Finalmente, se ha incluido el método de intercambio basado en no dominancia descrito en la Sección 4.2 a modo de comparación.

A continuación se muestran los resultados obtenidos al término de la experimentación. La Figura 6.25 muestra la mediana de la métrica hypervolumen al evaluar los problemas DTLZ con $m = 5$ objetivos. La Tabla 6.9 muestra un resumen del análisis estadístico realizado al evaluar la métrica

Problema	t	NSGA-II	NSGA-II V1	DIM-E	DIM-F	DIM-G	DIM-H	DIM-I	DIM-J	DIM-K	DIM-L
DTLZ1	50		✓						✓*		✓
	100		✓						✓*	✓	✓
	150		✓						✓	✓*	✓
DTLZ2	50		✓*							✓	✓
	100		✓				✓			✓*	✓
	150					✓	✓			✓	✓*
DTLZ3	50		✓						✓*	✓	✓
	100		✓						✓*	✓	✓
	150		✓						✓*	✓	✓
DTLZ4	50		✓							✓*	✓
	100		✓							✓*	✓
	150		✓							✓*	✓
DTLZ5	50	✓	✓					✓	✓*		
	100	✓*	✓								
	150	✓*									

Tabla 6.9: Análisis estadístico realizado al evaluar la métrica hipervolumen en DTLZ con $m = 5$ objetivos en las generaciones $t \in \{50, 100, 150\}$. Por cada fila se denota al método de control mediante * y se identifican a aquellos algoritmos similares al método de control mediante ✓.

hipervolumen durante las generaciones $t \in \{50, 100, 150\}$. A partir de esta tabla se puede observar que el uso de un tipo de vector de pesos influye en el desempeño del método de intercambio. Considere los modelos DIM-F y DIM-J, los cuales emplean el mismo método de intercambio basado en no dominancia y escalarización pero mediante un tipo de vector de pesos distinto. Por un lado, DIM-F emplea vectores reales, pero no logró destacar durante la experimentación. Por otro lado, el modelo DIM-J emplea un único vector de pesos discreto logrando mejorar el desempeño del esquema de islas, especialmente en los problemas multimodales DTLZ1 y DTLZ3. Este mismo comportamiento se presenta al comparar DIM-G con DIM-K y DIM-H con DIM-L, en donde DIM-K y DIM-L muestran un mejor desempeño que sus contrapartes DIM-G y DIM-L, respectivamente. Al comparar únicamente los métodos de intercambio, se puede observar que los tres métodos propuestos descritos en la Sección 6.6.3, correspondientes a los modelos DIM-J, DIM-K y DIM-L, fueron superiores al método de intercambio basado en no dominancia, correspondiente al modelo DIM-I, en la mayoría de los problemas de prueba. El método de intercambio basado en no dominancia y escalarización (DIM-J) sobresale en los problemas multimodales DTLZ1 y DTLZ3, mientras que los métodos de intercambio basado en subregiones (DIM-K y DIM-L) resaltan en los problemas DTLZ1, DTLZ2, DTLZ3 y DTLZ4.

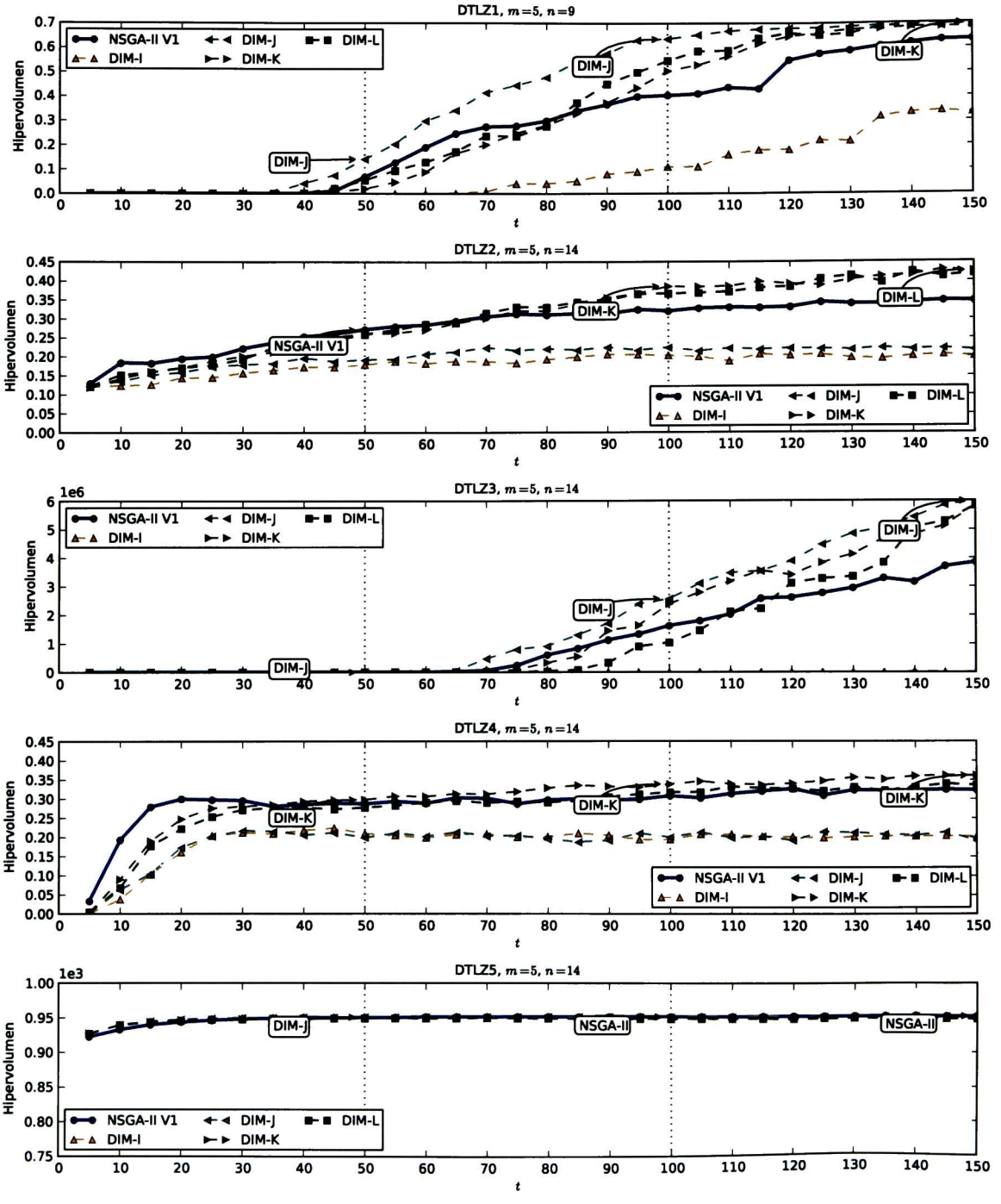


Figura 6.25: Mediana de la métrica hipervolumen obtenida al evaluar las funciones DTLZ con $m = 5$ objetivos. Se muestra el método de control en las generaciones $t \in \{50, 100, 150\}$.

6.6.5 Sumario de la experimentación

Luego de analizar el esquema propuesto basado en un modelo de islas en la Sección 6.5, se ha encontrado un deterioro en su rendimiento al abordar problemas con $m = 5$ objetivos, atribuido al uso del método de filtrado basado en no dominancia. Por este motivo, se propusieron y evaluaron tres nuevos métodos de filtrado con el fin de mejorar el desempeño de este esquema en los problemas de prueba con $m = 5$ objetivos. Al término de la experimentación, se encontró que los métodos de filtrado propuestos basados en subregiones permitieron mejorar el esquema de islas en función del hipervolumen alcanzado.

Con base en las observaciones realizadas en la experimentación, se concluye que el método de filtrado basado en no dominancia es recomendado para problemas con $m \in \{2, 3\}$ objetivos. Por otro lado, se recomienda emplear un método de filtrado basado en escalarización o en subregiones para problemas con un mayor número de objetivos.

7

Conclusiones y trabajo futuro

Un problema de optimización multi-objetivo puede ser transformado mediante diferentes técnicas con el fin de generar una formulación alternativa del problema original. En la literatura se ha encontrado que es posible mejorar el desempeño de un algoritmo evolutivo al evaluar diferentes formulaciones aleatoriamente durante el proceso de búsqueda.

En esta tesis se proponen dos nuevos esquemas de hibridación capaces de abordar diferentes formulaciones de un problema de optimización. El primero está basado en un modelo dinámico de islas y el segundo está basado en aprendizaje por refuerzo. Adicionalmente, el desempeño de ambos esquemas de hibridación se ha comparado con respecto a una propuesta desarrollada previamente en la literatura [24, 27] utilizando diferentes problemas de prueba. A continuación se resumen las observaciones efectuadas por cada uno de los tres esquemas analizados.

Uso probabilístico de diferentes formulaciones

El esquema de hibridación de Ishibuchi *et al.* [24, 27] incorpora un método de escalarización durante la evaluación de la calidad de cada individuo en la población, permitiendo el uso probabilístico de dominancia de Pareto y de escalarización. A partir de este esquema de hibridación propusieron tres modelos: NSGA-II V1, NSGA-II V2 y NSGA-II V3. En esta tesis, se han evaluado los tres modelos anteriores en problemas continuos con $m \in \{2, 3, 5\}$ objetivos.

A partir de la comparación preliminar realizada en la Sección 6.5, en la pág. 72, se encontró que NSGA-II V1 fue superior a NSGA-II V2 y NSGA-II V3, incluso fue superior a NSGA-II en diferentes problemas con $m \in \{3, 5\}$ objetivos. La única diferencia entre los tres modelos es el modo en que se definen los vectores de pesos empleados por el método de escalarización. NSGA-II V2 y NSGA-II V3 emplean vectores de pesos discretos no normalizados, brindando a cada objetivo una importancia diferente, mientras que NSGA-II V1 emplea un solo vector de pesos, definido por $\boldsymbol{v} = [1, \ldots, 1]^T$, brindando la misma importancia a cada objetivo.

La experimentación realizada en la Sección 6.5 ha mostrado que la selección de los vectores de pesos empleados por el esquema de hibridación de Ishibuchi *et al.* impacta en su desempeño al abordar problemas continuos.

Modelo de islas

El Capítulo 4, en la pág. 45, presentó el esquema de hibridación basado en un modelo dinámico de islas. En este esquema, la población se divide en diferentes grupos o islas, las cuales evolucionan independientemente empleando una formulación distinta del problema.

A raíz de la comparación preliminar realizada en la Sección 6.5, se encontró que el desempeño de este esquema es superior al de NSGA-II en problemas multimodales con $m \in \{2, 3\}$ objetivos. Sin embargo, también se observó que el desempeño del método propuesto tiende a deteriorarse a medida que aumenta el número de objetivos. Este declive se debe al uso de dominancia de Pareto durante

el intercambio de individuos entre cada isla. Para afrontar esta situación, en la Sección 6.6 (pág. 90) se presentaron tres nuevos métodos para el intercambio de individuos con la finalidad de mejorar el desempeño de este esquema en problemas con $m = 5$ objetivos. Adicionalmente, se exploró cómo la definición de los vectores de pesos influye en su funcionamiento.

Con base en la experimentación realizada en la Sección 6.6.4 (pág. 96), se encontró que tanto el diseño del método de intercambio de individuos como la selección de los vectores de pesos contribuyen al desempeño del método propuesto. De acuerdo al análisis desarrollado, los métodos de intercambio basados en subregiones y el uso de vectores discretos presentaron los mejores resultados para problemas con $m = 5$ objetivos. A partir de los resultados obtenidos en las Secciones 6.5 y 6.6, se recomienda utilizar un método de filtrado basado en no dominancia para problemas con un número reducido de objetivos, $m \in \{2, 3\}$, mientras que se recomienda emplear un método de filtrado basado en escalarización o en subregiones en problemas con un mayor número de objetivos.

Aprendizaje por refuerzo

El Capítulo 5 presentó el esquema de hibridación basado en aprendizaje Q y una política voraz ϵ con la finalidad de evaluar diferentes formulaciones de un problema de optimización. La selección entre una formulación u otra se realiza bajo un esquema de recompensa; si una formulación brinda un beneficio a la búsqueda, se recompensa, en otro caso, se penaliza. Para inferir el desempeño de cada formulación se emplearon tres indicadores: métrica Δ, métrica C y contribución, propuesta en esta tesis.

En la comparación preliminar realizada en la Sección 6.5, se encontró que este método fue incapaz de superar a NSGA-II. Además, se observó que su desempeño tiende a deteriorarse en algunos problemas. Este deterioro se atribuye al nivel de granularidad de la población usada al evaluar cada formulación.

Además de no ser un esquema competitivo con el resto de los métodos analizados, se ha observado que el uso de diferentes formulaciones no brinda necesariamente una mejora en el desempeño

promedio de un algoritmo. Es decir, el diseño de un esquema de hibridación es relevante para hacer un mejor uso de diferentes formulaciones de manera simultánea.

Trabajo futuro

A continuación se describen algunas posibles ideas por realizar a futuro:

- **Uso simultáneo de diferentes operadores de dominancia**. El modelo de islas propuesto en esta tesis puede ser adaptado para emplear un operador de dominancia (o de otro tipo) por cada isla. En la literatura existen diferentes operadores de dominancia, tal como dominancia ϵ, dominancia α y dominancia por expansión y contracción [43]. De este modo, se podrían utilizar diferentes operadores con la finalidad de modificar la presión de selección en distintas etapas de la búsqueda o bien, emplear diferentes configuraciones de un mismo operador de manera simultánea.

- **Evaluación de diferentes esquemas de hibridación en problemas con muchos objetivos (mayor que tres)**. Este estudio se ha limitado a analizar tres esquemas de hibridación con $m \in \{2, 3, 5\}$ objetivos. Sin embargo, resulta importante conocer si dichos esquemas son apropiados para abordar problemas con un mayor número de objetivos.

Bibliografía

[1] Afanasyeva, A. and Buzdalov, M. (2011). Choosing best fitness function with reinforcement learning. In *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, volume 2, pages 354–357.

[2] Afanasyeva, A. and Buzdalov, M. (2012). Optimization with auxiliary criteria using evolutionary algorithms and reinforcement learning. In *Proceedings of the 18th International Conference on Soft Computing MENDEL 2012*.

[3] Alba, E. (2005). *Parallel Metaheuristics: A New Class of Algorithms*. Wiley-Interscience.

[4] Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., and Zitzler, E. (2007). Do Additional Objectives Make a Problem Harder? In *Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO '07*, page 765, New York, New York, USA. ACM Press.

[5] Buzdalov, M. and Buzdalova, A. (2013). Adaptive selection of helper-objectives for test case generation. *2013 IEEE Congress on Evolutionary Computation*, pages 2245–2250.

[6] Buzdalov, M., Buzdalova, A., and Shalyto, A. (2013). A First Step towards the Runtime Analysis of Evolutionary Algorithm Adjusted with Reinforcement Learning. *2013 12th International Conference on Machine Learning and Applications*, pages 203–208.

[7] Buzdalova, A. and Buzdalov, M. (2012). Adaptive Selection of Helper-Objectives with Reinforcement Learning.

[8] Candan, C., Goeffon, A., Lardeux, F., and Saubion, F. (2012a). A dynamic island model for adaptive operator selection. In *Proceedings of the fourteenth international conference on Genetic*

and evolutionary computation conference - GECCO '12, page 1253, New York, New York, USA. ACM Press.

[9] Candan, C., Goëffon, A., Lardeux, F., and Saubion, F. (2012b). A Dynamic Island Model for Adaptive Operator selection. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference - GECCO '12*, page 1253, New York, New York, USA. ACM Press.

[10] Candan, C., Goëffon, A., Lardeux, F., and Saubion, F. (2013). Parameter Setting with Dynamic Island Models. In Nicosia, G. and Pardalos, P., editors, *Learning and Intelligent Optimization*, Lecture Notes in Computer Science, pages 253–258. Springer Berlin Heidelberg, Berlin, Heidelberg.

[11] Coello, C., Lamont, G., and Van Veldhuizen, D. (2007a). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation Series. Springer US.

[12] Coello, C. A., Lamont, G. B., and Van Veldhuizen, D. A. (2007b). Basic Concepts. In *Evolutionary Algorithms for Solving Multi-Objective Problems*, Genetic and Evolutionary Computation Series. Springer US.

[13] Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Inc.

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

[15] Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2001). Scalable Test Problems for Evolutionary Multi-Objective Optimization. (1990):1–27.

[16] Derrac, J., García, S., Molina, D., and Herrera, F (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18.

[17] Eiben (2005). *Introduction to Evolutionary Computing*. Wiley-Interscience.

[18] Eiben, A., Horvath, M., Kowalczyk, W., and Schut, M. (2007). Reinforcement learning for online control of evolutionary algorithms. In Brueckner, S., Hassas, S., Jelasity, M., and Yamins, D., editors, *Engineering Self-Organising Systems*, volume 4335 of *Lecture Notes in Computer Science*, pages 151–160. Springer Berlin Heidelberg.

[19] Fogel, L. J., Owens, A. J., and Walsh, M. J. (1965). Artificial intelligence through a simulation. In *Biophysics and Cybernetic System*, pages 131–156.

[20] García, S., Molina, D., Lozano, M., and Herrera, F (2008). A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: acase study ontheCEC 2005 Special Session onReal Parameter Optimization. *Journal of Heuristics*, 15(6):617–644.

[21] Hernández Gómez, R. (2013). A New Multi-Objective Evolutionary Algorithm Based on the R2 Indicator.

[22] Holland, J. H. (1963). Genetic algorithms and the optimal allocation of trials. In *SIAM J. of Computing*, pages 88–105.

[23] Huband, S., Hingston, P., Barone, L., and While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506.

[24] Ishibuchi, H., Doi, T., and Nojima, Y. (2006a). Incorporation of Scalarizing Fitness Functions into Evolutionary Multiobjective Optimization Algorithms. In Runarsson, T. P., Beyer, H.-G., Burke, E., Merelo-Guervós, J. J., Whitley, L. D., and Yao, X., editors, *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 493–502. Springer Berlin Heidelberg.

[25] Ishibuchi, H., Hitotsuyanagi, Y., Nakashima, Y., and Nojima, Y. (2010). Multiobjectivization from Two Objectives to Four Objectives in Evolutionary Multi-Objective Optimization Algorithms. In *2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pages 502–507. IEEE.

[26] Ishibuchi, H. and Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 28(3):392–403.

[27] Ishibuchi, H., Nojima, Y., and Doi, T. (2006b). Comparison between Single-Objective and Multi-Objective Genetic Algorithms: Performance Comparison and Performance Measures. In *2006 IEEE International Conference on Evolutionary Computation*, number 1, pages 1143–1150. IEEE.

[28] Jaszkiewicz, A. (2002). On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment. *Evolutionary Computation, IEEE Transactions on*, 6(4):402–412.

[29] Jensen, M. T. (2004). Helper-Objectives: Using Multi-Objective Evolutionary Algorithms for Single-Objective Optimisation. *Journal of Mathematical Modelling and Algorithms*, 3(4):323–347.

[30] Karafotias, G., Eiben, A. E., and Hoogendoorn, M. (2014). Generic parameter control with reinforcement learning. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 1319–1326, New York, NY, USA. ACM.

[31] Knowles, J. D., Thiele, L., and Zitzler, E. (2006). A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Technical report.

[32] Knowles, J. D., Watson, R. A., and Corne, D. W. (2001). Reducing Local Optima in Single-Objective Problems by Multi-objectivization. In *Evolutionary Multi-Criterion Optimization*, pages 269–283.

[33] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

[34] Lardeux, F. and Goëffon, A. (2010). A Dynamic Island-Based Genetic Algorithms Framework. In *Simulated Evolution and Learning*, volume 6457, pages 156–165. Springer Berlin Heidelberg.

[35] Liu, H.-L., Gu, F., and Zhang, Q. (2014). Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *Evolutionary Computation, IEEE Transactions on*, 18(3):450–455.

[36] Louis, S. and Rawlins, G. J. E. (1993). Pareto Optimality , GA-easiness and Deception. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 118—123. Morgan Kaufmann.

[37] Luengo, J., García, S., and Herrera, F. (2009). A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions and non-parametric tests. *Expert Systems with Applications*, 36(4):7798–7808.

[38] McClymont, K. and Keedwell, E. (2011). Markov Chain Hyper-heuristic (MCHH): an Online Selective Hyper-heuristic for Multi-objective Continuous Problems. pages 2003–2010.

[39] Miettinen, K. M. (1999). *Nonlinear Multiobjective Optimization*. Kluwer Academic.

[40] Murata, T. and Ishibuchi, H. (1995). Moga: multi-objective genetic algorithms. In *Evolutionary Computation, 1995., IEEE International Conference on*, volume 1, pages 289–.

[41] Rao, S. S. (2009). Introduction to optimization. In *Engineering Optimization*, pages 1–62. John Wiley & Sons, Inc.

[42] Rechenber, I. (1973). Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.

[43] Sato, H., Aguirre, H., and Tanaka, K. (2007). Controlling Dominance Area of Solutions and Its Impact on the Performance of MOEAs. In Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., and Murata, T., editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 5–20. Springer Berlin Heidelberg.

[44] Segura, C., Coello Coello, C. A., Miranda, G., and León, C. (2013). Using multi-objective evolutionary algorithms for single-objective optimization. *4OR*, 11(3):201–228.

[45] Sutton, R. S. and Barto, A. G. (2012). Reinforcement Learning: An Introduction.

[46] van Otterlo, M. and Wiering, M. (2012). Reinforcement learning and markov decision processes. In Wiering, M. and van Otterlo, M., editors, *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 3–42. Springer Berlin Heidelberg.

[47] Weise, T., Chiong, R., and Tang, K. (2012). Evolutionary optimization: Pitfalls and booby traps. *Journal of Computer Science and Technology*, 27(5):907–936.

[48] Zhang, Q. and Li, H. (2007). MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731.

[49] Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis.

[50] Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary computation*, 8(2):173–95.

[51] Zitzler, E. and Thiele, L. (1998). An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. (43).

[52] Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.

Cinvestav
Tamaulipas

2014 | AÑO DE OCTAVIO PAZ

COORDINACIÓN ACADÉMICA

LABORATORIO DE
TECNOLOGÍAS DE INFORMACIÓN

Parque Científico y Tecnológico
TECNOTAM

Carretera a Soto la Marina
Km. 5.5 C.P. 87130,
Ciudad Victoria, Tamaulipas, México

Tel: +52 (834) 107 0220
Fax: +52 (834) 107 0224
E-mail: admin@tamps.cinvestav.mx

www.tamps.cinvestav.mx

Cd. Victoria, Tamaulipas, a 10 de diciembre de 2014.

Los abajo firmantes, integrantes del jurado para el examen de grado que sustentará el C. Auraham Camacho García, declaramos que hemos revisado la tesis titulada:

"Estudio sobre esquemas de hibridación para el uso simultáneo de diferentes formulaciones de un problema de optimización multi-objetivo"

Y consideramos que cumple con los requisitos para obtener el grado de Maestro en Ciencias en Computación.

Atentamente,

Dr. Ricardo Landa Becerra

Dr. Luis Gerardo de la Fraga

Dr. Gregorio Toscano Pulido