Christian von Lücken Martínez,
Benjamín Barán Cegla

Reporte técnico del estado del arte sobre algoritmos evolutivos basados en descomposición para problemas con muchos objetivos

Desde los algoritmos genéticos a la optimización con muchos objetivos

6 de Diciembre 2020

Facultad Politécnica - Universidad Nacional de Asunción (FP-UNA)
Consejo Nacional de Ciencias y Tecnología (CONACYT)

Prefacio

Este trabajo tiene como objetivo presentar el estado del arte de algoritmos evolutivos basados en descomposición para problemas con muchos objetivos. En este contexto, se presenta en un único material el desarrollo de los Algoritmos Evolutivos Multiobjetivo (*Multi-objective Evolutionary Algorithms*- MOEA) desde los primeros métodos aparecidos en la década de los 90s hasta los algoritmos para lidiar con problemas con muchos objetivos de optimización (*many-objective optimization problems (MaOP)*) de la actualidad considerando especialmente a los métodos basados en descomposición. Además de servir como una actualización de referencia el trabajo busca reflejar algunas áreas de investigación futura en el área y que pueden considerarse de relevancia.

El material consiste de 5 capítulos. El contenido de estos proviene, principalmente, de trabajos previos de los autores [von03, von16, vLBB14, vLBB19], clases impartidas a nivel de grado y postgrado así como abundante bibliografía reflejando los temas de investigación más recientes. En algunos capítulos se mantiene la estructura original de los trabajos previos, enriquecido con nuevas referencias, gráficos, tablas y apuntes relevantes. Además, el contenido ha sido integrado, actualizado y revisado con el fin de ser utilizado como material base para un libro de texto que pueda ser utilizado en próximos cursos así como servir de guía para el inicio de la investigación en el área. Cabe destacar que, de acuerdo a lo mejor de nuestro conocimiento, el presente trabajo es el primero en desarrollar de manera completa el desarrollo de los algoritmos evolutivos multi-objetivo hasta los métodos basados en descomposición. Además, como un valor agregado, al encontrarse en español una versión revisada del presente trabajo podrá acercar el área al público de habla hispana.

El Capítulo 1 presenta una revisión histórica general de los algoritmos multiobjetivo. El capítulo inicia describiendo el área de investigación donde se ubican los algoritmos evolutivos, luego se presentan los principales operadores, y el teorema de esquemas (una explicación sobre el funcionamiento de los Algoritmos Genéticos). Luego, se presentan brevemente otros operadores para luego explicar los aspectos claves de la introducción de los algoritmos evolutivos y las diversas generaciones de los mismos.

En el Capítulo 2 se formalizan los principios de optimización con objetivos múltiples en el contexto de los algoritmos evolutivos, particularmente la definición de optimalidad de Pareto, la noción de óptimo comúnmente aceptada en problemas que consideran varios criterios de manera simultánea. Así mismo, se describe el modelo general del proceso de solución utilizado con problemas de optimización multi-objetivo. Por último, basados en [vLBB14] se presenta el tema de la influencia de la existencia de muchos objetivos a ser considerados de manera simultánea en la resolución de problemas utilizando algoritmos evolutivos, así como el problema de visualización.

El Capítulo 3 presenta los algoritmos Evolutivos para objetivos múltiples que corresponden a la primera y segunda generación y que han sido evaluados por el autor en [von03]. Luego, en el Capítulo 4 se introducen los algoritmos de tercera generación desarrollados para resolver problemas con muchos objetivos. El contenido de este capítulo esta basado en [vLBB14, von16] con modificaciones menores. El objetivo de estos capítulos es brindar un marco de completitud al material presentado.

El Capítulo 5 reporta de manera específica los algoritmos basados en descomposición, los avances y las posibilidades de investigación que derivan de ellos. Estos algoritmos han recibido especial atención por parte de la comunidad de investigadores en los últimos años [OEBY20] y, por lo tanto, este capítulo ha sido especialmente dedicado a estos. Se presentan aquí varios algoritmos que caen en esta categoría y que han sido desarrollados y estudiados con especial énfasis en los últimos años [Hug03, ZL07, LGZ14, DJ14a, CJOS16]. Además, se discute brevemente algunas alternativas de trabajo futuro.

Este trabajo fue preparado en el marco del proyecto de investigación PINV18-949 del CONACYT y la FPUNA.

Diciembre 2020, *Christian Daniel von Lücken*
Benjamín Barán

Contenido

Capítulo 1
De los algoritmos genéticos a los algoritmos evolutivos para problemas con muchos objetivos

Uno de los temas de mayor actualidad en el área de la la inteligencia artificial (*Artificial Intelligence* - AI) busca responder a la pregunta sobre cómo construir sistemas dotados de un comportamiento inteligente, que sean capaces de obtener soluciones óptimas para una amplia gama de problemas a partir de una descripción de alto nivel de los mismos. Para dar respuesta a esta pregunta, una de las ramas de la AI trata con la resolución de problemas utilizando como técnica básica la exploración de un gran número de posibilidades en la búsqueda de soluciones. La importancia de la búsqueda para la AI radica en que muchos problemas del mundo real que implican esfuerzo intelectual pueden plantearse en forma directa como problemas de búsqueda.

Para poder construir las cajas negras capaces de solucionar una amplía gama de problemas de manera robusta, es deseable contar con algoritmos que requieran un mínimo de información acerca del espacio de búsqueda explorado. Debido a que la información sobre el dominio del problema con que estos pueden contar es débil, en el sentido de ser incompleta, parcial, inexacta o con cierto grado de incertidumbre, a los algoritmos desarrollados para tratarlos se los agrupa en el área conocida como computación blanda o *Soft Computing* (SC).

Como se señala en [Ray15], el término *soft computing* fue acuñado por Lotfi A. Zadeh a comienzo de los noventas en contraposición a la computación dura donde el objetivo principal es la precisión, la certeza y el rigor. La computación blanda, en cambio, se basa en la premisa que la precisión y la certeza tienen un alto costo y que, en la medida de lo posible, el cálculo, el razonamiento y la toma de decisiones deben aprovechar la tolerancia a la imprecisión y la incertidumbre que existe en un gran número de problemas reales en la ciencia e ingeniería. De esta forma, estas técnicas pueden ser de gran utilidad en aplicaciones del mundo real donde una solución donde obtener una solución óptima no es viable mientras que una sub-óptima es suficiente.

Los métodos débiles tienen en general la dificultad que, para muchos problemas, el espacio de búsqueda crece exponencialmente con respecto al tamaño de la instancia considerada, siendo impráctico examinar las distintas posibilidades sin incorporar información adicional. En estos casos, es deseable la utilización de métodos que además de precisar poco conocimiento a priori, sean capaces de generar por

sí mismos la información adicional que permita orientar el proceso de exploración a fin de comportarse de forma eficiente en espacios de búsqueda complejos y de alta dimensionalidad. Para abordar esta problemática, durante años, se ha estado investigando la manera en la que se desarrollan los procesos naturales y el comportamiento de los seres vivos. En base a estas investigaciones se propusieron una serie de técnicas que en su conjunto se conocen como computación bio-inspirada o computación inspirada en la naturaleza.

Entre los algoritmos bio-inspirados se encuentran los que se conocen como basados en población, los cuales trabajan con un conjunto de soluciones e intentan encontrar soluciones por medio de la mejora iterativa de este conjunto de soluciones. De acuerdo a la naturaleza del fenómeno simulado por el algoritmo, los algoritmos de optimización basados en población pueden clasificarse en dos grupos principales: los algoritmos evolutivos (*Evolutionary Algorithms* (EA) y los algoritmos de inteligencia de enjambre (*Swarm Intelligence*) [KB07]. Una visión general de las diferentes técnicas y aplicaciones de la computación inspirada en la naturaleza se encuentra en [MDL19].

La inteligencia de enjambre intenta modelar la población de agentes que interactúan entre sí con la capacidad de auto-organizarse. Entre las técnicas de inteligencia de enjambre más representativas están:

- La optimización por colonia de hormigas (*Ants Colony Optimization* - ACO) [DMC96], que simula el proceso natural que utilizan las hormigas para construir el camino desde el nido hasta el alimento.
- Las colonias de abejas artificiales (*Artificial Bees Colony* - ABC) [KB07]: que simulan la danza de las abejas para indicar a otras la ubicación del alimento.
- La optimización por enjambre de partículas (*Particle swarm optimization* - PSO) de Eberhart y Kennedy [KE95] que simula el comportamiento de bandadas de aves o cárdumenes de peces en su desplazamiento hacia un destino.

Los algoritmos evolutivos son métodos basados en poblaciones que simulan el proceso de la evolución natural para encontrar soluciones y, siendo el tema principal del presente trabajo, se describen en la siguiente sección.

1.1 Algoritmos evolutivos

Los algoritmos evolutivos consideran la metáfora de la evolución como una búsqueda que opera sobre una población para encontrar, en el conjunto de secuencias de genes, aquellas que correspondan a los mejores individuos de forma a propagar estas características en las futuras generaciones. Esta búsqueda es robusta, puesto que permite localizar en un espacio de posibles soluciones enorme y cambiante, de forma eficaz y eficiente, a aquellos individuos que mejor se adapten al medio. Las especies evolucionan por medio de la selección natural, esto es, la conservación de las diferencias y variaciones individualmente favorables y la posible destrucción de las que son perjudiciales.

En la naturaleza, los individuos que posean algún tipo de ventaja evolutiva sobre otros tienen tendrán más probabilidades de sobrevivir y procrear su especie. Los hijos son semejantes a sus padres, por lo que cada nueva generación tiene individuos semejantes a los mejores individuos de la generación anterior. De esta forma, las características genéticas individuales que resulten positivas se propagan en el tiempo, mientras que las perjudiciales se van perdiendo. Además, se producen mutaciones al azar de manera ocasional. Estas mutaciones pueden introducir características mejoradas, las cuales serán propagadas a las futuras generaciones. Por el contrario, las características negativas introducidas por la mutación tenderán a desaparecer pues conducen a individuos pobremente adaptados. Cuando las modificaciones del ambiente son relativamente menores, esto permite a las especies ir evolucionando gradualmente con éste; sin embargo, es muy probable que la existencia de un cambio súbito de ambiente provoque la desaparición de especies enteras y el nacimiento de otras nuevas, capaces de sobrevivir en el nuevo entorno.

Entonces, los algoritmos evolutivos son algoritmos de búsqueda y optimización que basados en los mecanismos esenciales de la evolución Darwiniana, intentan reproducir las características de robustez y simplicidad existentes en la naturaleza para evolucionar hacia mejores soluciones en problemas computacionales [Gol89].

Los algoritmos evolutivos se caracterizan por trabajar con un conjunto de soluciones candidatas del problema que se intenta resolver. Estas soluciones representan individuos y el conjunto de ellos a una población. Como sabemos, en los entornos naturales, la información relativa a las características de los individuos viene dada en cadenas de genes llamados cromosomas. Los genes son responsables de la carga hereditaria. Por ello, para la realización de búsquedas utilizando algoritmos evolutivos, se requiere la codificación de las soluciones representando a las cadenas de genes o cromosomas (representación genética) y la aplicación cíclica y secuencial de un conjunto de operadores que imitan los procedimientos básicos de la evolución darwiniana a un conjunto de soluciones.

Los operadores evolutivos básicos son la asignación de aptitud (*fitness*), la selección, la recombinación o cruzamiento, y la mutación. La aplicación de estos operadores de forma iterativa permite que las características de las mejores soluciones, también llamadas individuos, se propaguen durante la ejecución de un algoritmo evolutivo (*Evolutionary Algorithm* - EA). Al conjunto de individuos se lo conoce como población. La Figura 1.1 muestra el flujo típico de un EA básico.

Las primeras ideas de búsquedas genéticas o evolutivas aparecieron a principios de los 50's como una curiosidad científica. El primero en establecer una relación entre inteligencia de máquina y la búsqueda fue Allan Turing cuando a fines de la década del 40 identificó varios enfoques que podrían ser utilizados para crear programas inteligentes a partir de búsquedas [Tur95]. Los paradigmas principales de los algoritmos evolutivos se establecieron en la década de los 60s: los Algoritmos Genéticos (*Genetic Algorithm - GA*) [Bre62, Hol62, Hol75], las Estrategias de Evolución (*Evolution Strategies - ES*) [Rec65, Sch65] y la Programación Evolutiva (*Evolution Programming - EP*) [Fog62]. Aunque cada uno de estos tiene sus características particulares como la representación o los operadores que utilizan, en

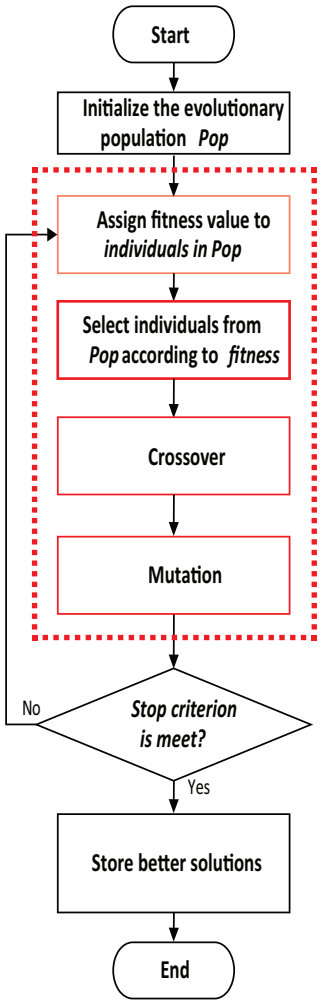

Fig. 1.1 Flujo de un Algoritmo Evolutivo (EA) Básico.

la actualidad existen un gran número de algoritmos evolutivos que las combinan, siendo difuso el límite entre éstos.

Los algoritmos genéticos pueden considerarse como los Evolutionary Algorithm (EA) que han recibido mayor atención de la comunidad científica, por ello en algunos casos se utiliza el término algoritmo genético como sinónimo de algoritmo evolutivo existiendo un gran número de aplicaciones en las que se utilizan estos algoritmos en sus diversas variantes. Algunos ejemplos de estas aplicaciones son el diseño industrial [SGCGG19] así como las aplicaciones de ingeniería [SK20], entre otros. Los algoritmos genéticos enfatizan el operador de cruzamiento como el operador de búsqueda más importante y aplican la mutación con una pequeña probabilidad

como un operador de segundo plano. Además, en estos algoritmos se utiliza un operador de selección probabilístico. Los primeros algoritmos genéticos utilizaban la codificación de las soluciones como cadenas de números binarios aunque hoy pueden encontrarse representaciones muy diversas.

1.2 Codificación y función de adaptación

En general, para resolver un problema utilizando una computadora, primeramente buscamos la secuencia de pasos que deben ejecutarse para resolverlo (el algoritmo). Luego, utilizando algún lenguaje de programación, se codifica esta secuencia de pasos en un conjunto de instrucciones que se han de ejecutar. Cuando se trabaja con algoritmos evolutivos, sin embargo, la preocupación principal no es acerca de qué pasos conducirán a la solución del problema sino más bien en la manera de representar las soluciones. Es el algoritmo evolutivo el que se encargará de evolucionar la solución. Así, la computación evolutiva representa un cambio radical con respecto a la manera tradicional de enfrentar los problemas computacionales [Fog06].

Cuando se utilizan las técnicas de computación evolutiva, las posibles soluciones son codificadas de manera tal que cada una de las variables del problema de optimización pueda ser vista como un gen o un conjunto de genes, de forma que en su totalidad represente a los cromosomas. Los parámetros del problema se codifican ya sea como una cadena de números binarios [Gol89], números reales [SMO01], letras del alfabeto, listas, etc. Además, se pueden combinar diferentes tipos de representaciones ya que cada parámetro puede tener una codificación independiente de acuerdo a la conveniencia específica de cada problema. Una descripción sobre diferentes posibilidades de representación utilizando algoritmos genéticos y evolutivos se encuentra en [Rot06]. En analogía con la naturaleza, cada parámetro del problema codificado representa un gen y considerados en conjunto, forman una cadena de genes o cromosoma. Los elementos en el conjunto de parámetros son denominados usualmente genotipos, mientras que los elementos en el espacio objetivo suelen denominarse fenotipos. Las definiciones 1.1 al 1.5 formalizan estas ideas [BFM97, von03].

Definición 1.1 *Codificación:* Sea $\mathcal{S}$ un espacio de búsqueda, i.e. una colección de objetos sobre el cual se realizará la búsqueda. Sean $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ conjuntos finitos arbitrarios y sea:

$$\mathcal{I} = \mathcal{A}_1 \times \mathcal{A}_2, \ldots, \times \mathcal{A}_n$$

Finalmente sea

$$\mathsf{g} : \mathcal{I} \rightarrow \mathcal{S}$$

una función que mapea vectores en $\mathcal{I}$ a soluciones en el espacio de búsqueda. Se dice que el par $(\mathcal{I}, \mathsf{g} : \mathcal{I} \rightarrow \mathcal{S})$ es una codificación o representación de $\mathcal{S}$. El conjunto $\mathcal{I}$ es denominado espacio de codificación, la función g es llamada la función de mapeo.

Entonces, n es el número de variables utilizadas para mapear una solución de $\mathcal{S}$ en $\mathcal{I}$. □

Definición 1.2 *Alelo:* Los conjuntos $\mathcal{A}_i$, $i \in \{1, \ldots, n\}$, a partir de los cuales se compone $\mathcal{I}$ en la definición 1.1, es llamado el conjunto de alelos y sus miembros alelos.

Definición 1.3 *Cromosoma:* Sea la codificación $(\mathcal{I}, g : \mathcal{I} \rightarrow \mathcal{S})$, los miembros de $\mathcal{I}$ son llamados cromosomas y menos comúnmente genomas o genotipos. Un cromosoma $\mathbf{x} \in \mathcal{I}$ puede ser escrito como:

$$(x_1, x_2, \ldots, x_n) \in \mathcal{A}_1 \times \mathcal{A}_2, \ldots, \times, \mathcal{A}_n$$

o como una cadena $x_1 x_2 \ldots x_n$.

Definición 1.4 *Gen:* Los componentes x_i de $\mathbf{x}$, cuando se tratan como variables, son conocidos como genes, tal que el i-ésimo gen toma sus valores del conjunto de alelos $\mathcal{A}_i$.

Definición 1.5 *Locus:* La posición i de un gen en el cromosoma es conocida como su *locus*. □

Como ejemplo, considerando el espacio de búsqueda $\mathcal{S} = \{0, 1, 2, \ldots, 9\}$ y un espacio de representación $\mathcal{I} = \mathcal{A}_1 \times \mathcal{A}_2$, donde $\mathcal{A}_1 = \{0, 1\}$ y $\mathcal{A}_2 = \{0, 1, 2, 3, 4\}$. Una función de mapeo $g : \mathcal{I} \rightarrow \mathcal{S}$ puede definirse adecuadamente como:

$$g(x_1, x_2) = 5 \cdot x_1 + x_2$$

donde x_1 toma sus valores del conjunto de alelos $\mathcal{A}_1$ y x_2 toma sus valores del conjunto de alelos $\mathcal{A}_2$. Entonces, para la codificación $(\mathcal{I}, g : \mathcal{I} \rightarrow \mathcal{S})$, un cromosoma $\mathbf{x} = x_1 x_2 = 13$ en $\mathcal{I}$ representa a la solución $8 \in \mathcal{S}$. Además, se dice que el gen con *locus* 1 (x_1), tiene como valor el alelo 1 y el gen con *locus* 2 tiene como valor el alelo 3 [BFM97].

En muchas representaciones, todos los genes tienen la misma cardinalidad y comparten un mismo conjunto de alelos. En la mayoría de las aplicaciones de GAs, las variables de decisión son codificadas utilizando representaciones binarias, en las cuales todos los genes tienen los alelos 0 o 1. Esto es, $\mathcal{A}_i = \{0, 1\}$ para todo $i \in \{1, \ldots, n\}$.

La representación binaria es adecuada para los problemas llamados problemas de optimización combinatoria seudo-Boléanos de la forma $f : \{0, 1\}^n \rightarrow \mathbb{R}$. En éstos problemas, el espacio de búsqueda ($\{0, 1\}^n$) puede ser representado en forma directa por cadenas binarias de longitud $l = n$. Es decir, para este caso se tiene $\mathcal{S} = \mathcal{I}$. Algunos ejemplos de tales problemas son el problema del conjunto independiente máximo en grafos [KB94] o el problema de la mochila [KBH94], los cuales pueden ser representados por vectores binarios simplemente incluyendo (o excluyendo) un

vértice o item i en (de) una solución candidata cuando la entrada correspondiente se hace $x_i = 1$ ($x_i = 0$).

La representación binaria también ha sido muy utilizada en el caso de problemas $f : S \rightarrow \mathbb{R}$ donde el espacio de búsqueda S es fundamentalmente distinto del espacio vectorial binario $\{0,1\}^l$. Los mecanismos para codificar y decodificar entre los dos espacios diferentes $\{0,1\}^l$ y $\mathbb{R}^n$ restringen el espacio continuo a intervalos finitos $[u_i; v_i]$ para cada variable $x_i \in \mathbb{R}$, dividiendo el vector binario de tamaño l en n segmentos. Por lo general estos segmentos tienen igual longitud l_x, tal que $l = n \cdot l_x$. Siendo a_j el valor (0 o 1) específico en la posición j de la cadena binaria, entonces, se interpreta el subsegmento $(a_{(i-1)\cdot l_x+1}, \ldots, a_{i\cdot l_x})(i = 1, \ldots, n)$ como la codificación binaria de la variable x_i. Entonces, la decodificación del i-ésimo gen representado en forma binaria con una subcadena de tamaño $l_i = l_x$ se realiza de acuerdo a la función de decodificación binaria $\Gamma^i : \{0,1\}^{l_x} \rightarrow [u_i; v_i]$, donde [BFM97]:

$$\Gamma^i(a_{i\cdot 1}, \ldots, a_{i\cdot l_x}) = u_i + \frac{v_i - u_i}{2^{l_x} - 1}\left(\sum_{j=0}^{l_x-1} a_{i\cdot(l_x-j)} 2^j\right) \tag{1.1}$$

Por ejemplo, una variable real x_1 en el rango $[0.5\ ;\ 16]$ puede ser codificada en cadenas de 5 bits. Entonces, utilizando la ecuación 1.1, las cadenas (00000) y (11111) representan los valores reales 0.5 y 16 respectivamente, mientras que cualquiera de las otras 30 cadenas posibles representan una solución en el intervalo $(0.5\ ;\ 16)$:

$$\begin{aligned}
\Gamma^1(00000) &= 0.5 + \frac{16 - 0.5}{2^5 - 1} \cdot 0 = 0.5 \\
\Gamma^1(00001) &= 0.5 + \frac{16 - 0.5}{2^5 - 1} \cdot (1 \cdot 2^0) = 1.0 \\
\Gamma^1(00010) &= 0.5 + \frac{16 - 0.5}{2^5 - 1} \cdot (1 \cdot 2^1 + 0 \cdot 2^0) = 1.5 \\
\Gamma^1(00011) &= 0.5 + \frac{16 - 0.5}{2^5 - 1} \cdot (1 \cdot 2^1 + 1 \cdot 2^0) = 2.0 \\
&\vdots \\
\Gamma^1(11111) &= 0.5 + \frac{16 - 0.5}{2^5 - 1} \cdot 31 = 16.0
\end{aligned}$$

Luego de definir la codificación, cuando se diseña un AE se debe establecer la forma en la cual, una vez evaluada esta solución, se relacionará el valor del objetivo (o los objetivos) con el valor de adaptación del individuo respecto a la población. Idealmente, a partir de un cromosoma dado la función de adaptación debe retornar un valor proporcional a la utilidad de la solución representada por tal cromosoma. Es decir, cuanto mejor es la solución representada por un cromosoma, mejor debe ser su valor de adaptabilidad. La forma de calcular este valor de adaptación debe ser, idealmente, independiente del problema considerado. Básicamente, dado cualquier par de elementos a y b en el espacio de codificación representando soluciones en

el espacio de búsqueda, la función de adaptación debe asignar un valor mayor a a respecto al asignado a b cuando a es una mejor solución que b. Para ciertos problemas, es fácil determinar que función de adaptación debería ser utilizada; sin embargo, este no es el caso de todos los problemas prácticos.

La función de adaptación se puede definir formalmente como:

Definición 1.6 *Función de adaptación:* Sea el espacio de búsqueda $\mathcal{S}$ y sea la codificación $(\mathcal{I}, \mathsf{g} : \mathcal{I} \rightarrow \mathcal{S})$. Entonces, una función de adaptación es cualquier función

$$\mathsf{f} : \mathcal{I} \rightarrow \mathbb{R} \tag{1.2}$$

tal que, cumpla con la propiedad que

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{I}, \mathsf{f}(\mathbf{x}) > \mathsf{f}(\mathbf{x}'), \text{ si } \mathsf{g}(\mathbf{x}) \text{ se considera mejor que } \mathsf{g}(\mathbf{x}') \text{ en } \mathcal{S} \tag{1.3}$$

Es decir, dado cualquier par de cromosomas $\mathbf{x}$, $\mathbf{x}'$, al evaluar la función de adaptabilidad (en inglés *fitness*) sobre estos, si $\mathbf{x}$ es una mejor solución al problema que $\mathbf{x}'$, el valor de adaptación de $\mathbf{x}$ será mejor que el de $\mathbf{x}'$.

A partir de una codificación dada, podemos definir individuo y población de manera formal como:

Definición 1.7 *Individuo:* Sea la codificación $(\mathcal{I}, \mathsf{g} : \mathcal{I} \rightarrow \mathcal{S})$, y sea el cromosoma $\mathbf{x} \in \mathcal{I}$, se utiliza el término individuo para referenciar una estructura de datos compuesta por un cromosoma y otras propiedades, como por ejemplo su valor de adaptabilidad f. □

Definición 1.8 *Población:* Una población es una estructura de datos que representa un grupo de individuos. Generalmente se representa a una población por medio de un vector de individuos.

Para continuar, resulta conveniente introducir algunas notaciones. El conjunto de soluciones o población utilizado por los EAs para simular el proceso evolutivo en este trabajo se representa con la letra P. Como se verá posteriormente, esta población cambia, generación a generación, conforme el proceso simulado de evolución avanza. Para representar esto se denota como $P(t)$ a una población P en una generación particular t. Por simplicidad, esta notación se utilizará sólo cuando es preciso señalar el número t de generaciones transcurrido desde el inicio de la ejecución del algoritmo. En otros casos, se utilizará sólo P. A un individuo que se encuentra en una posición cualquiera i de la población $P(t)$ se le representa, de acuerdo a la conveniencia, ya bien utilizando un subíndice como $P_i(t)$ o utilizando la notación $P[i](t)$. Igualmente, se utiliza sólo P_i o $P[i]$ cuando no resulta necesario indicar el número de generación. Las propiedades de los individuos varían de acuerdo al algoritmo y la representación considerada. Como un ejemplo de la notación utilizada, el valor de adaptación de un individuo se representa con el subíndice $fitness$. Así, el valor de adaptación del individuo en la posición i de la población P, puede representarse como: $P_{i_{fitness}}$ o más convenientemente como $P[i]_{fitness}$; mientras que el respectivo cromosoma como: $P[i]_{chrom}$. El tamaño de una población genética P se representa con N.

1.3 Operadores evolutivos básicos

La literatura reporta varios operadores de selección como la selección de muestreo uniforme, por torneo, clasificación o estocástico [Sai17, SPM15]. Se reportan, además, varias opciones para el cruzamiento como el cruce de un punto, dos puntos, uniforme [PG16, US15, KSR+20, KY17]. Igualmente, existen diferentes operadores de mutación como de uno o más puntos, de rotación, etc. [LSS+17, KSR+20]. Estos operadores están relacionados generalmente al tipo de codificación de las soluciones o bien están desarrollados para la resolución de cierto problema o conjunto de problemas específicos .

Aún teniendo en cuenta la existencia de múltiples alternativas de operadores, se puede considerar al método de asignación de valor de aptitud como el elemento diferenciador más importante entre los diferentes algoritmos evolutivos que han sido desarrollados. Los primeros algoritmos evolutivos fueron desarrollados para resolver problemas con un único objetivo (mono-objetivo). En estos problemas el espacio de búsqueda se encuentra ordenado por lo que es posible comparar las soluciones a partir del valor de la función objetivo y determinar que una solución es mejor que otra. Resulta relativamente simple transformar este valor de la función objetivo en una medida del desempeño y utilizar esto como valor de aptitud.

Los individuos en las poblaciones intercambian información por medio de operadores evolutivos. Existen tres operadores genéticos principales: selección, cruzamiento y mutación. Las siguientes secciones explican en forma breve cada uno de ellos.

1.3.1 Selección

La selección es un proceso por medio del cual los individuos son escogidos de la población de acuerdo al valor de su función de adaptación para someterse a la acción futura de otros operadores. Entre los métodos de selección más utilizados están la selección vía ruleta (*roulette-whell selection*) y la selección por torneo (*tournament selection*) [Gol89].

El procedimiento de selección por medio de ruleta es la estrategia de selección más simple. Este procedimiento se muestra en el Algoritmo 1. En la selección por ruleta, la probabilidad de que un individuo sea seleccionado se calcula en forma directa a partir del valor de adaptación del individuo. Básicamente consiste en la construcción de una ruleta donde a cada individuo se le asigna una fracción de la misma en proporción directa a su adaptabilidad. Luego se selecciona en forma aleatoria una posición en la ruleta, el individuo al cual le corresponda dicha posición es el que será seleccionado. Entonces la probabilidad de selección del individuo $P[i]$ como padre es:

$$p_s(i) = \frac{P[i]_{fitness}}{\sum_{j=1}^{N} P[j]_{fitness}} \tag{1.4}$$

Otro método de selección comúnmente utilizado es el método de selección por torneo (*tournament selection*). En la selección por torneo un número dado de individuos de la población es preseleccionado en forma aleatoria. Estos individuos se comparan unos con otros de acuerdo a su valor de adaptación de una manera similar a la efectuada en competencias deportivas. Aquel con mejor valor de adaptación es seleccionado. Cuando el número de individuos elegidos es dos, a este proceso de selección se le llama selección por torneo binario. Ajustando el tamaño del torneo se puede obtener algún control sobre la presión de selección y por tanto de la velocidad de convergencia. El menor tamaño de torneo (torneo binario) exhibe una convergencia más lenta que cualquier tamaño de torneo mayor. El Algoritmo 2 señala los pasos de una selección por torneo de tamaño k.

Algoritmo 1: Algoritmo de selección por ruleta.

```
1 sum = ∑_{j=1}^{N} P[j]_fitness;
2 r = rnd(0, sum) ;      // rnd retorna un número aleatorio entre 0 y sum
3 ;
4 para j = 1 to N hacer
5 |   r = r − P[j]_fitness;
6 |   si r ≤ 0 entonces
7 |   |   return P[j];
8 |   fin
9 fin
```

Algoritmo 2: Algoritmo de selección por torneo.

```
1  Sea T una lista de elementos, pop(i) el elemento i de la población pop, n el tamaño de la
   población, k un número par de elementos de la población que participarán en la
   competencia
2  para j = 1 to k hacer
3  |   r = rnd(1, n)
4  |   T = T ∪ pop(r)
5  fin
6  para l = k/2 to l >= 1 hacer
7  |   para j = 1 to j <= l hacer
8  |   |   si fitness(T_{j+1}) < fitness(T_{j+l}) entonces
9  |   |   |   T_{l+j} = T_{l+j}
10 |   |   fin
11 |   |   j = j + 1
12 |   fin
13 |   l = l/2
14 fin
15 retornar T_{l+j}
```

1.3.2 Cruzamiento

El proceso de cruzamiento se inicia a partir de la obtención, utilizando algún operador de selección, de los individuos que serán sometidos al operador de cruzamiento que se encargará de intercambiar los componentes de los individuos seleccionados para producir nuevas soluciones. Así, el operador de cruzamiento se encarga de la transferencia de las características de los mejores individuos de una generación a la siguiente por herencia, siendo de esta manera la contraparte artificial de la recombinación sexual.

Existen varias opciones para implementar el operador de cruzamiento siendo el cruzamiento de un sólo punto la forma más común de implementar este operador cuando se trabaja con una codificación de cadenas de bits. Este tipo de cruzamiento requiere la selección de dos cadenas como padres, luego, se selecciona un punto de cruzamiento, de forma tal que dicho punto se encuentre entre 1 y $(l-1)$, donde l es la longitud de la cadena. Los dos padres son cortados en este punto para producir cuatro subcadenas, que se intercambian para generar dos hijos. La figura 1.2 muestra este procedimiento.

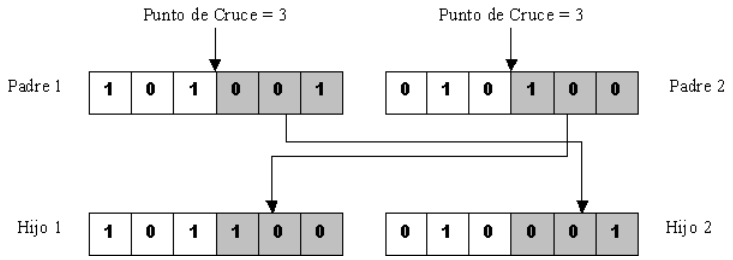

Fig. 1.2 Funcionamiento del operador de cruzamiento de un sólo punto.

1.3.3 Mutación

En los GAs la mutación es utilizada como un operador de segundo plano cuyo propósito es la exploración aleatoria de nuevas porciones del espacio de búsqueda. Es este operador el encargado de introducir nuevo material genético en la búsqueda de soluciones ya que el cruzamiento no introduce ningún material nuevo.

Cuando la codificación utilizada es binaria el operador de mutación simplemente modifica el valor de un bit en la cadena con una probabilidad de mutación dada.

1.4 Una explicación sobre el funcionamiento de los GA: Teorema de esquemas

Hasta aquí hemos presentado los operadores básicos que conforman a los algoritmos evolutivos y, también, se ha explicado de manera informal el porqué del éxito de los EA basados en estos operadores. Se puede decir que la mayor parte de la investigación sobre algoritmos evolutivos se ha concentrado en la realización de pruebas experimentales y métricas para comparar los resultados obtenidos. El desarrollo de una teoría general ampliamente aceptada que explique de forma exacta el comportamiento de los distintos algoritmos evolutivos que han sido desarrollados es aún un tema pendiente de investigación a ser abordado, aunque existen varias hipótesis que han sido desarrolladas y que pueden ser utilizadas parcialmente para explicar el éxito de los GAs en la búsqueda de soluciones.

Entre las hipótesis que explican el comportamiento de los GA, y en general de los algoritmos evolutivos, se destaca el el teorema de esquemas (*schemata*) [Hol75]. Esta explicación fue el primer intento de brindar una explicación rigurosa sobre cómo funcionan los GAs y se popularizó como tal a partir del trabajo seminal de difusión de Goldberg [Gol89]. En esta sección, con algunas modificaciones menores, presentamos esta explicación como se realiza en [von03].

Un esquema (*schema*) es un patrón que describe un conjunto de cadenas de genes con similaridades en ciertas posiciones de la cadena y puede definirse formalmente como:

Definición 1.9 *Esquema:* Sea un espacio de representación $\mathcal{I} = \mathcal{A}_1 \times \mathcal{A}_2, \ldots, \times \mathcal{A}_n$, para cada conjunto de alelos $\mathcal{A}_i$, se define un conjunto extendido $\mathcal{A}_i^\star = \mathcal{A}_i \cup \{\star\}$, donde $\star$ es un símbolo especial "no importa". Entonces, un esquema es cualquier miembro del conjunto $I^+ = \mathcal{A}_1^\star \times \mathcal{A}_2^\star, \ldots, \times \mathcal{A}_n^\star$.

Un esquema $H = (h_1, h_2, \ldots, h_n)$ describe un conjunto de cromosomas que tiene los mismos alelos que H en todas las posiciones i tal que $h_i \neq \star$. Para una cadena binaria, esto es:

$$H = \{x \in \mathcal{I} | \forall i \in \{1, 2, \ldots, n\} \ , \ h_i \in \{0, 1, \star\}\}.$$

Las posiciones en donde H no tiene un símbolo $\star$ se llaman posiciones fijas. □

Además, para poder trabajar sobre la idea de esquemas y similaridades entre distintos esquemas se definen dos métricas. El orden de un esquema, que es el número de posiciones fijas y el tamaño de definición, que es la distancia entre el primero y el último de los símbolos fijos. Las definiciones 1.10 y 1.11 describen formalmente las métricas mencionadas.

Definición 1.10 *Orden de un esquema:* El orden de un esquema $H \in I^+$, denotado por $o(H)$, es el número de posiciones fijas presentes en el esquema.

Definición 1.11 Tamaño de definición de un esquema: El tamaño de definición de un esquema $H \in I^+$, denotado por $\delta(H)$, es la distancia entre la primera y última posición de la cadena con símbolos fijos . □

Como ejemplo, consideremos el alfabeto binario $\{0, 1\}$, al cual se le agrega el símbolo especial $\star$ obteniendo $\{0, 1, \star\}$, un esquema es una cadena sobre este alfabeto. El esquema $H = \star 11\star$, describe un subconjunto de S con cuatro elementos $\{0110, 0111, 1110, 1111\}$. El orden de H ($o(H)$) es 2 puesto que posee dos posiciones fijas, mientras que su distancia de definición ($\delta(H)$) es 1 ya que el primer valor fijo se encuentra en la posición 2 y el último en la 3. El esquema $\star\star\star 1$ corresponde a cualquier cadena de tamaño 4 que termina con un 1. Por ejemplo, $\delta(1**) = 0$, $\delta(1**1) = 3$. El número de esquemas diferentes para una cadena de tamaño l y un alfabeto de cardinalidad k es $(k+1)^l$.

El objetivo del teorema de esquemas es analizar el efecto que tienen los operadores evolutivos en la búsqueda de soluciones y proveer un límite inferior sobre los cambios en la tasa de muestreo para un esquema, de una generación a otra. En un dado tiempo t existen m elementos de un esquema particular H en un población genética $P(t) = \{P_1, \ldots, P_N\}$, esto se representa por $\mathsf{m} = \mathsf{m}(H, t)$. Sea i_H la representación del índice de un individuo P_j en el conjunto $\{H \wedge P(t)\}$, $i_H \in \{1, \ldots, \mathsf{m}\}$ y f_{i_H} su correspondiente valor de adaptación. Aplicando a la población una selección por medio de ruleta ponderada, esperamos tener en el conjunto de padres un número de elementos del esquema H igual a:

$$\begin{aligned}\mathsf{m}(H, t+1) &= \frac{\mathsf{f}_{1_H}}{\sum_{j=1}^{N} P[j]_{fitness}} \cdot N + \cdots + \frac{\mathsf{f}_{\mathsf{m}_H}}{\sum_{j=1}^{N} P[j]_{fitness}} \cdot N \\ &= \frac{N}{\sum_{j=1}^{N} P[j]_{fitness}} \cdot (\mathsf{f}_{1_H} + \cdots + \mathsf{f}_{\mathsf{m}_H})\end{aligned}$$

$$= \frac{N}{\sum_{j=1}^{N} P[j]_{fitness}} \cdot (\sum_{i_H=1}^{m} \mathsf{f}_{i_H}) \cdot \frac{\mathsf{m}}{\mathsf{m}} = \frac{\hat{\mathsf{f}}(H)}{\hat{\mathsf{f}}(P)} \cdot \mathsf{m} = \frac{\hat{\mathsf{f}}(H)}{\hat{\mathsf{f}}(P)} \cdot \mathsf{m}(H, t) \qquad (1.5)$$

donde $\hat{\mathsf{f}}(P)$ es el valor de adaptación promedio de toda la población y $\hat{\mathsf{f}}(H)$ es el valor de adaptación promedio de las cadenas que son representadas por el esquema H en $P(t)$.

La Fórmula (1.5) explica que los esquemas con valores de adaptación por encima del promedio de la población recibirán un número mayor de muestras en la siguiente generación, mientras que los esquemas con valores de adaptación por debajo del promedio de la población recibirán un número cada vez menor de muestras [Gol89].

Ahora se considerarán los efectos del operador de cruzamiento. Existen elementos en la población que no van a ser cruzados, y estos elementos serán copiados en forma directa a la siguiente generación con una probabilidad $(1 - p_c)$. Además, existe una probabilidad de ruptura de un esquema debido al operador de cruzamiento. Siendo que un esquema sobrevive al cruzamiento simple sólo cuando el punto de cruce cae fuera del tamaño de definición, entonces, la probabilidad de sobrevivencia al cruzamiento simple es $1 - \frac{\delta(H)}{l-1}$.

Por simplicidad se supondrá que el cruzamiento siempre destruye. Pero, de echo, existe una probabilidad de que elementos que no pertenezcan a H generen por

cruzamiento elementos que pertenecen a H. Sea p_c la probabilidad de cruzamiento. Entonces el número de elementos de H en $t+1$ es, al menos:

$$\mathsf{m}(H,t+1) \geq \overbrace{(1-p_c)\cdot \mathsf{m}(H,t)\cdot\frac{\hat{\mathsf{f}}(H)}{\hat{\mathsf{f}}(P)}}^{\text{no elegidos para cruzamiento}} + \underbrace{(p_c)\cdot \mathsf{m}(H,t)\cdot\frac{\hat{\mathsf{f}}(H)}{\hat{\mathsf{f}}(P)}\cdot\overbrace{(1-\frac{\delta(H)}{l-1})}^{\text{no destruidos}}}_{\text{elegidos para cruzamiento}}$$

$$\mathsf{m}(H,t+1) \geq \mathsf{m}(H,t)\cdot\frac{\hat{\mathsf{f}}(H)}{\hat{\mathsf{f}}(P)}\cdot[1-p_c+p_c-p_c\cdot\frac{\delta(H)}{l-1}] = \mathsf{m}(H,t)\cdot\frac{\hat{\mathsf{f}}(H)}{\hat{\mathsf{f}}(P)}\cdot[1-p_c\cdot\frac{\delta(H)}{l-1}] \tag{1.6}$$

Entonces, considerando la selección y el cruzamiento a la vez, aquellos esquemas con valor de adaptación por encima del promedio y tamaño de definición pequeño reciben más muestras para la siguiente generación.

El último operador a considerar es la mutación de un bit. Para que un esquema sobreviva, la mutación no debe ocurrir en una posición con un valor fijo. Siendo p_m la tasa de mutación, la posibilidad de que un valor de un bit permanezca es $(1-p_m)$, entonces la probabilidad de sobrevivir para un esquema es $(1-p_m)^{o(H)}$, para pequeños valores de p_m ($p_m \ll 1$) la probabilidad de supervivencia del esquema puede ser aproximada por $1-o(H)\cdot p_m$. Entonces considerando todos los operadores, la expresión para $\mathsf{m}(H,t+1)$ es:

$$\mathsf{m}(H,t+1) \geq \mathsf{m}(H,t)\cdot\frac{\hat{\mathsf{f}}(H)}{\hat{\mathsf{f}}(P)}\cdot[1-p_c\cdot\frac{\delta(H)}{l-1}-o(H)\cdot p_m] \tag{1.7}$$

En conclusión, los esquemas cortos, de orden bajo, con adaptación por encima del promedio reciben muestras de manera exponencialmente creciente en cada generación. Esto se conoce como teorema de esquemas [Gol89, Hol75].

Muchos esquemas diferentes son muestreados cuando una población de cadenas es evaluada, y de echo, son muestreados más esquemas que el número de cadenas contenidas en la población. El efecto acumulativo de evaluar una población de puntos provee información estadística acerca de cualquier subconjunto particular de esquemas. Esto implica que muchas competencias entre esquemas son resueltas en forma paralela, Holland llamó a esto *paralelismo implícito* [Hol75].

Es importante recalcar que el término paralelismo implícito no se refiere a la potencialidad de implementar algoritmos genéticos paralelos. De echo, Holland primero uso el término paralelismo intrínseco en [Hol75], pero luego decidió cambiarlo al término paralelismo implícito para evitar confusiones con la terminología de la computación paralela. Desafortunadamente, el término paralelismo implícito en la comunidad de cómputo paralelo se refiere a paralelismo que es obtenido a partir de código escrito en lenguajes funcionales que no tienen constructores paralelos explícitos [Whi93].

1.5 Otros operadores y técnicas

A fin de ir mejorando el desempeño de los algoritmos evolutivos, además de los operadores básicos, se le han incorporado nuevas estrategias y técnicas como métodos para el manejo de restricciones, métodos de búsqueda local, métodos de *niching* [Sud20, LEDE16], métodos de subrogación [DRH20], de co-evolución [MLZ+18, AC17], de paralelización [von03, vBB15], etc. La calidad de las soluciones que pueden ser encontradas dependen de la manera en la que se combinan los diferentes operadores y métodos así como los parámetros específicos que son utilizados [WMS19].

Luego de varias generaciones, es posible que el operador de selección conduzca a todos los bits en algunas posiciones a un sólo valor (0 o 1 si se utiliza un código binario). Un gen se dice que ha convergido cuando al menos el 95% de la población comparte el mismo valor. Una población se dice que convergió cuando todos los genes han convergido [De 95].

Una vez que la población converge, la habilidad de un GA para continuar buscando mejores soluciones por cruzamiento desaparece. Cuando los individuos padres poseen cromosomas idénticos, el intercambio no produce ningún efecto y los individuos generados tienen cromosomas iguales al de sus padres. Sólo el operador de mutación permite explorar nuevas soluciones. Cuando esto ocurre, la búsqueda llevada a cabo por el algoritmo genético es sólo una búsqueda aleatoria. Idealmente, la convergencia debería ocurrir sólo cuando la solución es alcanzada. Si la convergencia ocurre sin que el GA alcance una solución satisfactoria, se dice que el GA ha convergido en forma prematura.

La preponderancia de individuos con un valor de adaptación demasiado elevado con respecto al promedio de la población lleva a que los métodos de selección estándar converjan en forma prematura. Este es un problema particularmente importante cuando se trabaja con poblaciones evolutivas pequeñas. Otro problema ocurre cuando todos los individuos tienen una probabilidad de selección similar y el GA posee una convergencia demasiado lenta. La convergencia hacia un pico u otro sin una ventaja importante entre un punto y otro es causada por la similaridad genética [Gol89].

Para prevenir problemas causados por la similaridad genética, se han desarrollado varios métodos. Los métodos de nicho (*niching*), por ejemplo, son técnicas que fomentan la formación y mantenimiento de subpoblaciones estables en GAs. El mantenimiento de la diversidad permite a los algoritmos genéticos buscar muchos picos en paralelo, evitando quedar atrapados en óptimos locales del espacio de búsqueda. Los métodos de *niching* pueden ser aplicados para la formación y el mantenimiento de subsoluciones interinas en la búsqueda de una única solución final. Sin embargo, estas técnicas han sido tradicionalmente utilizadas en el contexto de la formación y el mantenimiento de múltiples soluciones finales [Mah95, Mah97]. El método de repartija de *fitness* (*fitness sharing*) es probablemente el mejor y más conocido método de entre los métodos de *niching* [SK98].

1.5.1 Fitness Sharing

Los métodos de *fitness sharing* fueron introducidos en 1987 por Goldberg y Richardson para modificar el horizonte de búsqueda reduciendo el valor de adaptabilidad de los individuos que poseen cierta similaridad con respecto a otros individuos de una población [Mah95]. Los métodos de *fitness sharing* requieren que el valor de adaptación sea compartido entre individuos similares como un sólo recurso. Para esto a cada individuo $P[i]$ se le asigna un valor llamado su *cuenta de nicho*, denotado por $P[i]_{ncount}$, que se utiliza como indicador del nivel de similaridad del individuo. El *fitness compartido* de un individuo $P[i]$ es igual a su viejo valor de adaptación, dividido este valor de cuenta de nicho.

El valor de la cuenta de nicho para un individuo se calcula a partir de una función *sharing*. Ésta mide el nivel de similaridad entre dos elementos de la población. Tipicamente, la función retorna 1 si los elementos son idénticos, 0 si cruzan algún umbral de diferenciabilidad, y un nivel intermedio en un nivel intermedio de diferenciabilidad. El umbral de diferenciabilidad es especificado por una constante, σ_{share}, el cual indica la máxima distancia permitida para que un *sharing* diferente de cero ocurra. Típicamente la función de *sharing* es:

$$sh(d(P_i, P_j)) = \begin{cases} 1 - (\frac{d(P_i,P_j)}{\sigma_{share}})^{\alpha}, & \text{si } d(P_i, P_j) \leq \sigma_{share} \\ 0, & \text{de otro modo} \end{cases}$$

donde :

α es un parámetro constante que regula la forma de la función de *sharing*, y

d es una función de distancia, que calcula la distancia entre sus argumentos.

(1.8)

Cuando se implementa en el espacio objetivo, la función d usualmente corresponde a la distancia Euclidiana [DG89]. Mientras que, cuando se implementa en el espacio de genes, es usual utilizar el número de bits diferentes (o distancia de Hamming) entre las soluciones. Esta última distancia es útil en los problemas definidos en el espacio de los genes [DG89].

El valor de la *cuenta del nicho* para $P[i]$ es igual a la sumatoria de los valores de la función de *sharing* entre $P[i]$ y cada individuo de la población $P[j]$

$$P[i]_{ncount} = \sum_{j=1}^{N} sh(d(P_i, P_j)) \tag{1.9}$$

Finalmente, el valor de adaptación compartido del elemento $P[i]$ es:

$$P[i]_{fitness} = \frac{P[i]_{fitness}}{P[i]_{ncount}} \tag{1.10}$$

1.5.2 Restricción de apareamiento

Cuando el procedimiento de *niching* se implementa en problemas con objetivo único, Deb y Goldberg [DG89] notaron que la recombinación entre cromosomas de nichos diferentes usualmente producen individuos pobremente adaptados. Esto lleva a la degradación del desempeño del GA. Con el fin de evitar este problema, en [DG89] se propone restringir el apareamiento (*matting restriction*) sólo entre individuos que están a una cierta distancia en el espacio genotípico.

1.5.3 Elitismo

La aplicación iterativa de los operadores evolutivos puede hacer que una buena solución se pierda. Esto puede ocurrir porque, aunque sea mejor que todas las otras, la cantidad de muestras existentes es insignificante y no es seleccionado para el cruzamiento. Además, la aplicación del operador de mutación puede destruirlo. En vista a esto, en [De 95] se sugiere una política en la cual siempre se incluye el mejor individuo de $P(t)$ en $P(t+1)$ a fin de preservarlo de la posible acción negativa de los operadores genéticos. A esta estrategia se le conoce como elitismo.

Entonces, básicamente en un algoritmo elitista las mejores soluciones obtenidas son almacenadas en un archivo externo a fin de asegurar que no se pierdan. Además, estas soluciones pueden participar en el proceso de búsqueda a fin de acelerar el proceso de convergencia.

1.6 Aplicación de algoritmos evolutivos en problemas multi-objetivo

Como dijimos, los paradigmas principales de los algoritmos evolutivos se establecieron en la década de los 60s, sin embargo, debido fundamentalmente a la falta de recursos computacionales suficientes para su implementación en problemas reales, sufrieron un periodo de rechazo y poco interés. Posteriormente, con el aumento de poder computacional en la década del 80, las técnicas evolutivas recibieron mayor atención como métodos para la resolución de problemas con un único objetivo. Sin embargo, recién a partir de la década del noventa estos métodos se establecieron como alternativas prácticas cuando fueron utilizados con éxito para proponer soluciones en un gran número de problemas complejos de optimización y búsqueda, requiriendo poca información sobre los mismos [BFM97, Gol94]. A partir de entonces, el éxito de la técnica llamo la atención de varios investigadores para diversas aplicaciones en problemas de naturaleza multi-objetivo.

En los problemas de optimización multi-objetivo (*Multiobjective Optimization Problems* - MOP) cada solución representa compromisos entre los diferentes objetivos considerados. De esta forma, tipicamente, se obtiene un conjunto de soluciones

en las que no existe ningún objetivo o criterio en el que se pueda conseguir una mejora sin empeorar el valor de al menos una de las otras funciones objetivo. El conjunto de estas soluciones, recibe el nombre de conjunto Pareto Óptimo (*Pareto Optimal Set*).

Puesto que ninguna solución que compone el conjunto Pareto Óptimo puede considerarse mejor que otra al considerar todos los objetivos, todas ellas pueden considerarse como igualmente buenas [CLV07, Deb01]. La selección de *una* solución para resolver un MOP específico requiere, por lo general, la intervención de un tomador de decisiones capaz de establecer algún tipo de preferencia entre estas. De echo, para establecer un orden entre las soluciones Pareto óptimas es necesario especificar algún tipo de preferencia entre los objetivos o en función a las características de las soluciones propuestas. Estas preferencias, en muchos casos, no pueden ser determinadas a priori sino luego del análisis del las soluciones alternativas y los valores de éstas en el espacio objetivo, lo que se conoce como Frente Pareto.

1.6.1 Primera generación de MOEA: dominancia Pareto

Las tres metas principales cuando se resuelve un MOP son:

- *convergencia*: encontrar un conjunto de soluciones cuyos valores en el espacio objetivo se encuentren tan próximos al Frente Pareto como sea posible;
- *diversidad*: encontrar un conjunto de soluciones que represente adecuadamente la variedad de soluciones posibles; y
- *cobertura*: encontrar un conjunto de soluciones que, además de ser diverso, se encuentre bien distribuido en el Frente Pareto.

A mediados de la década de los 80s se propuso el primer algoritmo evolutivo que intentaba resolver un problema de objetivos múltiples [Sch85], el VEGA (*Vector Evaluated Genetic Algorithm*), evidenciando el potencial de los EA para aproximar soluciones a problemas con muchos objetivos. Sin embargo, no fue sino hasta inicios de los 90s que investigadores propusieron los primeros Algoritmos Evolutivos Multi-objetivo (*Multi-objective Evolutionary Algorithms*- MOEA) basados en dominancia Pareto [CLV07, Deb01, CSHM19, ZX17]. Sumado a la habilidad de encontrar soluciones con un mínimo de información, propia de los algoritmos genéticos, el interés en la utilización de MOEA para la resolución de MOP radica en la capacidad que estos tienen de conseguir cumplir con las diferentes metas antes señaladas al aproximar el conjunto Pareto en una única corrida, además, de la inexistencia de técnicas exactas y directas para obtener tales aproximaciones para un gran número de MOP. El concepto de dominancia Pareto permite comparar una solución respecto a otra en un espacio multi-objetivo. Si una solución A no es peor en ningún objetivo y es estrictamente mejor en al menos un objetivo que la solución B, se dice que A domina a B. Una solución no está dominada con respecto a un conjunto dado de soluciones si en dicho conjunto no hay ninguna solución que la domine. Consecuentemente, el conjunto de Pareto está compuesto por todas las soluciones no

dominadas que pertenecen a todo el espacio factible [CLV07]. Cuando se resuelve un problema multi-objetivo se desea, por lo general, encontrar un conjunto aproximado de soluciones no dominadas lo más cerca posible del conjunto de Pareto real y que posea una amplia distribución de soluciones.

La aparición de los primeros algoritmos basados en el concepto de dominancia Pareto [FF93a, FF93b, HN93, HNG94, SD93, SD94, SD95] marcó un hito y estos algoritmos empezaron su consolidación como solucionadores robustos de problemas de optimización multi-objetivo aplicándose en varias áreas.

1.6.2 Segunda generación de MOEA: utilización de elitismo

Aunque los MOEAs de la primera generación basados en Pareto, tuvieron cierto éxito para encontrar soluciones en una amplia gama de problemas, durante el proceso iterativo de aplicación de operadores, algunas buenas soluciones que eran encontradas podían perderse. A pesar del éxito inicial obtenido por los MOEAs de la primera generación basados en Pareto, las buenas soluciones encontradas podían perderse cuando se aplicaban los operadores genéticos en las sucesivas generaciones. Como una primera mejora significativa a estos algoritmos, el concepto de *elitismo* que venía siendo aplicado en los algoritmos mono-objetivo, se extendió a los algoritmos multi-objetivo. La utilización de elitismo en los MOEA permitió mejorarlos considerablemente marcando un nuevo hito en la evolución de la técnica. La utilización de conceptos de Pareto conjuntamente con alguna forma de elitismo para la búsqueda de soluciones es la característica común de varios algoritmos de la segunda generación como el *Strenght Pareto Evolutionary Algorithm* (SPEA) de Zitzler y Thiele [ZT99a] y el NSGA-II de Deb, Agrawal, Pratab y Meyarivan [DPAM02], entre otros. Esta característica se extiende hasta los métodos más modernos [DJ14b, ZL08].

Los MOEA basados en dominancia Pareto han mostrado ser especialmente útiles en problemas con funciones complejas, difíciles de tratar con métodos exactos y donde es imposible establecer a priori preferencias entre los objetivos considerados cuando se optimizan simultáneamente dos o tres objetivos. Sin embargo, cuando se utilizan para resolver problemas de muchos objetivos (*Many-objective Optimization Problems* - MaOP), es decir, problemas que tienen 4 o más objetivos, surgen dificultades de convergencia [IAN15, IDN17, LLTY15, vLBB14, MOBE18].

La principal dificultad con la que tienen que lidiar los MOEA basados en dominancia Pareto cuando son aplicados en la resolución de problemas con muchos objetivos es lo que se conoce como resistencia a la dominancia. Esto significa que, dada una población, la proporción del número de soluciones no dominadas en esa población con respecto a su tamaño es cercana a uno [FA02]. Entonces, en el caso de muchos objetivos, la dominancia Pareto no puede ser utilizada para discriminar suficientemente bien entre las soluciones ya que bajo ese criterio prácticamente todas son igualmente buenas [vLBB14, vLBB19].

Por otra parte, para obtener una buena aproximación al frente de Pareto completo se requiere un número de soluciones que aumenta exponencialmente con el

número de funciones objetivo. Esto significa que se necesitan poblaciones cada vez más grandes para obtener una aproximación adecuada al conjunto Pareto [Deb01]. Cuando los algoritmos trabajan con poblaciones más grandes, se requieren tiempos de ejecución más largos y podría ser necesario adaptar las estructuras y procedimientos para ser capaces de lidiar efectivamente con el aumento de escala. La visualización de las soluciones obtenidas, y la evaluación de los algoritmos resultan cuestiones adicionales a encarar a fin de realizar una adecuada selección de las soluciones a implementar así como una correcta evaluación de los algoritmos [LLTY15, vLBB14].

Adicionalmente, otra cuestión relevante que debe ser abordada para la resolución de problemas con muchos objetivos utilizando MOEA es que estos, por lo general, incorporan en el proceso de optimización algún método para estimar la diversidad de la población en el espacio de objetivos para guiar la búsqueda. Estas técnicas en general permiten evitar la convergencia prematura hacia ciertas regiones al modificar la probabilidad de que un individuo sea seleccionado o para su aceptación como parte de un archivo en línea a fin de lograr una buena distribución de soluciones. Cuando el número de objetivos se incrementa la calidad de esta estimación se degrada. Peor aún, este valor es un criterio de búsqueda secundario en los MOEAs que los utilizan. Sin embargo, cuando la proporción de soluciones no-dominadas en la población evolutiva aumenta, la selección tiende a basarse únicamente en la diversidad lo que, como señalan [FA02], en la búsqueda *many-objective* puede producir que la población evolucione hacia zonas lejanas del frente Pareto óptimo.

1.6.3 Tercera generación de MOEA: múltiples enfoques y optimización con muchos objetivos

A fin de continuar el proceso de mejora de los MOEA aplicados a problemas multi-objetivo surgieron una diversidad de alternativas para la resolución de estos problemas utilizando computación evolutiva. La tercera generación de MOEA se puede caracterizar por esta variedad: co-evolución cooperativa [MLZ+18], algoritmos basados en índices [FCC20], descomposición [OEBY20, WSL+20], entre otros. Una de las principales motivaciones en está búsqueda de alternativas ha sido la necesidad de la utilización de MOEAs en problemas de mayor número de objetivos. Justamente, las dificultades de los algoritmos evolutivos multi-objetivo tradicionales han llevado a varios investigadores a desarrollar nuevos métodos para la resolución de problemas donde el número de objetivos a optimizar de manera simultánea es elevado [LLTY15, vLBB14, vLBB19, BES17, IS19].

A fin de estudiar los métodos desarrollados para la optimización con muchos objetivos, [vLBB14] los clasifican en dos grupos principales de acuerdo a la idea central que los caracterízan: (i) métodos que utilizan relaciones de preferencia alternativas y (ii) métodos que transforman el problema *many-objective* original en uno relacionado.

En los *métodos basados en relaciones de preferencias* se utiliza algún tipo de información adicional en forma de preferencias a fin de proporcionar una clasificación más precisa de las soluciones. Esta información puede resultar de comparaciones entre los diferentes objetivos o el conteo de objetivos en las cuales una posible solución es mejor que otra. En cualquier caso, la relación a utilizar dependerá del objetivo de la optimización, esto es, si lo que se quiere obtener es un muestreo amplio del conjunto o bien si lo que se busca es una solución específica con ciertas características; esto a su vez depende del grado de conocimiento del problema específico [vLBB14, LLTY15, MSW+20].

Entre las técnicas que transforman el problema *many-objective* original en uno relacionado se encuentran: (i) basadas en agregación, (ii) basadas en técnicas de reducción de dimensionalidad, (iii) basadas en indicadores, y (iv) basadas en descomposición. Dado que existen hibridaciones con estos enfoques, en algunos casos no está claro qué clasificación se ajusta mejor a un dado algoritmo.

Los enfoques basados en agregación consiguen transformar un problema de muchos objetivos en un problema de optimización de múltiples objetivos (2 o 3 objetivos) o incluso de un solo objetivo utilizando técnicas de escalarización comunes como: suma ponderada, Tchebycheff ponderado y escala de distancia angular [LLTY15, vLBB14].

Para asignar el valor de adaptabilidad a cada solución, los métodos basados en indicadores utilizan alguna métrica de evaluación escalar o indicador para evaluar los elementos de la población. Algunos indicadores utilizados para esto son el hipervolumen [DSP+20, Sin19, SI20], métricas de estimación a la distancia al frente Pareto y basados en la métrica R2 [LLKH19]. La principal desventaja de estos métodos es el alto costo computacional para calcular algunas de estas métricas, así como la necesidad de conocer el verdadero Frente Pareto o al menos una muy buena aproximación a él, así como la dependencia a la forma de este para un comportamiento eficiente [BLIS17, TZCJ16].

En el caso de las técnicas basadas en la reducción de la dimensionalidad, estas utilizan alguna técnica, de las muchas existentes, para transformar un conjunto de funciones objetivo en un número reducido de ellas. Esto es posible solo bajo el supuesto de que el *verdadero* problema es de menor dimensión y por tanto se pueden establecer relaciones entre los objetivos para reducir el número de estos [BZ06b, PF03]. Estos métodos tienen múltiples ventajas respecto a otras técnicas, en particular, permiten la utilización de métodos ya conocidos y utilizados en optimización evolutiva y, por tanto, pueden producir una reducción en el costo computacional [LRLL20]. Sin embargo, las técnicas de reducción de dimensionalidad son por lo general muy dependientes de las características del problema que se está resolviendo, por lo que, si bien estos métodos pueden ser útiles, el potencial para la reducción de objetivos en problemas con muchos objetivos es limitado cuando un gran número (o todos) los criterios están en conflicto entre sí.

En el caso de los algoritmos basados en descomposición, estos plantean dividir el problema original en varios subproblemas escalares o multi-objetivo para optimizarlos simultáneamente [Hug03, ZL07, DJ14b, CJOS16, HDD+19, WSL+20, TSSG16]. Existen dos métodos básicos para la descomposición de problemas: (i) el uso de fun-

ciones de agregación y (ii) el uso de puntos de referencia. Estos algoritmos se caracterizan por su relativamente baja complejidad computacional y generalmente utilizan una relación explícita de vecindad. La idea de la descomposición para resolver MOP se hizo popular con la introducción en 2007 del *Multi-objective Evolutionary Algorithm based on Decomposition* (MOEA/D) por Zhang y Li [ZL07].

Otros algoritmos basados en descomposición de gran importancia en el contexto de muchos objetivos son el *Non-dominated Sorting Genetic Algorithm-III*- NSGA-II) [DJ14b] y el *Reference Vector Guided Evolutionary Algorithm* - (RVEA) [CJOS16]. Estos algoritmos serán tratados en el Capítulo 5.

En el próximo capítulo presentaremos un conjunto de conceptos básicos que hacen a la optimización multi-objetivo.

Capítulo 2
Problemas de Optimización Multi-objetivo

Este capítulo presenta formalmente los principios de optimización con objetivos múltiples en el contexto de los algoritmos evolutivos. Así mismo, se describe el modelo general del proceso de solución utilizado con problemas de optimización multi-objetivo.

2.1 Conceptos de optimización multi-objetivo

Los primeros algoritmos evolutivos fueron desarrollados para resolver problemas con un único objetivo (mono-objetivo). Cuando se tiene un único objetivo, por lo general es posible hablar de *un* óptimo global ya que el espacio de búsqueda se encuentra totalmente ordenado por medio de la función objetivo. El óptimo global se define como sigue [von03]:

Definición 2.1 *Óptimo global (monobjetivo):* dada una función $f : \mathbb{R}^n \rightarrow \mathbb{R}$, para $\mathbf{x} \in \mathbb{R}^n$ el valor $f^\star = f(\mathbf{x}^\star)$ es llamado un óptimo global (máximo o mínimo) sí y sólo sí:

$$\forall \mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}^\star) \textit{ es mejor que } f(\mathbf{x}) \tag{2.1}$$

Entonces, $\mathbf{x}^\star$ es la solución global óptima, f es la función objetivo. El problema de determinar la solución global óptima es llamado *problema de optimización global*.□

En los problemas de objetivo único es posible aprovechar la existencia de un orden total en el dominio para asignar el valor de adaptabilidad a cada elemento del conjunto de soluciones relacionado a su posición en dicho orden. De esta forma, el proceso de optimización consiste en encontrar el mínimo (o el máximo) de una función. Cuando se tienen múltiples funciones objetivos pero es posible determinar alguna preferencia entre los objetivos a priori, es posible la utilización de un vector de pesos para, por ejemplo, realizar una suma ponderada transformando un problema de optimización con múltiples funciones en una sola función objetivo que combine el valor de las mismas de acuerdo al peso asignado a cada objetivo para así transformar el problema con muchos objetivos en un problema de un único objetivo.

En los problemas que nos interesan en este trabajo, sin embargo, no se tiene determinado a priori un orden de preferencia entre objetivos, y si existe, se desea obviarlo al momento de realizar la búsqueda de soluciones hasta conseguir un mejor conocimiento del problema para luego definir mejor las eventuales preferencias y ajustar la búsqueda.

Un gran número de problemas de toma de decisión en ingeniería y ciencia necesitan considerar la existencia de múltiples criterios o objetivos. En estos problemas podemos notar los siguientes aspectos [von03]:

- los distintos objetivos considerados pueden ser conflictivos, tener distintas unidades de medidas, y ser inconmensurables,
- al existir objetivos contradictorios no existe una solución única que sea mejor que las otras con respecto a todos los objetivos, generalmente se tiene un conjunto de alternativas de solución, las cuales representan los mejores compromisos entre los distintos objetivos,
- de las alternativas existentes sólo interesa el conjunto de soluciones que cumplen con ciertas restricciones (soluciones factibles) y finalmente,
- existiendo un conjunto de alternativas posibles, es necesario que un tomador de decisiones proporcione información de preferencia a fin de seleccionar la solución a ser implementada.

En este trabajo, sin pérdida de generalidad, consideramos problemas de optimización multi-objetivo (*Multi-objective Optimization Problem (MOP)*) en un contexto de minimización; por lo tanto, un MOP se define de la siguiente manera [Deb01]:

Definición 2.2 *Problema de optimización multi-objetivo:* sea $\mathcal{F}$ un conjunto con m funciones objetivo $\{f_1, \ldots, f_m\}$, $f_i : \mathbb{R}^n \Rightarrow \mathbb{R}$, un MOP se define como:

$$\begin{aligned} \text{Minimizar} \quad & \mathbf{y} = \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})) && (2.2)\\ & \mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X} \subseteq \mathbb{R}^n \\ & \mathbf{y} = (y_1, \ldots, y_m) \in \mathcal{Y} \subseteq \mathbb{R}^m \end{aligned}$$

sujeto a

$$\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_c(\mathbf{x})) \leq \mathbf{0} \quad (2.3)$$

$$x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad \forall i \in \{1, \ldots, n\} \quad (2.4)$$

$\mathbf{x}$ es un vector de n variables de decisión, mientras que $\mathbf{y}$ representa un vector objetivo m-dimensional. Las restricciones (2.4) representan los $2n$ límites de las variables que ayudan a definir el espacio de las variables de decisión o espacio de decisión $\mathcal{X}$. Las funciones objetivo constituyen el espacio objetivo, denominado $\mathcal{Y}$.
El vector de restricciones $\mathbf{g}$ se compone de c funciones de restricción que delimitan la forma de la región de factibilidad. Las soluciones que no satisfacen las restricciones son llamadas soluciones no factibles, mientras que las soluciones que cumplen con todas las restricciones en (2.4) y (2.3) son soluciones factibles. El conjunto de todas las soluciones factibles $\mathcal{X}_f$ es conocido como la región de factibilidad.

Para cada solución $\mathbf{x} \in \mathcal{X}_f$ existe un punto $\mathbf{y}$ en el espacio objetivo. Entonces, $\mathcal{X}_f$ define el espacio objetivo factible $\mathcal{Y}_f$:

$$\mathcal{Y}_f = \mathbf{F}(\mathcal{X}_f) = \bigcup_{\mathbf{x} \in \mathcal{X}_f} \{\mathbf{F}(\mathbf{x})\} \tag{2.5}$$

□

Como se ha señalado previamente, la mayoría de los MOEAs utilizan dominancia Pareto a fin de comparar sus soluciones y proveer un valor de adaptación que guíe la búsqueda. En el caso de problemas de optimización con muchos objetivos, algunos MOEAs utilizan este concepto sobre un subconjunto de objetivos del MOP original que se desea resolver [BZ06b, dPKS07, LJCCC08]. Para considerar estos casos, en un contexto de minimización, se utiliza la siguiente definición de dominancia Pareto débil sobre un subconjunto de funciones objetivo [BZ06b]:

Definición 2.3 *Relación de dominancia Pareto débil:* la dominancia Pareto sobre un conjunto de objetivos $\mathcal{F}' \subseteq \mathcal{F}$ se define como:

$$\preceq_{\mathcal{F}'} = \{(\mathbf{x}, \mathbf{x}') | \mathbf{x}, \mathbf{x}' \in \mathcal{X}_f \wedge \forall f_i \in \mathcal{F}', f_i(\mathbf{x}) \leq f_i(\mathbf{x}')\} \tag{2.6}$$

Si $(\mathbf{x}, \mathbf{x}') \in \preceq_{\mathcal{F}'}$, $\mathbf{x}$ domina débilmente a $\mathbf{x}'$ sobre $\mathcal{F}'$, también denotado como $\mathbf{x} \preceq_{\mathcal{F}'} \mathbf{x}'$. Si $(\mathbf{x}, \mathbf{x}') \notin \preceq_{\mathcal{F}'}$, se dice que la solución $\mathbf{x}'$ es débilmente no-dominada por $\mathbf{x}$ con respecto al conjunto de objetivos $\mathcal{F}'$, denotado por $\mathbf{x} \npreceq_{\mathcal{F}'} \mathbf{x}'$. Para simplificar la notación, si no existe posibilidad de ambigüedad, en el caso que $\mathcal{F}' = \mathcal{F}$ y $\mathbf{x} \preceq_{\mathcal{F}'} \mathbf{x}'$, se dice simplemente que $\mathbf{x}$ domina débilmente a $\mathbf{x}'$, denotado por $\mathbf{x} \preceq \mathbf{x}'$. Igualmente la relación de no dominancia débil sobre el conjunto de todos los objetivos se denota por $\npreceq$.

Definición 2.4 *Relación de dominancia Pareto:* la relación de dominancia Pareto con respecto a un conjunto de objetivos $\mathcal{F}' \subseteq \mathcal{F}$ se define como:

$$\prec_{\mathcal{F}'} = \{(\mathbf{x}, \mathbf{x}') | \mathbf{x}, \mathbf{x}' \in \mathcal{X}_f \wedge \forall f_i \in \mathcal{F}', f_i(\mathbf{x}) \leq f_i(\mathbf{x}') \wedge \exists f_j \in \mathcal{F}', f_j(\mathbf{x}) < f_j(\mathbf{x}')\} \tag{2.7}$$

Si $(\mathbf{x}, \mathbf{x}') \in \prec_{\mathcal{F}'}$, se dice que la solución $\mathbf{x}$ domina a la solución $\mathbf{x}'$ con respecto a $\mathcal{F}'$, denotado por $\mathbf{x} \prec_{\mathcal{F}'} \mathbf{x}'$, mientras que, si $(\mathbf{x}, \mathbf{x}') \notin \prec_{\mathcal{F}'}$ se dice que la solución $\mathbf{x}'$ es no dominada por $\mathbf{x}$ sobre el conjunto de objetivos $\mathcal{F}'$, denotado por $\mathbf{x} \nprec_{\mathcal{F}'} \mathbf{x}'$. En caso que $\mathcal{F}' = \mathcal{F}$ y $\mathbf{x} \prec_{\mathcal{F}'} \mathbf{x}'$, se dice simplemente que $\mathbf{x}$ domina a $\mathbf{x}'$, denotado por $\mathbf{x} \prec \mathbf{x}'$. Igualmente la relación de no dominancia fuerte sobre el conjunto de todos los objetivos se denota por $\nprec$.

La Figura 2.1 muestra gráficamente la noción de dominancia Pareto donde puede visualizarse la comparación de tres soluciones en el espacio objetivo: A, B y C. Se puede ver que A no domina a C y, C no domina a la solución A, esto se debe a que ninguna de las dos es mejor que otra cuando se consideran en conjunto los dos objetivos. las soluciones dominadas, no-dominadas y no comparables. También, que A domina a B ya que no es peor que B en ningún objetivo y existe, al menos uno donde A es mejor a B.

Definición 2.5 *Optimalidad de Pareto:* una solución $\mathbf{x} \in \mathcal{X}_f$ se dice que es no dominada considerando los objetivos $\mathcal{F}' \subseteq \mathcal{F}$ con respecto a un conjunto $\Omega \subseteq \mathcal{X}_f$,

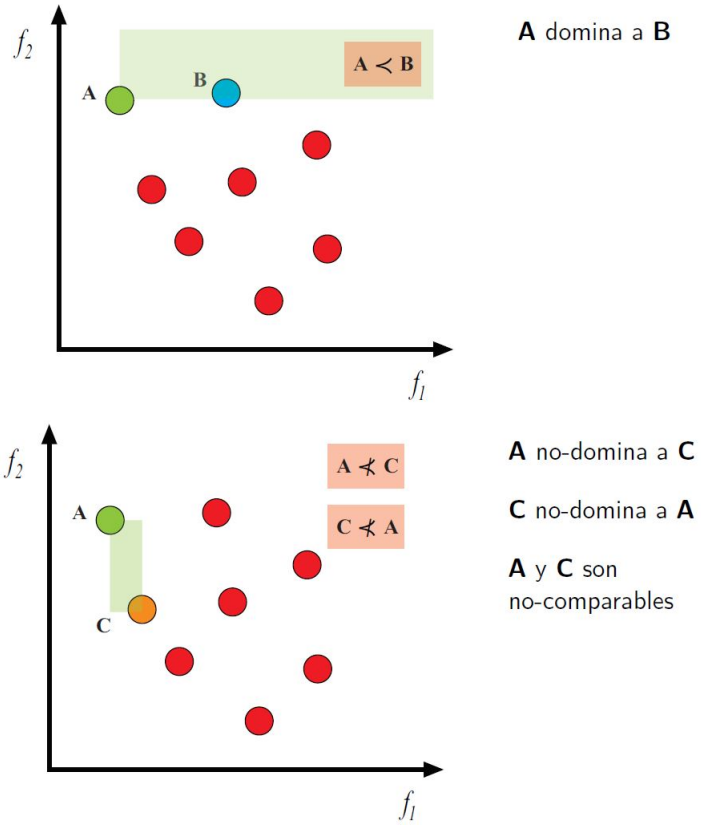

Fig. 2.1 Dominancia Pareto: comparación de tres vectores A, B y C .

si y solo si $\nexists \mathbf{x}' \in \Omega$ para la cual $\mathbf{x}' \prec_{\mathcal{F}'} \mathbf{x}$. Si $\mathbf{x}$ es no dominada con respecto a $\mathcal{X}_f$ considerando $\mathcal{F}'$, se dice que es una solución Pareto óptima para el subespacio de objetivos dado, mientras, si $\mathcal{F}' = \mathcal{F}$ se dice que esta es una solución Pareto óptima del problema o simplemente una solución Pareto óptima.

Las soluciones Pareto óptimas forman el conjunto Pareto, definido como:

Definición 2.6 *Conjunto Pareto:* para un MOP dado con un conjunto de objetivos $\mathcal{F}$, el conjunto Pareto considerando los objetivos $\mathcal{F}' \subseteq \mathcal{F}$ se define como:

$$\mathcal{P}^{\star}_{\mathcal{F}'} = \{\mathbf{x} \in \mathcal{X}_f \,|\, \nexists \mathbf{x}' \in \mathcal{X}_f \text{ tal que } \mathbf{x}' \prec_{\mathcal{F}'} \mathbf{x}\} \tag{2.8}$$

Los correspondientes vectores de $\mathcal{P}^{\star}_{\mathcal{F}'}$ en el espacio objetivo definido por $\mathcal{F}'$ forman el frente Pareto, denotado por $\mathcal{PF}^{\star}_{\mathcal{F}'}$. Cuando $\mathcal{F}' = \mathcal{F}$, los conjuntos $\mathcal{P}^{\star}_{\mathcal{F}}$ y $\mathcal{PF}^{\star}_{\mathcal{F}}$ se llaman conjunto Pareto y frente Pareto, denotados por $\mathcal{P}^{\star}$ y $\mathcal{PF}^{\star}$, respectivamente.

La Figura 2.2 muestra el Frente Pareto de un conjunto de soluciones. Como puede verse, el conjunto de elementos que lo componen no puede ser considerado mejor a ningún otro en el conjunto cuando se consideran todos los objetivos de manera simultánea.

2.2 Proceso de solución de un MOP

Un problema de optimización multiobjetivo se considera matemáticamente resuelto cuando el conjunto Pareto Óptimo es encontrado. En problemas reales, usualmente se requiere una única solución. Puesto que desde un punto de vista estrictamente matemático todas las soluciones en el conjunto Pareto son igualmente buenas, para obtener *la* solución a ser implementada en un caso concreto, se necesita información adicional de preferencia. Un tomador de decisiones (*decision maker* - Decision Maker (DM)) humano es quien se encarga de proporcionar la información de preferencia necesaria para seleccionar una única solución del conjunto Pareto. Así, el proceso de solución de un MOP puede dividirse en dos procesos conceptualmente distintos [Hor97, Mie01]:

- *Proceso de búsqueda u optimización*, por el cual se explora el conjunto de soluciones factibles en busca de soluciones Pareto Óptimas.
- *Proceso de toma de decisiones*, por el cual se selecciona una solución de compromiso adecuada, a partir del Conjunto Pareto Óptimo hallado por el proceso anterior.

La importancia de la toma de decisiones como parte del proceso de solución es una buena razón para clasificar los distintos métodos para solucionar problemas de optimización multiobjetivo de acuerdo con la manera en que se combinan la

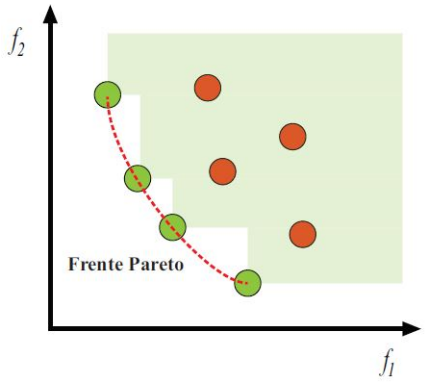

Fig. 2.2 Frente Pareto de un conjunto de soluciones.

búsqueda del conjunto de soluciones y la toma de decisiones [Mie01]. Así, los métodos se clasifican en:

- **Métodos a priori:** la toma de decisiones se realiza antes de la búsqueda de soluciones. Usualmente los distintos objetivos se combinan en uno solo, el cual implícitamente incluye información de preferencia proporcionada por el tomador de decisiones. Esto hace que efectivamente el MOP se convierta en un problema de optimización monobjetivo, antes de la optimización.
- **Métodos a posteriori:** la toma de decisiones se realiza de forma posterior a la búsqueda. La optimización se realiza sin ninguna información de preferencia. El resultado del proceso de búsqueda es un conjunto de soluciones candidatas, idealmente Pareto-Optimas, a partir del cual la decisión final será realizada por el tomador de decisiones, conociendo las alternativas disponibles.
- **Métodos interactivos o progresivos:** la toma de decisiones se realiza durante la búsqueda de soluciones de manera interactiva. Luego de cada paso de optimización, un conjunto de soluciones de compromiso es presentado al tomador de decisiones, quien proporcionará información de preferencia que guiará el proceso de búsqueda.

Con los métodos a priori, donde se combinan múltiples objetivos en un único criterio de optimización, cuando son aplicables, se tiene la ventaja que las estrategias clásicas de optimización monobjetivo pueden ser utilizadas sin modificaciones. La desventaja es que requiere un conocimiento profundo del dominio del problema que permita realizar la escalarización de forma correcta. Este conocimiento del dominio del problema requerido por los métodos a priori usualmente no se encuentra disponible. Es más, en varios problemas de diseño en ingeniería, lo que se desea específicamente es ganar mayor conocimiento sobre el problema y sobre las soluciones alternativas. Realizar la toma de decisiones luego del proceso de búsqueda (métodos a posteriori) resuelve el problema, pero excluye la articulación de la preferencia del tomador de decisiones, lo que podría a su vez reducir la complejidad del espacio de búsqueda. Los métodos interactivos superan las desventajas de los métodos a priori y posteriori permitiendo al tomador de decisiones expresar sus preferencias a medida que avanza en el conocimiento del problema considerado.

2.3 Influencia de la dimensión sobre la dificultad del problema para los MOEAs

En 1978, [BKST78] derivaron una recurrencia para computar el número promedio de vectores no-dominados con m dimensiones en un conjunto dado de tamaño N, para un valor fijo de m, bajo la suposición de que las magnitudes de los elementos están distribuidas de manera independiente, y que, para cada componente, las magnitudes elegidas para cada vector son distintas. Luego, resolviendo la recurrencia, [BKST78] demostraron que el número de vectores no-dominados está acotado por $O((logN)^{m-1})$.

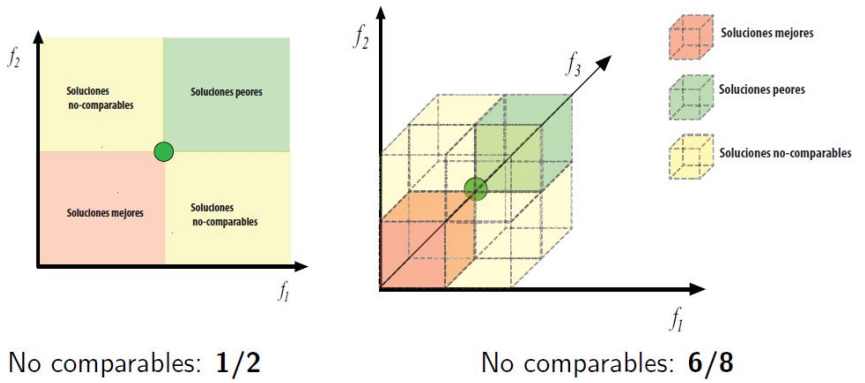

Fig. 2.3 Proporción de soluciones no comparables [FA02].

Apenas aparecieron los primeros MOEAs basados en Pareto, [FF95] tomaron nota de que, debido al incremento en el número de soluciones no-dominadas, estos métodos podrían no obtener soluciones satisfactorias en problemas donde se consideran muchos objetivos conflictivos simultáneamente; sin embargo, los autores no realizaron un análisis más acabado sobre la cuestión. Luego, Deb presentó en [Deb01, págs. 400-405] un estudio empírico donde se cuenta el número de soluciones no-dominadas que existen en un millón de poblaciones generadas aleatoriamente para diferentes combinaciones de tamaño de población y número de objetivos. El resultado experimental muestra como, un incremento del número de funciones objetivo hace que un gran número de soluciones pertenezcan al conjunto de soluciones no-dominadas. Entonces, en [Deb01] se concluye que los MOEAs basados en Pareto no pueden ser capaces de proveer la presión de selección necesaria para conducir la búsqueda evolutiva hacia las mejores soluciones, e incluso que el elitismo puede volverse difícil de implementar.

En la Figura 2.3 puede verse por un lado la forma en que se clasificaría una solución de valor $(0.5; 0.5)$ en un espacio de dos objetivos donde cada valor puede estar en el rango de 0 a 1. Como se señala, la proporción de elementos no comparables es 1/2, mientras que agregando una dimensión más la proporción de elementos no comparables es de 6/8. En el trabajo de Farina y Amato [FA02] se provee la siguiente expresión general para el crecimiento esperado de la proporción e del espacio del dominio con los puntos que la relación de dominancia Pareto clasifica como no comparable con respecto a una solución dada:

$$e = \frac{2^m - 2}{2^m} \tag{2.9}$$

Entonces, como señala la Ecuación (2.9), para una solución dada, conforme el número de objetivos m crece, la proporción de soluciones no comparables con respecto a dicha solución tiende a uno. De esta forma, [FA02] explican que la dominancia Pareto puede ser inadecuada para discriminar entre las soluciones en problemas

many-objective. De esta forma, los algoritmos como el NSGA-II [DPAM02] y otros presentados ene el Capítulo 3 que utilizan el concepto de dominancia, no podrán guiar adecuadamente la búsqueda en problemas con muchos objetivos ya que la mayor parte de las soluciones en la población evolutiva serán no comparables y por tanto el mecanismo de selección será guiado por el cálculo de la distancia de *crowding*, incluso al comienzo del proceso evolutivo.

Teytaud [Tey07] comparó el límite inferior para el tiempo de computación de una clase dada de MOEA contra el límite superior de un procedimiento puramente aleatorio. Los MOEAs analizados fueron aquellos que se basan solo en dominancia Pareto binaria para comparar soluciones. El trabajo considera una familia de problemas multi-objetivo con conjuntos Pareto suaves (*smooth Pareto sets*), que se ajustan a la desigualdad de Lipschitz [Tey07] en el Frente Pareto y para las cuales una reducción en el número de sus objetivos no es posible. Para estos problemas, el tiempo de computación se define como la cuenta de evaluaciones de la función objetivo y comparaciones por dominancia necesarias para obtener, con una probabilidad dada, un conjunto Pareto aproximado con una precisión dada para la distancia de Haussdorf [Tey07]. El análisis presentado en [Tey07] demostró que, para los problemas considerados, cuando el número de objetivos se incrementa, el límite superior del tiempo computacional de una búsqueda aleatoria pura es muy cercana al límite inferior del tipo de MOEA estudiado; por tanto, estos MOEAs son (en el mejor caso) resultan equivalentes a la búsqueda aleatoria cuando el número de objetivos es grande.

Contrariamente a la idea de que incrementar el número de objetivos hace a un problema más difícil, [BFH+07] desarrollaron un análisis teórico del tiempo de ejecución de un optimizador multi-objetivo mostrando que, con un objetivo adicional, una función meseta (*plateau function*) dada, puede volverse tanto más fácil como más difícil. Consecuentemente, aumentar el número de objetivos puede tanto acelerar como frenar a un MOEA dependiendo del problema y los objetivos elegidos; de esta forma, el número de objetivos no es un indicador suficiente de la dificultad de un problema.

[SLCC11] analizaron la influencia, en términos de convergencia, del número de objetivos sobre la dificultad de un problema de optimización multi-objetivo continuo. En [SLCC11], los autores propusieron un conjunto de funciones de prueba cuadráticas para simplificar la determinación de la distancia de una solución al conjunto Pareto óptimo y al frente Pareto óptimo. Usando observaciones empíricas y teóricas, los mismos concluyeron que agregar un objetivo a estos problemas puede hacerlos más difíciles; sin embargo, la diferencia no es significativa. Estos resultados fueron usados para analizar los problemas de escalabilidad señalados por varios investigadores en problemas más complejos, sugiriendo considerar las siguientes preguntas para un tratamiento numérico eficiente de los problemas *many-objective* utilizando algoritmos evolutivos [SLCC11]: (i) cómo determinar qué solución mantener y cuál descartar a fin de converger hacia el frente Pareto, (ii) cómo aumentar la probabilidad de mejorar un individuo y (iii) cómo lidiar con la multimodalidad de un MOP.

Como se puede notar, mientras los primeros trabajos sobre la dificultad de los problemas de optimización con muchos objetivos prestaron atención al número de

objetivos y al ratio de soluciones no-dominadas, los trabajos más recientes se enfocan principalmente en analizar las interrelaciones entre los objetivos. A pesar de grandes avances, aún quedan por responder las preguntas sobre cómo la dimensión y las interacciones entre objetivos afectan la dificultad de diferentes tipos de problemas, y cómo predecir el efecto de incluir objetivos adicionales en el desempeño de un MOEA. Más aún, no existe una definición formal comúnmente aceptada de lo que significa un problema con muchos objetivos que considere tanto, el número de objetivos como las interrelaciones entre ellos.

2.4 Visualización en optimización evolutiva de problemas con muchos objetivos

En general cuando un algoritmo basado en poblaciones tal como un MOEA termina una corrida de optimización, se requiere un tomador de decisiones para seleccionar una solución particular del conjunto de posibles soluciones obtenidas por el algoritmo. En la optimización multiobjetivo, el análisis visual juega un papel importante ayudando a los tomadores de decisiones a: explorar el conjunto de alternativas, estimar la ubicación, rango y forma del frente de Pareto, evaluar conflictos y compensaciones entre objetivos, seleccionar soluciones preferidas, monitorear el progreso o la convergencia de una ejecución de optimización, evaluar el desempeño relativo de diferentes MOEA, entre otros [TF15]. Con más de dos objetivos la visualización de soluciones de compromiso es una tarea no trivial, que se vuelve más cada vez más difícil a medida que aumenta el número de objetivos [Mie03].

Existen tres clases o categorías en las que los métodos para visualizar un conjunto de soluciones para un problema *many-objective* pueden agruparse [vLBB14]. En la primera clase, los objetivos se muestran en grupos de dos o tres a la vez; un ejemplo de este tipo de métodos de visualización son los diagramas de dispersión matricial *scatter plot matrix* [Deb01]. En este tipo de gráficos se muestran los compromisos entre pares de objetivos para facilitar el análisis. Sin embargo cuando el número de objetivos es muy grande mostrar los objetivos en pares puede ser impracticable. Además, la visualización parcial de las soluciones puede impedir visualizar las relaciones existentes entre todo el conjunto de objetivos. Por otra parte, puede resultar complicado comparar gráficamente dos (o más) grandes conjuntos de aproximaciones simultáneamente [vLBB14, TF15].

La segunda clase de métodos de visualización se compone de aquellos que muestran todos los objetivos en un solo gráfico [HY16, IRMD16, IRMD18, LZY17, PMN07, WFE12]. Ejemlos de gráficos en esta categoría son: el gráfico de coordenadas paralelas, los gráficos de barra, sistema coordenado de estrella, diagramas de pétalos, representaciones pentagonales y los mapas de calor [Mie03, PMN07]. Entre las alternativas antes señaladas, el gráfico de coordenadas paralelas [ID90] puede ser considerado el método de visualización más aplicado en los trabajos de investigación sobre resolución de problemas con muchos objetivos utilizando MOEA [FPL05, LZS+10, SAT12]. El gráfico de coordenadas paralelas es un gráfico de dos

dimensiones, donde las etiquetas correspondientes a los objetivos están en el eje horizontal y los valores normalizados de cada uno de los objetivos están en el eje vertical. Usando gráficos de coordenadas paralelas, puede ser posible visualizar las relaciones de conflicto o armonía entre objetivos de manera heurística [FPL05]. Sin embargo, a pesar de su popularidad, el uso de gráficos de coordenadas paralelas para mostrar soluciones de problemas de optimización de muchos objetivos no es sencillo. Por tanto, muy recientemente, Li et al. [LZY17] presentaron un trabajo con pautas para un uso adecuado de gráficos de coordenadas paralelas con muchos objetivos.

Los mapas de calor (*heatmaps*) constituyen otra alternativa para visualizar conjuntos de soluciones en problemas *many-objective* considerando simultáneamente todos los objetivos [PMN07]. Un *heatmap* es un arreglo de dos dimensiones donde cada fila representa una solución y cada columna representa un parámetro u objetivo, y los colores de las celdas representan el valor. A fin de mejorar el análisis visual utilizando *heatmaps*, [PMN07] han propuesto técnicas de agrupamiento jerárquico para mantener juntas soluciones con valores de objetivo similares. Más recientemente, [WFE12] presentaron un método para reordenar un *heatmap* de acuerdo a un ranking de vectores utilizando una medida de similaridad basada en análisis espectral.

El último grupo de paradigmas de visualización presentado aquí considera las características del conjunto solución para reducir el número de objetivos y facilitar un análisis cuantitativo [dFFG15, OS03, KY07b]. Como un ejemplo, [OS03] propusieron un procedimiento de dos etapas en el cual la primera etapa usa un modelo de red neuronal no supervisado, llamado mapa auto-organizado (*self organizing maps*), para proyectar soluciones con m dimensiones en un espacio de menor dimensionalidad, y, en una segunda etapa, se utiliza un agrupamiento jerárquico para facilitar el análisis cuantitativo. Otro trabajo que usó un procedimiento de dos etapas para reducir la dimensionalidad a efectos de visualización es [KY07b]. En este caso, la primera etapa mapea el conjunto solución a un conjunto Pareto de dos dimensiones, mientras que la segunda etapa es el mapeo de las soluciones no-dominadas.

A pesar de las alternativas señaladas anteriormente, aun faltan por desarrollar técnicas que resulten intuitivas, escalables y fáciles de entender para visualizar conjuntos de soluciones de compromiso en problemas con muchos objetivos, siendo la visualización un tópico de investigación interesante, que se espera sea cada vez más explorada.

Capítulo 3
Algoritmos Evolutivos Multiobjetivo

La falta de métodos determinísticos eficientes y eficaces para la resolución de problemas con objetivos múltiples, motivó la búsqueda de métodos alternativos. El notable éxito obtenido por los algoritmos evolutivos en optimización monobjetivo y las características propias de los mismos despertaron el interés de los investigadores en utilizarlos también en optimización multiobjetivo. En la actualidad, la optimización evolutiva multiobjetivo (*Evolutionary Multi-objective Optimization* -EMOO) es un área de investigación muy importante tanto para científicos como para ingenieros, no sólo porque la mayor parte de los problemas consideren por naturaleza objetivos múltiples, sino también porque aún quedan por resolver un sin número de interrogantes en esta disciplina.

Si bien la noción de búsqueda genética para la exploración de soluciones óptimas en problemas con varios objetivos se remonta a fines de la década de los 60s, los primeros algoritmos evolutivos que consideraban de forma simultánea objetivos múltiples se desarrollaron recién a inicios de los noventa. El primer congreso internacional de EMOO se realizó en marzo del 2001 [ZDT+01] y a partir de entonces se vienen realizando bianualmente siendo un importante espacio de intercambio de ideas en el área. En relativamente poco tiempo, la computación evolutiva multiobjetivo, se ha establecido como *el* método para aproximar el frente Pareto-óptimo en problemas de este tipo. Esto se debe fundamentalmente al paralelismo intrínseco de los algoritmos evolutivos que les permite explorar similaridades entre las soluciones de forma eficiente y a la capacidad de capturar varias soluciones Pareto-óptimas en una única corrida de optimización [ZDT00].

Los diferentes métodos para trabajar con objetivos múltiples utilizando algoritmos evolutivos se pueden clasificar, en forma sencilla, en técnicas de primera, segunda y tercera generación. Pertenecen a la primera generación las propuestas iniciales que no consideran conceptos de Pareto así como a los primeros algoritmos evolutivos multiobjetivo basados en Pareto. La segunda generación está caracterizada básicamente por algoritmos basados en dominancia Pareto que incorporan alguna forma de elitismo. Mientras que la tercera generación se caracteriza por una diversidad de enfoques como: la utilización de índices, nuevos criterios de dominancia, técnicas de descomposición, algoritmos coevolutivos, además de la implementación y de-

sarrollo de los primeros algoritmos evolutivos para la resolución de problemas con muchos objetivos.

Puesto que los algoritmos genéticos requieren información escalar sobre el valor de adaptabilidad de los individuos, no es extraño que los primeros enfoques evolutivos para lidiar con objetivos múltiples se basen en la idea de combinar un algoritmo genético simple con métodos de escalarización de la función objetivo. Así, estos primeros enfoques se encargaban de optimizar una función agregada en vez de optimizar la verdadera función multiobjetivo [Coe00, FF94, Vel99].

Schaffer presentó en [Sch84] el primer EA que no utiliza funciones de agregación para resolver problemas multiobjetivo, al que llamó *Vector Evaluated Genetic Algorithm* (VEGA). Este algoritmo utiliza un procedimiento evolutivo que básicamente divide una población genética en subpoblaciones, en las que se considera el valor de adaptación de los individuos de acuerdo a un objetivo distinto para cada subpoblación y realiza el cruzamiento mezclando individuos de distintas subpoblaciones. El resultado final del procedimiento de selección del VEGA corresponde a promediar los valores de cada uno de los objetivos [FF94]. A pesar que Schaffer tuvo algún éxito, especialmente para la resolución de problemas en aprendizaje de máquina [SG85], el método propuesto resultó ineficiente para explorar espacios de objetivos no convexos comportándose bien sólo en una dimensión [Coe00].

En los años siguientes a la presentación de VEGA, se publicaron muy pocos trabajos en el área de optimización evolutiva para problemas con objetivos múltiples, los cuales tampoco lograron el éxito esperado. Estos trabajos proponían, por lo general, alguna combinación de métodos tradicionales con algoritmos evolutivos [FF94, Vel99]. Por ejemplo, en [Fou85] se presenta un algoritmo que utiliza un método de selección basado en ordenamiento lexicográfico, donde pares de individuos se comparan de acuerdo a un objetivo prioritario. En caso de empate se considera el segundo objetivo de mayor prioridad, y así sucesivamente para cada objetivo.

Un renacimiento en el campo se produjo en la década de los noventa con la aparición de los primeros algoritmos que consideraban de forma simultánea la optimización de objetivos múltiples utilizando el concepto de dominancia Pareto. El *Multiobjective Genetic Algorithm* (MOGA) de Fonseca y Flemming [FF93a, FF93b], el *Niched Pareto Genetic Algorithm* (NPGA) de Horn y Nafpliotis [HN93, HNG94], y el *Nondominated Sorting Genetic Algorithm* (NSGA) de Srinivas y Deb [SD93, SD94, SD95] fueron los primeros MOEAs basados en Pareto. Estos algoritmos, pertenecientes a la primera generación de algoritmos evolutivos multiobjetivo, fueron aplicados a una amplia gama de problemas multiobjetivo, donde demostraron un mejor desempeño que el de los enfoques no basados en Pareto.

Como dijimos, a pesar del éxito obtenido por los MOEAs de la primera generación basados en Pareto, una vez que estos encontraban una solución nodominada en una generación, éstas podían perderse cuando se aplicaban los operadores genéticos en las sucesivas generaciones. Para evitar la pérdida de buenas soluciones, el concepto de *elitismo*, utilizado en algoritmos monobjetivo, se extendió al campo de la optimización evolutiva multiobjetivo considerando la existencia de múltiples soluciones posibles. Así, los MOEAs de segunda generación están caracterizados por la

utilización de conceptos de Pareto conjuntamente con alguna forma de elitismo para la búsqueda de soluciones. Algunos algoritmos correspondientes a la segunda generación son: el *Strenght Pareto Evolutionary Algorithm* (SPEA) de Zitzler y Thiele [ZT98, ZT99b], el NSGA-II de Deb, Agrawal, Pratab y Meyarivan [DAPM00], entre otros.

Durante la ejecución de un MOEA basado en Pareto, un conjunto de soluciones Pareto óptimas con respecto a la población genética actual es encontrado en cada generación. A dicho conjunto se le denota como $P(t)$, donde t representa el número de generaciones transcurridas desde el inicio del procedimiento de evolución. Mientras que el frente Pareto correspondiente a $P(t)$ se denota como $PF(t)$. El conjunto de soluciones obtenidas al final de la ejecución de un MOEA basado en Pareto, esto es, el conjunto Pareto conocido, se denota con P_{known}. La notación utilizada para el frente Pareto asociado es PF_{known} respectivamente.

Puesto que los MOEAs se implementan sobre un modelo computacional finito (una computadora), los conjuntos generados son finitos. El conjunto Pareto "verdadero" para el modelo computacional se denota con P_{true} y su correspondiente frente Pareto Óptimo como PF_{true}. Los diferentes conjuntos generados deben ser suficientemente fieles al modelo matemático original, utilizado en el planteamiento del MOP. Cuando se resuelve un MOP utilizando MOEAs, la suposición implícita es que se cumple con al menos una de las siguientes relaciones: $P_{known} = P_{true}$, $P_{known} \subset P_{true}$ o $P_{known} \approx \mathcal{P}^{\star}$ [vZL03]. Donde $\mathcal{P}^{\star}$ representa el frente Pareto óptimo teórico real.

En las secciones que siguen se presentan los MOEAs de primera y segunda generación más relevantes y, luego, en el Capítulo 4 se presentan los algoritmos de tercera generación para problemas con muchos objetivos, mientras que en el Capítulo 5 se introducen los MOEA basados en descomposición.

3.1 MOEAs de primera generación

3.1.1 Multiobjective Genetic Algorithm

El MOGA (*Multiobjective Genetic Algorithm*) fue propuesto en [FF93a, FF93b] como un algoritmo genético multiobjetivo basado en una modificación al algoritmo genético simple en la etapa de selección. El algoritmo se basa en un método de clasificación (*ranking*) de los individuos de la población genética de acuerdo al número de elementos que lo dominan. A partir de la información del procedimiento de clasificación, a cada individuo se le asigna un valor de adaptación. Luego, se utiliza un mecanismo de formación de nichos para evitar la convergencia prematura y mantener la diversidad.

El procedimiento de clasificación utilizado en MOGA asigna en cada generación a todos los elementos de la población genética un *ranking* igual al número de elementos

que lo dominan más uno. Esto es, denotando como $p_i^{(t)}$ al número de individuos en la población que dominan a un individuo $P[i](t)$ y como $P[i](t)_{rank}$ a su *ranking* para todo $i \in \{1, \ldots, N\}$, el procedimiento de clasificación del MOGA hace:

$$P[i](t)_{rank} = 1 + p_i^{(t)}$$

siendo:

$$p_i^{(t)} = ||\{\ j \mid j \in \{1, \ldots, N\} \wedge P[j] \prec P[i]\ \}||_c\ , \tag{3.1}$$

donde:

$||.||_c$ denota cardinalidad

Todos los elementos nodominados de la población $P(t)$ recibirán una clasificación de 1, mientras los dominados tendrán una clasificación mayo a 1. En este esquema de clasificación, no todos los niveles están necesariamente representados en la población en una generación particular. Clasificados todos los elementos de la población, la asignación de *fitness* en MOGA se realiza de la siguiente manera:

1. Ordenar la población de acuerdo a la clasificación.
2. Asignar valor de adaptación a los individuos interpolando de la mejor clasificación a la peor de todas, de acuerdo a alguna función (usualmente lineal).
3. Promediar el valor de adaptación de los individuos con la misma clasificación, de tal forma que todos puedan ser muestreados a la misma tasa. Este procedimiento mantiene el *fitness global* de la población constante mientras que se mantiene una presión de selección apropiada.

Debido a errores aleatorios en el proceso de selección, cuando se tienen múltiples óptimos equivalentes, las poblaciones finitas convergen hacia una sola de ellas [FF93a, FF95]. Este fenómeno ha sido observado en algoritmos genéticos utilizados para la optimización de funciones multimodales monobjetivo. Para evitarlo, se desarrollaron e implementaron con éxito las técnicas de distribución del valor de adaptación y los métodos de formación de nichos (sección 1.5). En MOGA se utiliza una extensión de estas técnicas para la optimización multiobjetivo implementando *sharing* en el espacio objetivo, utilizando la ecuación 1.8 entre pares de elementos nodominados y norma infinita o euclidiana para el cálculo de distancia, a fin de evolucionar una representación uniformemente distribuida de la superficie de compromiso global.

El MOGA presentado en [FF93a] utiliza restricción de apareamiento a fin de evitar la recombinación entre cromosomas de nichos diferentes (sección 1.5.2) como en [DG89]. Si bien dicha técnica demostró ser adecuada para problemas con objetivo único, la pobre efectividad de la misma para problemas multiobjetivo ha hecho que en trabajos posteriores no se haya continuado con su utilización [PF01, ZDT99].

3.1.2 Nondominated Sorting Genetic Algorithm

El *Nondominated Sorting Genetic Algorithm* (NSGA) [SD93, SD94, SD95], al igual que el MOGA, difiere del algoritmo genético simple sólo en la manera en que procede la selección, específicamente en la manera de asignar el valor de adaptabilidad. Los operadores de cruzamiento y mutación permanecen sin modificaciones. Así mismo, tanto el NSGA como el MOGA se basan en la utilización de un método de selección por clasificación para enfatizar los puntos actuales nodominados y un método de *niching* [Mah95, Mah97] para preservar la diversidad en la población. Sin embargo, el NSGA utiliza un procedimiento de clasificación que, a diferencia del utilizado en MOGA, no considera el número de elementos que dominan a un individuo para clasificarlo, sino su nivel de dominancia. Esto es, el NSGA utiliza en forma directa la idea de realizar un procedimiento de búsqueda de óptimos para optimización multiobjetivo basado en una clasificación según la nodominancia, conforme fuera presentado en [Gol89].

El procedimiento de asignación del valor de adaptación basado en clasificación por nodominancia (*nondominated sorting procedure*) utilizado en NSGA se presenta en el Algoritmo 3. En este procedimiento, primeramente se identifican los elementos nodominados de la población considerada; estos constituyen el primer frente nodominado de la población. Para ello, inicialmente, se marcan todos los elementos de la población como nodominados. En el Algoritmo 3, se denota con un subíndice dom a un campo del individuo que indica si es dominado o no. Además, se utiliza el sub-índice $objs$ para indicar el vector objetivo de un individuo. Inicialmente, a todos los individuos de la población se le asigna en su correspondiente campo dom un valor de falso, esto es, inicialmente todos son nodominados. Se inicializa a cero un contador de frentes f. Luego, el primer individuo de la población, $P[i]$ con $i = 1$, se compara con relación a la dominancia con respecto a cada uno de los individuos de la población hasta encontrar un elemento que lo domine. Cuando esto ocurre, se marca el individuo $P[i]$ como dominado y se repite el procedimiento para el siguiente individuo en la población hasta que $i = N$.

Una vez que todos los individuos se comparan con respecto a la dominancia, los individuos no marcados son los nodominados con respecto a la población P. Estos individuos constituyen el primer frente de individuos nodominados, denotado por $\mathcal{F}^f$, con $f = 0$. Note que, en el peor de los casos, cuando todos son nodominados, el procedimiento de clasificación presentado requiere N^2 comparaciones con respecto a la dominancia.

A todos los elementos del frente identificado se le asigna un valor de adaptación df, que en el caso del Algoritmo 3 inicialmente es igual al tamaño de la población genética N. Este valor de adaptación asignado a cada elemento del frente identificado, es un valor de adaptación ficticio (*dummy fitness*) que provee a todos los individuos del primer frente de un igual potencial reproductivo [SD93]. Con el fin de mantener la diversidad en la población, el valor de adaptación asignado a cada individuo del primer frente se comparte entre los elementos que componen dicho frente, dividiéndolo por la cuenta del nicho en el frente. Para ello se implementa el procedimiento de *fitness sharing* presentado en el Algoritmo 4 [SD93].

Algoritmo 3: Procedimiento de asignación de valor de adaptación en NSGA.

```
1  Procedimiento: fitness_assignement_nsga
2  Entrada : Población genética P
3  Salida  : Población genética P con el valor de adaptación asignado a cada individuo
4  ∀ P[i] ∈ P; P[i]_dom = f also // marcar todos los elementos como no-dom.
5  f = 0 // hacer la cuenta de frentes igual a 0
6  df = N ;  // iniciar el dummy fitness igual al tamaño de la población P
   // mientras queden elementos sin considerar
7  mientras ||P||_c > 0 hacer
8      f = f + 1 ;                          // incrementar la cuenta de frentes
       // para cada elemento P[i] de P
9      para i = 1 to ||P||_c hacer
           // para cada elemento P[j] de P
10         para j = 1 to ||P||_c hacer
               // si el elemento P[j] domina a P[i]
11             si P[j] ≺ P[i] entonces
12                 P[i]_dom = true;     // marcar elemento P[i] como dominado
13                 break ;                                  // salir del ciclo
14             fin
15         fin
16     fin
       // los elementos de P no marcados conforman el frente f
17     F^f = {P[i] | P[i]_dom = f also};
       // asignar a cada elemento del frente f un fitness grande
18     ∀F[i]^f ∈ F^f ; F[i]^f_fitness = df ;
       // aplicar sharing a los elementos del frente F^f
19     fitness_sharing_procedure(F^f );
       // guardar el peor valor de fitness del frente f en df
20     df = min{F[i]^f_fitness | F[i]^f ∈ F^f } ;
       // reducir df en ε y usarlo como dummy fitness del siguiente
          frente
21     df = df − ε; // extraer de P el frente recientemente
          identificado reasignando los índices
22     P = P \ F^f ;
       // marcar todos los elementos de la población como no dominados
23     ∀P[i] ∈ P; P[i]_dom = f also;
24 fin
   // recomponer P con los elementos de todos los frentes
25 P = ∪_{i=1}^{f} F^i;
```

Determinado el primer frente de individuos nodominados de la población y finalizado el procedimiento de *fitness sharing*, se obtiene el peor valor de adaptación entre los individuos del primer frente y se guarda en df. Un valor ligeramente inferior a este será asignado como *dummy fitness* para los elementos del siguiente frente considerado, por lo que df se reduce en un $\epsilon > 0$. Luego, los individuos del primer frente son eliminados de la población genética P de forma temporal. El procedimiento de clasificación se repite para los elementos de la población P sin los

elementos del primer frente. Para ello, los elementos en P se vuelven a marcar todos como no dominados. El procedimiento continúa hasta que todos los frentes hayan sido identificados, es decir todos los elementos tengan asignado su correspondiente valor de adaptabilidad.

Algoritmo 4: Procedimiento de *fitness sharing* utilizado en NSGA.

fitness_sharing_procedure Entrada: conjunto de individuos $\mathcal{F}^f$ del f -ésimo frente nodominado, radio de nicho σ_{share}. **Salida :** conjunto de individuos $\mathcal{F}^f$ con *fitness* compartido asignado a cada elemento. **para** $i = 1$ *to* $||\mathcal{F}^f||_c$ **hacer**
  para $j = 1$ *to* $||\mathcal{F}^f||_c$ **hacer**
    `// calcular valor de sharing (Ecuación 1.8 para` $P = \mathcal{F}^f$ `)`
    $d_{ij} = ||\mathcal{F}^f[i]_{objs} - \mathcal{F}^f[j]_{objs}||$; `//` d_{ij} `dist. Euclid. entre objs. de` $\mathcal{F}^f[i]$ `y` $\mathcal{F}^f[j]$
  si $d_{ij} \leq \sigma_{share}$ **entonces**
    $sh = 1 - (\frac{d_{ij}}{\sigma_{share}})^2$
  en otro caso
    $sh = 0$
  $\mathcal{F}^f[i]_{ncount} = \mathcal{F}^f[i]_{ncount} + sh$; `// calcular la cuenta de nicho para` $\mathcal{F}^f[i]$ `(Ecuación 1.9)`
$\mathcal{F}^f[i]_{fitness} = \frac{\mathcal{F}^f[i]_{fitness}}{\mathcal{F}^f[i]_{ncount}}$; `// calcular el` *fitness compartido* `(Ecuación 1.10)`

Finalizado el procedimiento de asignación del valor de adaptabilidad, la población es sometida al operador de selección utilizando el método de ruleta ponderada (sección 1.3.1) y a los demás operadores genéticos como es usual. Puesto que los elementos del primer frente tienen valores de adaptación mejores que los de cualquier otro frente, estos siempre obtienen más copias que el resto de la población. Así, este método dirige la búsqueda hacia las regiones de nodominancia, lo que finalmente conduciría al frente Pareto óptimo.

3.1.3 Niched Pareto Genetic Algorithm

En [HN93, HNG94] se propone el *Niched Pareto Genetic Algorithm* (NPGA) el cual, como los casos anteriores, difiere del algoritmo genético simple únicamente en el procedimiento de selección. Sin embargo, este algoritmo no utiliza ningún procedimiento de clasificación por dominancia, sino que propone una variación del procedimiento de selección por torneo (sección 1.3.1) llamado torneo por dominancia Pareto y la utilización de un procedimiento de *sharing* para romper empates y determinar un ganador.

En los procedimientos de selección por torneo, un conjunto de elementos son seleccionados en forma aleatoria de la población genética actual y el mejor de este conjunto es seleccionado. Estos procedimientos, diseñados para EAs aplicados en

problemas monobjetivo, asumen que se desea una única solución al problema. Esto es, se desea que luego de cierto número de generaciones la población converja a una solución [HN93]. Para evitar esta convergencia a un único punto, en [HN93] se propone utilizar el operador de dominancia para la selección de individuos que compiten para ser seleccionados.

Si bien la relación de dominancia Pareto conduce de forma directa a un torneo binario, en el cual dos individuos seleccionados de manera aleatoria se comparan de acuerdo a la domancia y aquel que domine al otro gana. Sin embargo, este tamaño de muestra es insuficiente para estimar que tan buena es una solución con respecto al conjunto. Por ello, en [HN93] se propone un método distinto para la realización de la selección por torneo Pareto.

El Algoritmo 5 presenta el procedimiento de selección por torneo Pareto utilizado en el NPGA para seleccionar un conjunto de individuos, de tamaño igual al de la población genética, para la aplicación posterior de otros operadores genéticos conocido como conjunto para apareamiento (*matting pool*). En el procedimiento de selección del NPGA se seleccionan de la población genética de forma aleatoria dos individuos candidatos $P[i]$ y $P[j]$. También se selecciona de la población genética un conjunto de comparación P_{dom}. Luego, cada uno de los candidatos se compara contra cada uno de los elementos del conjunto de comparación con relación a la dominancia. Si un candidato es dominado por el conjunto de comparación y el otro no, se selecciona el nodominado. El tamaño del conjunto de comparación (t_{dom}) proporciona el control sobre la presión de selección, lo que se llama presión de dominancia (*domination pressure*).

Si ninguno de los candidatos o ambos son dominados por el conjunto de comparación, entonces se utiliza una forma simplificada de *sharing* para elegir un ganador. En este procedimiento de *sharing*, se calcula el número de individuos previamente seleccionados que se encuentran en las cercanías de cada solución candidata [HNG94]. En caso que M se encuentre vacío, el cálculo de la cuenta de nicho se realiza sobre (P_{dom}). Debido al interés en mantener la diversidad, el candidato que tenga el menor número de individuos en su nicho es considerado como el mejor y por lo tanto seleccionado. Si el empate persiste, se elige uno de estos individuos de forma aleatoria.

La Figura 3.1 representa el concepto de niching para romper empates en NPGA, para un problema de minimización de dos objetivos [EG02] cuando el *matting pool* se encuentra vacío. En dicha figura, se presenta un conjunto de individuos para la comparación y dos individuos candidatos (1 y 2). Ambos candidatos no nodominados con respecto al conjunto comparación. Para romper el empate, se debe considerar el número de individuos en el conjunto comparación que se encuentran en las cercanías de los individuos. Aquel individuo con menos "vecinos" será el ganador. En este caso, existen más individuos en las cercanías del candidato 1 y menos en las cercanías del candidato 2. Entonces, el candidato 2 es el seleccionado para la posterior aplicación de otros operadores evolutivos.

Algoritmo 5: Procedimiento de selección por torneo Pareto, utilizado por el NPGA.

```
1  procedure: selection_npga
2  Entrada: Población P, radio de nicho σ_share, presión de dominancia t_dom.
3  Salida: Conjunto M de individuos seleccionados para la aplicación de otros operadores
   genéticos.
4  M = ∅ ;            // La población M inicialmente se encuentra vacía
   // Hasta que el número de elementos en M sea igual al de P
5  repetir
6      Seleccionar aleatoriamente dos individuos P[i], P[j] ∈ P
7      Seleccionar aleatoriamente un conjunto de comparación P_dom ⊆ P con t_dom
         individuos.
8      si P[i] es nodominado con relación a P_dom y P[j] es dominado entonces
9          M = M + {P[i]} ;              // agregar el individuo P[i] a M
10     en otro caso
11         si P[j] es nodominado con relación a P_dom y P[i] es dominado entonces
12             M = M + {P[j]} ;   // agregar el individuo P[j] al conjunto
                 M
13         en otro caso
14             si ||M||_c = 0 entonces
                   ;    // Calcular cuantos individuos en P_dom están a una
                    distancia σ_share de P[i] y P[j]
15                 n_i = ||{ k | k ∈ {1, ..., t_dom} ∧ ||P[i]_objs − P_dom[k]_objs||_d ≤
                    σ_share}||_c n_j = ||{ k | k ∈
                    {1, ..., t_dom} ∧ ||P[j]_objs − P_dom[k]_objs||_d ≤ σ_share}||_c
16             en otro caso
                   // Calcular cuantos individuos en M están a una
                      distancia σ_share de P[i] y P[j]
17                 n_i = ||{ k | k ∈ {1, ..., ||M||_c} ∧ ||P[i]_objs − M[k]_objs||_d ≤
                    σ_share}||_c n_j = ||{ k | k ∈
                    {1, ..., ||M||_c} ∧ ||P[j]_objs − M[k]_objs||_d ≤ σ_share}||_c
18             si n_i < n_j entonces
19                 M = M + {P[i]}; // si P[i] tiene menos individuos en su
                   radio de nicho agregarlo a M
20             en otro caso
21                 si n_j < n_i entonces
22                     M = M + {P[j]};// si P[j] tiene menos individuos en
                        su radio de nicho agregarlo a M
23                 en otro caso
24                     M = P[rand(j, i)] ;        // si el empate persiste se
                        selecciona en forma aleatoria
25 hasta que ||M||_c = N;
```

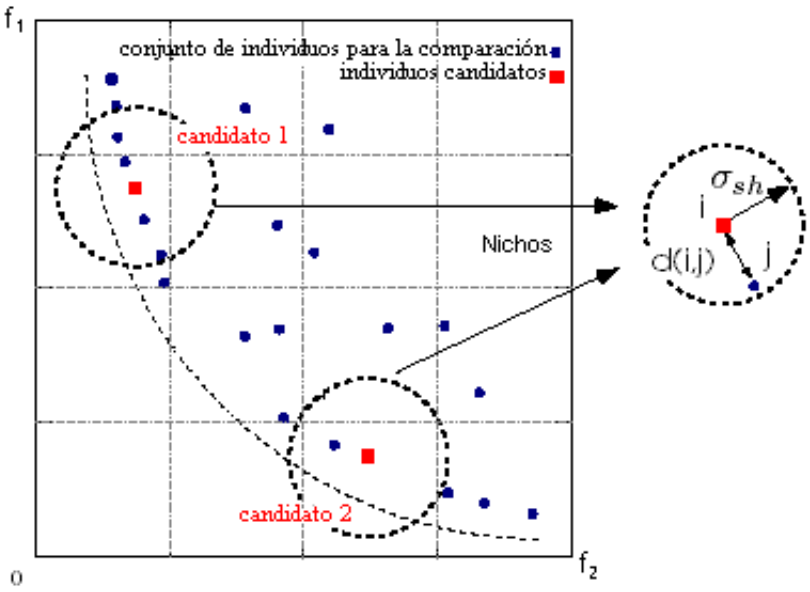

Fig. 3.1 Concepto de Niching en NPGA.

3.2 MOEAs de segunda generación

3.2.1 Strength Pareto Evolutionary Algorithm

En [ZT98, ZT99b] se presenta un nuevo enfoque evolutivo para la optimización multiobjetivo, el *Strength Pareto Evolutionary Algorithm* (SPEA). Este algoritmo difiere de los anteriores en varios aspectos. Primeramente, utiliza dos poblaciones, incorporando el concepto de elitismo a través del almacenamiento de las soluciones nodominadas en una población externa, la cual participa del procedimiento de selección. Además, el cálculo del valor de adaptación se realiza utilizando un procedimiento basado en la asignación de un valor de fuerza (*strength*) a todos los elementos de la población externa. Este procedimiento induce la formación de nichos a partir del concepto de dominancia Pareto, llamado *niching* por *strength* [ZT98]. Puesto que el conjunto de soluciones en la población externa puede ser grande y ésta interviene en el proceso evolutivo, también se utiliza un procedimiento de *clustering* para reducir el número de soluciones en dicho conjunto cuando este número supera un máximo permitido.

Los pasos principales y el diagrama de flujo del SPEA se muestran en el Algoritmo 6 y en la Figura 3.2 respectivamente. Inicialmente se crea una población genética P de tamaño N y un conjunto externo de soluciones nodominadas P' vacío. Luego, las soluciones nodominadas de la población genética son copiadas al conjunto P'.

Algoritmo 6: Procedimiento principal del SPEA.

1 **Paso 1**: Generar una población inicial P y crear el conjunto nodominado externo $P' = \emptyset$;
2 **Paso 2**: Copiar los miembros nodominados de P a P';
3 **Paso 3**: Eliminar las soluciones en P' cubiertas por cualquier otro miembro de P';
4 **Paso 4**: Si el número de soluciones en el almacenamiento externo excede un máximo N', reducir P' por medio de *clustering* (Algoritmo 7);
5 **Paso 5**: Calcular el *fitness* de cada individuo en P, así como en P' (Algoritmo 8);
6 **Paso 6**: Seleccionar individuos de $P(t) + P'(t)$ (unión multiconjunto), hasta que el pool de apareamiento $P(t+1)$ se llene;
7 **Paso 7**: Aplicar los operadores de mutación y cruzamiento específicos del problema como es usual. [**Paso 6**] $t = t + 1$;
8 **Paso 8**: Si se alcanza el máximo número de generaciones parar, sino ir al Paso 2;

De este conjunto se eliminan las soluciones cubiertas por cualquier otro miembro de dicho conjunto.

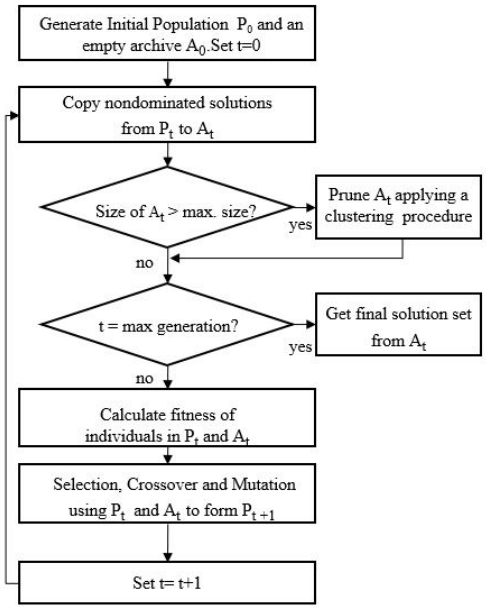

Fig. 3.2 Diagrama de flujo principal del SPEA.

Definición 3.1 *Soluciones cubiertas en una población:* Dada una población P de tamaño N, para cualquier par de individuos en la población $P[i]$ y $P[j]$, $i, j \in \{1, \ldots, N\}$, se dice que $P[i]$ cubre a $P[j]$ si y sólo si: $i \neq j$ y $P[i] \preceq P[j]$. Es decir, si dados dos individuos en una población, con índices distintos i y j, el individuo

en la posición i cubre al individuo en j si y sólo si considerando sus objetivos el individuo $P[i]$ es mejor o igual al individuo $P[j]$, esto es $P[i] \preceq P[j]$.

En ciertos problemas, el conjunto Pareto puede ser extremadamente grande, incluso infinito. Desde el punto de vista del tomador de decisiones, recibir todas las soluciones nodominadas encontradas es poco útil cuando el número de éstas excede límites razonables. Más aún, el tamaño del conjunto externo nodominado influencia el comportamiento del SPEA. Por un lado, debido a que P' participa en la selección, demasiadas soluciones nodominadas podrían reducir la presión de selección y hacer la búsqueda más lenta. Por otro lado, el mecanismo de *niching por strength* se basa en una disposición uniforme de los puntos en P'. Si estos puntos no están distribuidos en forma uniforme, el proceso de asignación de *fitness* posiblemente tenderá hacia ciertas regiones del espacio de búsqueda, conduciendo esto a una distribución desbalanceada de la población. Por lo tanto, si el número de elementos en el conjunto de nodominados supera un número máximo dado (N'), se procede a reducir el conjunto utilizando el procedimiento de *clustering* que se describe en el Algoritmo 7.

Algoritmo 7: Procedimiento de *clustering* para el SPEA.

1 **Paso 1:** Inicializar el *conjunto de clusters* C; cada punto nodominado externo $P'[i] \in P'$ constituye un cluster distinto, C es el conjunto unión :

$$C = \bigcup_{i}^{||P'||_c} \{P'[i]\}$$

Paso 2: Si $||C||_c \leq N'$, es decir si el número de clusters en C es menor o igual a N', ir al Paso 5, sino ir al Paso 3.

2 **Paso 3:** Calcular la distancia de todos los posibles pares de clusters. La distancia d_{12} entre dos clusters c_1 y $c_2 \in C$ está dada por la distancia promedio entre pares de individuos pertenecientes a los dos clusters.

$$d_{12} = \frac{1}{||c_1||_c \cdot ||c_2||_c} \cdot \sum_{i_1 \in c_1, i_2 \in c_2} \| i_1 - i_2 \|$$

3 **Paso 4:** Determinar dos clusters c_1 y c_2 con distancia mínima y amalgamarlos en un cluster mayor: $C = C \setminus \{c_1, c_2\} \cup \{c_1 \cup c_2\}$. Es decir se eliminan c_1 y c_2 del conjunto de clusters y se agrega a C un nuevo cluster formado de la unión c_1 con c_2. Ir al **Paso 2**.

4 **Paso 5:** Computar el conjunto nodominado reducido seleccionando de cada cluster el punto con distancia mínima a todos los otros puntos en el cluster considerado.

Luego, se calcula el valor de *fitness* utilizando el Algoritmo 8. Primero, a cada individuo del conjunto externo de nodominados P' se le asigna un valor real entre en $[0, 1)$. Este valor indica la *utilidad* de un individuo y se le denomina su *strength*. El *strength* de un individuo $P'[i]$ está directamente relacionado con el número de elementos en la población genética para los cuales es mejor o igual. Esto es, dados dos individuos cualquiera en la población externa, el que domina de forma débil a más elementos en la población genética tiene el valor de *strength* mayor. El cálculo

del valor del *strength* para un individuo en la posición i de la población externa P' es:

$$P'[i]_{strength} = \frac{||\{j|j \in \{1, \ldots N\} \wedge P'[i] \preceq P[j]\}||_c}{N+1} \tag{3.2}$$

El valor de adaptación del individuo $P'[i]$ será igual a la inversa de su valor de *strength*, esto es

$$P'[i]_{fitness} = \frac{1}{P'[i]_{strength}} \tag{3.3}$$

Luego de calcular el valor de adaptación de los individuos en la población externa, también basados en el cálculo del valor de *strength*, se calcula el valor de adaptación de los individuos en la población genética . El *strength* de un individuo $P[j] \in P$ se calcula a partir de los *strengths* de todas las soluciones externas no dominadas $P'[i] \in P'$ que lo dominen. Esto es:

$$P[j]_{strength} = 1 + \sum_{i, P'_i \preceq P_j} P'[i]_{strength} \tag{3.4}$$

Nuevamente, el valor de adaptación del individuo $P[j]$ será igual a la inversa de su valor de *strength*:

$$P[j]_{fitness} = \frac{1}{P[j]_{strength}} \tag{3.5}$$

Como resultado del procedimiento, los individuos en P' que cubren una cantidad menor de individuos en P reciben mayores valores de *fitness* que los otros miembros de la población. Los individuos que tienen muchos vecinos en el nicho son penalizados debido al alto valor del *strength* de los puntos nodominados asociados, esto es conocido como el *niching* por *strength*. La idea detrás de este mecanismo es preferir siempre los individuos que están más cerca del frente Pareto óptimo y al mismo tiempo, distribuirlos en toda la superficie factible. La principal diferencia de este método con respecto al *fitness sharing* es que el nicho no esta definido en términos de la distancia, sino de acuerdo a la dominancia de Pareto y no requiere el establecimiento de ningún parámetro predefinido. Esto resulta más adecuado ya que en muchos problemas del mundo real la distancia no tiene un significado práctico si se computa en el espacio objetivo, ya que cada objetivo puede estar expresado en magnitudes totalmente diferentes y no comparables entre sí, por ejemplo: millones de dólares, segundos, metros, etc.

Finalizado el procedimiento de asignación del valor de adaptabilidad, se procede a la selección de los individuos para el cruce. Esta selección se realiza entre los elementos de ambas poblaciones. En [ZT99b] se utiliza un procedimiento de selección basado en torneo binario. Luego se aplican los operadores genéticos como es usual hasta que el número máximo de generaciones sea alcanzado.

En [ZLT01a] se presenta el algortimo SPEA2, el cual busca eliminar algunos problemas potenciales del algoritmo original. Estos problemas son: la posibilidad de que individuos dominados por el mismo número de individuos tengan el mismo valor de adaptación y que el procedimiento de *clustering* puede perder soluciones en los

Algoritmo 8: Algoritmo de asignación de *fitness* para el SPEA.

1 **Paso 1:** A cada solución $P'[i] \in P'$ se le asigna un valor real $P'[i]_{strength} \in [0, 1)$, llamado su *strength*. $P'[i]_{strength}$ es proporcional al número de miembros de la población genética, $P[j] \in P$ para los cuales $P'[i] \preceq P[j]$. Sea N el número de individuos en P, entonces, el valor del *strength* para el individuo i de la población externa P' es:

$$P'[i]_{strength} = \frac{||\{j|j \in \{1, \ldots N\} \wedge P'[i] \preceq P[j]\}||_c}{N+1}$$

Esto es, el número de individuos en P que son cubiertos por $P'[i]_{strength}$ dividido el tamaño de la población genética más uno.

2 **Paso 2:** El valor de *fitness* de $P'[i]$ es:

$$P'[i]_{fitness} = \frac{1}{P'[i]_{strength}}$$

.

3 **Paso 3:** El *strength* de un individuo $P[j] \in P$ es calculado sumando los *strengths* de todas las soluciones nodominadas $P'[i] \in P'$ que lo cubren.

$$P[j]_{strength} = 1 + \sum_{i, P'_i \preceq P_j} P'[i]_{strength}$$

4 **Paso 4:** El valor de *fitness* de $P[j]$ es:

$$P[j]_{fitness} = \frac{1}{P[j]_{strength}}$$

.

límites. Por ello, la segunda versión del SPEA propone un mecanismo mejorado de asignación del valor de adaptación, el cual considera no solo el número de individuos que un individuo domina a otro sino también el número de individuos a los que el domina. Igualmente, se propone un método mejorado de cálculo de la densidad de la vecindad y un mecanismo para truncar la población externa que garantiza la preservación de las soluciones en los límites. La Figura 3.3 presenta el diagrama de flujo principal del SPEA2. Como se puede observar, guarda grandes semejanzas con el diagrama presentado en la Figura 3.2 correspondiente a la primera versión del algoritmo.

3.2.2 Nondominated Sorting Genetic Algorithm II

En [DAPM00] se presentan los detalles del *Nondominated Sorting Genetic Algorithm II* (NSGA-II), un nuevo algoritmo basado en clasificación por nodominancia para asignar el valor de adaptabilidad a los elementos de la población genética. Son varias las características del NSGA-II que lo hacen diferente del NSGA original [SD93, SD94]. En primer lugar, el NSGA-II incorpora un mecanismo de preservación de élites que asegura el mantenimiento de las buenas soluciones encontradas

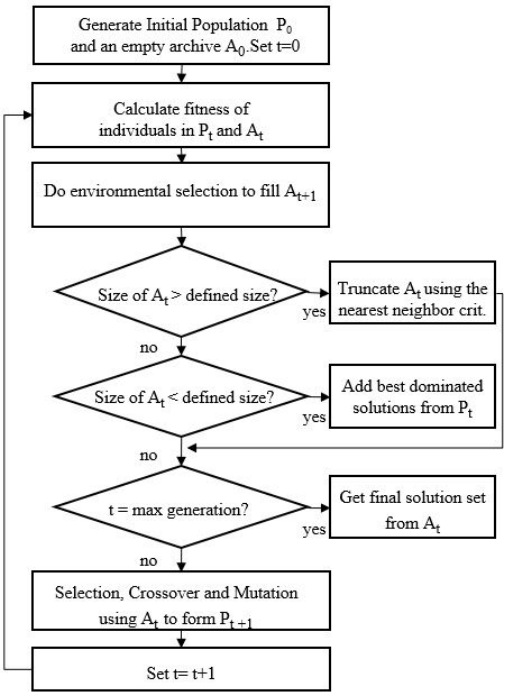

Fig. 3.3 Diagrama de flujo principal del SPEA2.

con anterioridad. En segundo lugar, el NSGA-II utiliza un procedimiento rápido de clasificación por nodominancia (*fast nondominated sorting procedure*) el cual incorpora un procedimiento especial de almacenamiento a fin de reducir la complejidad computacional del algoritmo presentado en [SD93]. Por último, a diferencia de su antecesor, el NSGA-II no requiere de ningún parámetro ajustable (ejemplo: σ_{share}) [DAPM00]. El conjunto de pasos básico del NSGA-II se muestra en la Figura 3.4. En lo que sigue se explicará el detalle del funcionamiento de este algoritmo.

El Algoritmo 9 presenta el procedimiento rápido de clasificación por nodominancia sobre el cual basa su funcionamiento el NSGA-II. En este algoritmo primeramente se determinan, para cada solución $P[i]$ de una población P a clasificar:

1. El conjunto S_i de las soluciones dominadas por $P[i]$.
2. El número nd_i de soluciones que dominan a $P[i]$.

Para ello, se comparan entre si con respecto a la dominancia todos los miembros de la población a clasificar. Si un elemento $P[i]$ domina a un elemento $P[j]$, este último se agrega a un conjunto S_i. En caso contrario, si $P[j]$ es dominado por $P[i]$ entonces se incrementa el valor del contador nd_i de soluciones que dominan a $P[i]$. Note que un elemento $P[j]$ de la población puede pertenecer simultáneamente a nd_j conjuntos de soluciones dominadas.

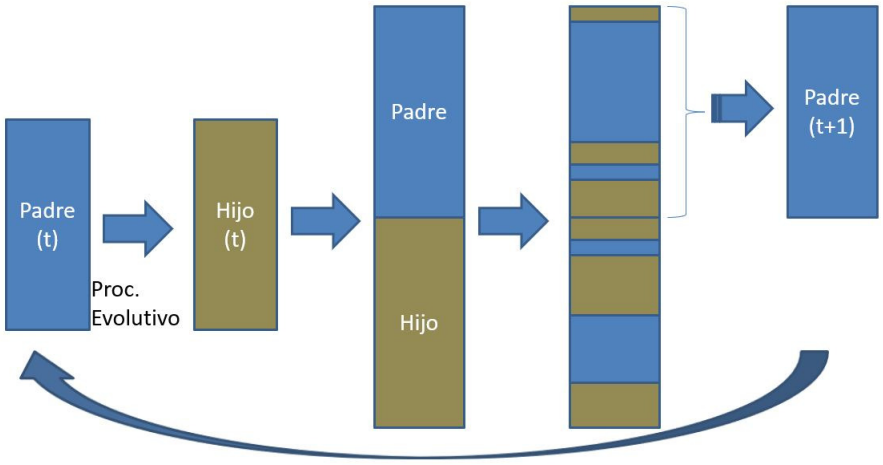

Fig. 3.4 Diagrama básico del algoritmo NSGA-II [DAPM00].

Algoritmo 9: Procedimiento Rápido de Ordenamiento por Nodominancia del NSGA-II.

```
1  ht]
2  Procedimiento :fast_nondominated_sort
3  Entrada :Una población P
4  Salida  : Una lista de los frentes nodominados F
5  para each P[i] ∈ P hacer
6      para each P[j] ∈ P hacer
7          si P[i] ≺ P[j] entonces
8              S_i = S_i ∪ P[j] ;                          // incluir P[j] en S_i
9          si no, si P[j] ≺ P[i] entonces
10             nd_i = nd_i + 1 ;                           // incrementar nd_i
11     si nd_i = 0 entonces
           // ninguna solución domina a P[i]
12         F^1 = F^1 ∪ {P[i]} ;                  // P[i] pertenece al primer frente
13     f = 1 ;                        // el contador de frentes se hace igual a 1
14     mientras F^f ≠ ∅ hacer
15         H = ∅
16         Foreach P[i] ∈ F^f ;                    // para cada miembro P[i] en F^f
17         Foreach P[j] ∈ S_i ;       // modificar cada miembro del conjunto S_i
18         nd_j = nd_j − 1 ;                          // decrementar nd_j en uno
19         si nd_j = 0 entonces
20             H = H ∪ {P[j]} ;  // P[j] es dominado sólo por elementos de
                 F^f
21     f = f + 1 ;                       // se incrementa el contador de frentes
22     F^f = H ;  // el frente actual está formado con los elementos de H
```

Una vez determinados tanto el conjunto de soluciones dominadas como el número de soluciones que lo dominan, para cada elemento de la población, se forma el primer frente de soluciones nodominadas, denotado $\mathcal{F}^1$, con todos los elementos cuya cuenta de individuos que los dominan es igual a 0. El algoritmo prosigue recorriendo para cada elemento $P[i] \in \mathcal{F}^1$ su respectivo conjunto de soluciones dominadas S_i, reduciendo para cada elemento $P[j] \in S_i$ el valor de nd_j que le corresponde. Cuando el valor de nd_j se hace igual a 0, se agrega $P[j]$ a una lista $\mathcal{H}$ inicialmente vacía. Cuando se han recorrido todos los elementos del primer frente, en $\mathcal{H}$ quedan los elementos que sólo son dominados por los elementos del primer frente, es decir, los elementos del segundo frente $\mathcal{F}^2$. Luego se consideran los elementos de $\mathcal{F}^2$ repitiendo el procedimiento para cada elemento de dicho frente. El procedimiento finaliza cuando no quedan elementos cuya cuenta de elementos que lo dominan se haga cero, esto es cuando todos los frentes han sido identificados y $\mathcal{H} = \emptyset$.

El NSGA-II incluye también un procedimiento para estimar la densidad de soluciones alrededor de cada solución particular $\mathcal{F}^f[i]$ con respecto a los demás elementos del frente $\mathcal{F}^f$. El Algoritmo 10 ilustra dicho procedimiento. En éste, para cada individuo $\mathcal{F}^f[i]$, se calcula un valor denotado $\mathcal{F}^f[i]_{distance}$ que sirve como un estimador del tamaño del cuboide más grande que encierra la solución sin incluir ningún otro punto de la población (a esto se le llama *crowding distance*). Para ello, se determina para cada individuo del frente considerado y para cada objetivo, cual es el siguiente menor y el siguiente mayor valor dentro de dicho frente. Esto se consigue ordenando de menor a mayor los elementos del frente para cada uno de los objetivos considerados. Luego, el valor de la distancia de *crowding* de un elemento $\mathcal{F}^f[i]$ se calcula sumando las distancias entre los individuos inmediatamente mayor y menor considerando cada objetivo. Note que los objetivos usualmente se encuentran expresados en unidades diferentes por lo que para obtener una estimación correcta es conveniente la normalización de los diferentes objetivos.

Algoritmo 10: Procedimiento de asignación de distancia de *crowding* utilizado en NSGA-II.

Procedimiento: crowding_distance_assignment
Entrada: un conjunto de soluciones $\mathcal{F}^f$
Salida : el conjunto $\mathcal{F}^f$ con las distancia de *crowding* de sus elementos calculada
$l = ||\mathcal{F}^f||_c$
`// inicializar a cero la distancia de todos los elementos de` $\mathcal{F}^f$
Para cada i, $\mathcal{F}^f[i]_{distance} = 0$
`// Evaluar para cada objetivo considerado`
para $j = 1$ *to* k **hacer**
    Ordenar $\mathcal{F}^f$ de acuerdo al objetivo j
    $\mathcal{F}^f[1]_{distance} = \mathcal{F}^f[l]_{distance} = \infty$ **para** $i = 2$ *to* $(l-1)$ **hacer**
        $\mathcal{F}^f[i]_{distance} =$
            $\mathcal{F}^f[i]_{distance} + (\mathcal{F}^f[i+1]_{objetivo[j]} - \mathcal{F}^f[i-1]_{objetivo[j]})$
    `// El sub-índice` $objetivo[j]$ `representa al valor del objetivo` j

Además de definir un procedimiento de asignación de *crowding*, se define también un operador de comparación por *crowding* ($\geq_n$). El objetivo de este operador es guiar el proceso de selección en las diferentes etapas del algoritmo hacia un frente Pareto óptimo uniformemente distribuido.

Definición 3.2 **Operador de** *Crowding* **($\geq_n$):** Asumiendo que cada uno de los individuos en la población tiene dos atributos: la posición en la clasificación por nodominancia ($P[i]_{rank}$) y su distancia local de *crowding* ($P[i]_{distance}$), se define el orden parcial $\geq_n$ como:

$$P[i] \geq_n P[j] \text{ si } (P[i]_{rank} < P[j]_{rank}) \text{ o}$$
$$((P[i]_{rank} = P[j]_{rank}) \text{ y } (P[i]_{distance} > P[j]_{distance}))$$
$$\text{donde } P[i]_{rank} = f \text{ si } P[i] \in \mathcal{F}^f .$$

Esto es, se define un orden lexicográfico con dos objetivos, con la posición en la clasificación por nodominancia como el de mayor importancia. Entonces, entre dos soluciones con diferente posición en la clasificación por nodominancia se prefiere aquella con la clasificación más baja. De otra forma, si ambas soluciones están localizados en el mismo frente, se prefiere la solución que está ubicada en una región con un menor número de puntos.

Presentadas todas las partes que conforman el NSGA-II, en el Algoritmo 11 muestra el bloque principal del mismo. Inicialmente se genera una población padre $P(0)$ de tamaño N. Esta población se clasifica en base a la nodominancia utilizando el Algoritmo 9. Terminado el proceso de clasificación, se asigna a cada solución un valor de inadaptabilidad igual a su nivel de nodominancia. Luego, se utilizan los operadores de selección por torneo binario, cruzamiento y mutación para generar, a partir de $P(0)$, una nueva población hijo $Q(0)$ también de tamaño N. Obtenidas las poblaciones padre e hijo iniciales, se realizan las sucesivas generaciones mientras la condición de parada no se cumpla.

En cada generación, se forma una población combinada $R(t) = P(t) \cup Q(t)$. Esto permite que las soluciones padres sean comparadas con la población hijo, asegurando el elitismo. Luego, la población $R(t)$, de tamaño $2N$, se clasifica de acuerdo a la nodominancia. Para ello se utiliza de nuevo el algoritmo de ordenamiento rápido por nodominancia, con el cual se identifican los diferentes frentes nodominados existentes en $R(t)$. Una vez que se obtienen todos los frentes, la nueva población padre $P(t+1)$ se forma agregando soluciones del primer frente $\mathcal{F}^1$, continuando con los demás frentes hasta que el tamaño exceda o sea igual a N. Luego, la población $P(t+1)$ se ordena de acuerdo al operador de *crowding* y se forma la población $P(t+1)$ con los primeros N elementos. Los individuos de cada frente se utilizan para calcular la distancia entre las soluciones vecinas (distancia de *crowding*) utilizando el procedimiento **crowding_distance_assignment** (Algoritmo 10). Las soluciones del último frente aceptado es ordenado de acuerdo a un criterio de comparación de *crowding* y se toman los elementos de este frente hasta completar el total de N soluciones en P. Finalmente, los elementos en $P(t+1)$ son utilizados para crear una nueva población $Q(t+1)$ utilizando selección, cruzamiento y mutación. A diferencia

Algoritmo 11: Algoritmo del NSGA-II.

```
1  t = 0;
2  Generar una población P(t) de tamaño N en forma aleatoria;
3  Utilizar el Algoritmo 9 para obtener una lista F con los frentes de P(t);
4  Asignar a cada elemento de P(t) un valor de inadaptabilidad igual a su nivel de
     nodominancia;
5  Utilizar torneo binario para seleccionar elementos de P(t) de acuerdo con su
     inadaptabilidad;
6  Efectuar cruzamiento y mutación para producir una población hijo Q(t) de tamaño N;
7  mientras El criterio de parada no se cumpla hacer
8      f = 1 ;                          // hacer la cuenta de frentes igual a 1
9      R(t) = P(t) ∪ Q(t) ;               // combinar la población padre e hijo
10     F =fast_non_dominated_sort(R(t)) ;     // Usar Alg. 9 para obtener los
         frentes de R(t)
       // Repetir hasta llenar la población padre
11     mientras ||P(t+1)||_c < N hacer
           // Calcular la distancia de crowding en F^f (Alg. 10)
12         crowding_distance_assignment(F^f)
13         P(t+1) = P(t+1) ∪ F^f ;       // incluir el f-ésimo frente en la
             población padre
14         f = f + 1
15     Ordenar de forma descendente de acuerdo al operador ≥_n
16     Tomar los primeros N elementos de P(t+1)
17     Seleccionar individuos de P(t+1) utilizando torneo binario de acuerdo al operador
         de crowding
18     Aplicar cruzamiento y mutación sobre los individuos seleccionados para obtener
         Q(t+1)
19     t = t + 1
```

del procedimiento de selección por torneo binario usual, se utiliza un procedimiento que considera el operador de *crowding* para la competencia entre individuos. El procedimiento se repite hasta que el número máximo de generaciones u otro criterio de parada sea alcanzado.

Capítulo 4
Algoritmos evolutivos multi-objetivo para problemas *many-objective*

En años recientes se ha puesto mucho interés en mejorar el desempeño de los MOEAs en problemas de optimización con muchos objetivos. Varios algoritmos han sido propuestos para lidiar con este tipo de problemas como lo describen [LLTY15, vLBB14, OEBY20, vLBB19, BES17, IS19]. Una taxonomía de los métodos reportados en la literatura considerando los conceptos claves que los inspiran se presenta en [vLBB14] como se muetra en la Figura 4.1. En dicha clasificación, los distintos métodos se clasifican primeramente en dos grupos:

1. métodos que utilizan relaciones de preferencia alternativas, y
2. métodos que transforman el problema *many-objective* original en uno relacionado.

Los métodos que utilizan relaciones de preferencia se basan en la idea que los tomadores de decisión, por lo general, están interesados solo en una parte del Frente Pareto. Los DM pueden entonces definir cierto tipo de características de las soluciones que son de mayor interés a fin de que los MOEAs puedan realizar la búsqueda concentrados en la región de interés del tomador de decisiones. Entonces, los métodos que usan relaciones de preferencia comparan las soluciones, un objetivo por vez, a fin de obtener una mejor clasificación de las soluciones y guiar la búsqueda utilizando información adicional tal como el número de objetivos para los cuales una solución es mejor que otra [DDB01, FA02], la magnitud de la mejora [FA02, SDD07], o el número de subespacios en los que una solución permanece como nodominada [dPKS07]. Además, existen trabajos que han propuesto relaciones que se basan en la utilización de funciones de pertenencia difusas para representar preferencias [FA02, KVGN05]. Por tanto, los métodos basados en relaciones de preferencia pueden a su vez ser clasificados como nítidos (*crisp*) o difusos (*fuzzy*).

Los diferentes métodos basados en relaciones de preferencia pueden requerir diferentes niveles de conocimiento de los problemas a resolver. Además de la utilización de relaciones de preferencia, existen otras formas de articulación de preferencia que han sido aplicadas para la optimización con muchos objetivos de manera reciente [XXL19].

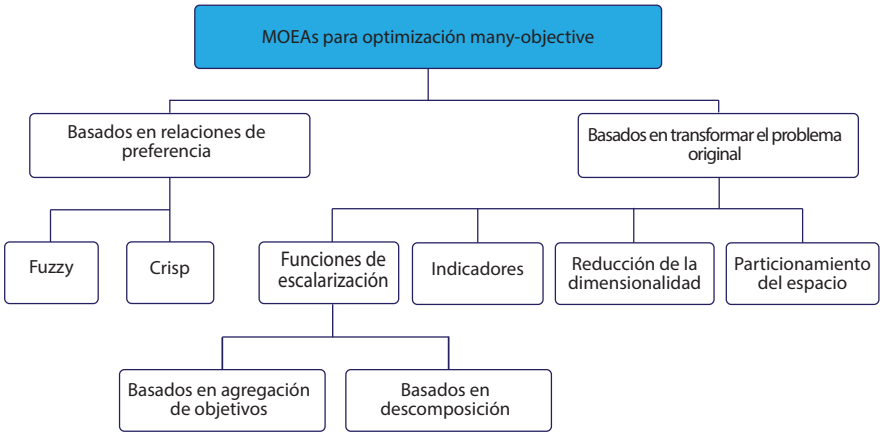

Fig. 4.1 Una taxonomía de los MOEA para optimización *many-objective* [vLBB14]

En el caso de los métodos que se basan en realizar una transformación del problema original, estos pueden clasificarse en:

1. basados en funciones de escalarización,
2. basados en indicadores,
3. basados en técnicas de reducción de la dimensionalidad, y
4. basados en el particionamiento del espacio.

Entre los métodos que utilizan funciones de escalarización se pueden considerar dos tipos principales: los basados en descomposición y los basados en agregación de objetivos. Los métodos basados en agregación de objetivos combinan grupos de objetivos para transformar el problema original en uno con un número menor de objetivos [KEB+09, MT09]; por su parte, los métodos basados en descomposición dividen el problema original en una colección de funciones escalares a ser optimizadas simultáneamente [Hug05, Hug07, ZL07, LGZ14, DJ14a, CJOS16].

Los métodos basados en indicadores evalúan las soluciones utilizando una métrica escalar de desempeño dada (indicador) [BZ11, ZK04]; entonces, un MOEA basado en indicadores transforma el problema de optimización *many-objective* en el problema de optimizar un indicador. Finalmente, los enfoques basados en la partición del espacio trabajan considerando distintos subconjuntos de objetivos en diferentes iteraciones del proceso de optimización [AT09]; de esta forma, estos métodos transforman el problema original en un conjunto de subproblemas relacionados.

Siendo uno de los objetivos del presente trabajo discutir las características de los algoritmos evolutivos para problemas con muchos objetivos de una manera integrada, basados en [von16, vLBB14, vLBB19] se presenta una descripción y ejemplos de los diferentes tipos de MOEAs para optimización *many-objective* que resumidos en la Figura 4.1. Estas secciones que siguen presentan modificaciones y actualizaciones menores respecto a los trabajos originales. Sin embargo, puesto que entre los diferentes tipos de MOEAs para problemas MaOP el modelo de descomposición

basado en la utilización de vectores de peso o puntos de referencia, tales como el MOEA/D [ZL07], han recibido mayor atención en cuanto a la optimización multi-objetivo, estos algoritmos se presentan en extenso en el Capítulo 5. En la Sección 4.1 se presentan los algoritmos basados en preferencias: en la Subsección 4.1.1 las alternativas crisp, en la Subsección 4.1.2 se presentan las difusas, mientras que, en la Subsección 4.1.3 se presenta un resumen sobre la utilización de MOEAs basados en relaciones de preferencias para problemas de optimización *many-objective*. Por su parte la Sección 4.2 presenta algunos algoritmos basados en transformaciones del problema original: en la Subsección 4.2.1 los métodos basados en agregación, en la Subsección 4.2.2 los basados en reducir el número de objetivos del problema original, luego, en la Subsección 4.2.3 los algoritmos basados en indicadores, la Subsección 4.2.4 presenta los métodos basados en particionamiento del espacio objetivo. Finalmente, la Subsección 4.2.5 presenta un resumen sobre los MOEAs para problemas con muchos objetivos basados en transformaciones del problema original.

4.1 Algoritmos basados en relaciones de preferencia

4.1.1 Alternativas *crisp*

4.1.1.1 Relación de dominancia $(1-k)$

A fin de refinar la clasificación de soluciones en problemas con muchos objetivos, [FA02] presentaron la relación de dominancia $(1-k)$, en inglés $(1-k)$-*dominance relation*. Esta relación se basa en contar el número de objetivos para los cuales una solución $\mathbf{x}$ dada es mejor, igual o peor que otra solución $\mathbf{x}'$. La relación de dominancia $(1-k)$ se define como sigue:

Definición 4.1 *Dominancia* $(1-k)$: Sean $\mathbf{x}, \mathbf{x}' \in \mathcal{X}_f$, $0 \leq k \leq 1$, $\mathbf{y} = \mathbf{F}(\mathbf{x})$, $\mathbf{y}' = \mathbf{F}(\mathbf{x}')$, entonces se dice que $\mathbf{x}$ $(1-k)$-domina a $\mathbf{x}'$, denotado $\mathbf{x} \prec_{(1-k)} \mathbf{x}'$, si y solo si:

$$n_e(\mathbf{y}, \mathbf{y}') < m \quad \text{y} \quad n_b(\mathbf{y}, \mathbf{y}') \geq \frac{m - n_e(\mathbf{y}, \mathbf{y}')}{k+1} \tag{4.1}$$

donde:

$$n_b(\mathbf{y}, \mathbf{y}') = |\{y_i |\ y_i < y'_i, i \in [1, m]\}| \tag{4.2}$$

$$n_e(\mathbf{y}, \mathbf{y}') = |\{y_i |\ y_i = y'_i, i \in [1, m]\}| \tag{4.3}$$

□

La relación $(1-k)$-*dominance* sirve para proponer un nuevo concepto de óptimo, llamado optimalidad k (k-*optimality*), y su correspondiente conjunto k-óptimo [FA02]:

Definición 4.2 *Optimalidad k (k-optimality):* una solución $\mathbf{x}$ es k-óptima con respecto a $\Omega \subseteq \mathcal{X}_f$ si y solo si $\nexists \mathbf{x}' \in \Omega$ tal que $\mathbf{x}'$ $(1-k)$ domina a $\mathbf{x}$. El conjunto

k-óptimo y el frente k-óptimo se definen como el conjunto de soluciones k-óptimas en el dominio de definición y el espacio objetivo, respectivamente.

Cuando k es cero, la relación de dominancia $(1-k)$ es equivalente a la relación de dominancia (Definición 2.5). Diferentes valores de k, en general, proveen diferentes subconjuntos de soluciones Pareto óptimas correspondientes a diferentes grados de optimalidad. A fin de clasificar las soluciones, se puede utilizar la relación de dominancia $(1-k)$ en vez de la relación de dominancia en el proceso de ordenamiento por no-dominancia [Deb01]. El proceso de ordenamiento por no dominancia clasifica las soluciones en niveles de dominancia. Este proceso primero determina el conjunto de soluciones no dominadas en la población, el cual es el primer nivel o frente de no-dominancia, denotado por $\mathcal{PF}_1$, luego, los elementos en $\mathcal{PF}_1$ se excluyen de la población para obtener el siguiente frente de no-dominancia $\mathcal{PF}_2$, y así por delante; el procedimiento continúa de manera iterativa hasta que se clasifican todos los elementos de la población. El siguiente ejemplo ilustra el uso de la relación de dominancia $(1-k)$ para clasificar soluciones.

Tabla 4.1 Valores de $n_e(\mathbf{y}, \mathbf{y}')$ para el Ejemplo 4.1

$n_e(row, colum)$	**A**	**B**	**C**	**D**	**E**
A = (1, 4, 2, 3)	4	0	1	0	0
B = (4, 3, 4, 2)	0	4	0	0	0
C = (2, 4, 1, 7)	1	0	4	1	1
D = (2, 6, 5, 1)	0	0	1	4	0
E = (5, 2, 1, 6)	0	0	1	0	4

Tabla 4.2 Valores de $n_b(\mathbf{y}, \mathbf{y}')$ para el Ejemplo 4.1

$n_b(row, colum)$	**A**	**B**	**C**	**D**	**E**
A = (1, 4, 2, 3)	0	2	2	3	2
B = (4, 3, 4, 2)	2	0	2	2	2
C = (2, 4, 1, 7)	1	2	0	2	1
D = (2, 6, 5, 1)	1	2	1	0	2
E = (5, 2, 1, 6)	2	2	2	2	0

Tabla 4.3 Resultados de $(m-n_e(\mathbf{y}, \mathbf{y}')/(k+1)$ del Ejemplo 4.1 con $k = 0.5$

	A	**B**	**C**	**D**	**E**
A	0.00	2.67	2.00	2.67	2.67
B	2.67	0.00	2.67	2.67	2.67
C	2.00	2.67	0.00	2.00	2.00
D	2.67	2.67	2.00	0.00	2.67
E	2.67	2.67	2.00	2.67	0.00

Tabla 4.4 Tabla de dominancia 0.5 del Ejemplo 4.1

$\prec_{0.5}$	**A**	**B**	**C**	**D**	**E**
A	0	0	1	1	0
B	0	0	0	0	0
C	0	0	0	1	0
D	0	0	0	0	0
E	0	0	1	0	0

Ejemplo 4.1 Considere A, B, C, D, y E como soluciones de un problema multi-objetivo de minimización tal que $\mathbf{F}(A) = \mathbf{A} = (1, 4, 2, 3)$, $\mathbf{F}(B) = \mathbf{B} = (4, 3, 4, 2)$, $\mathbf{F}(C) = \mathbf{C} = (2, 4, 1, 7)$, $\mathbf{F}(D) = \mathbf{D} = (2, 6, 5, 1)$, y $\mathbf{F}(E) = \mathbf{E} = (5, 2, 1, 6)$. Si estas soluciones son clasificadas utilizando el ranking de no-dominancia [Deb01], todas ellas estarán en el primer rango, es decir todas ellas son no-dominadas; sin embargo, utilizando la relación de dominancia $(1-k)$, se puede tener una clasificación de soluciones diferente. Considerando $k = 0.5$, la Tabla 4.1, la Tabla 4.2, y la Tabla 4.3 muestran los valores correspondientes para $n_e(\mathbf{y}, \mathbf{y}')$, $n_b(\mathbf{y}, \mathbf{y}')$, y $(m-n_e(\mathbf{y}, \mathbf{y}')/(k+1)$, respectivamente. Estos valores sirven para evaluar la Definición 4.1 y determinar la relación de dominancia 0.5 entre las soluciones en este ejemplo, como se muestra en la Tabla 4.4. Entonces, de acuerdo a la Tabla 4.4 se tienen las siguientes relaciones

de dominancia 0.5: $A \prec_{0.5} C$, $A \prec_{0.5} D$, $C \prec_{0.5} D$, $E \prec_{0.5} C$. En este caso, A, B, y E son 0.5 no-dominadas por ninguna otra solución considerada en este ejemplo; por tanto, éstas reciben un nivel 1. Entre las soluciones restantes, puesto que $C \prec_{0.5} D$, C recibe un nivel 2, y D un nivel 3. □

A fin de validar sus definiciones, [FA02] utilizaron ejemplos discretos simples así como casos analíticos continuos mostrando la validez de sus definiciones y la manera en que éstas se ajustan a un razonamiento de sentido común.

4.1.1.2 Ranking por clases de satisfacibilidad basado en la relación *favour*

A partir de un algoritmo previamente propuesto en [DDB99], Drechsler et al. [DDB01] propusieron un algoritmo para resolver problemas con muchos objetivos. Este algoritmo utiliza la relación *favour* para comparar las soluciones unas con otras y un método llamado ordenamiento por clases de satisfacibilidad (*Satisfiability Class Ordering - SCO*) para clasificar las soluciones. La relación *favour* se define como sigue:

Definición 4.3 *Relación Favour:* dadas $\mathbf{x}$, $\mathbf{x}' \in \mathcal{X}_f$, se dice que $\mathbf{x}$ es favorable a (*favoured to*) $\mathbf{x}'$, denotado por $\mathbf{x} \prec_{favour} \mathbf{x}'$, si y solo si:

$$n_b(\mathbf{y}, \mathbf{y}') > n_b(\mathbf{y}', \mathbf{y}) \tag{4.4}$$

donde: n_b fue definido en la Ecuación (4.2).

□

Debido a que la relación *favour* es no transitiva, para establecer un ordenamiento parcial entre las soluciones, en [DDB01] se utiliza el procedimiento SCO basado en un método para determinar los componentes fuertemente conectados. El método SCO inicia obteniendo un grafo de relaciones de las soluciones a ser clasificadas. Para el caso del grafo de la relación *favour*, los nodos representan las soluciones en el espacio objetivo y se tienen arcos dirigidos entre todos los pares de soluciones donde una es favorable a otra. Si existen ciclos, los nodos se juntan en un único nodo. Utilizando el grafo de relaciones comprimido, un algoritmo para determinar componentes fuertemente conectados basado en búsqueda por profundidad [CLR90] establece un orden parcial entre los nodos. Finalmente, las soluciones que fueron agrupadas en el mismo nodo reciben la misma posición en el ranking. En el algoritmo introducido por [DDB99], en cada generación, las soluciones se clasifican utilizando SCO, luego, la mitad de la población correspondiente a los mejores individuos se copia a la siguiente iteración sin modificación mientras que la otra mitad se reemplaza por nuevos elementos.

El siguiente ejemplo muestra cómo el método de clasificación SCO utilizando la relación *favour* mejora la clasificación de las soluciones no-dominadas entre sí.

Ejemplo 4.2 Utilizando el mismo conjunto de vectores del Ejemplo 4.1 y sus correspondientes valores de n_b mostrados en la Tabla 4.2, la Tabla 4.2 muestra la matriz de la relación *favour* para las cinco soluciones consideradas. Las filas corresponden

a la parte izquierda de la relación, mientras que las columnas son para el lado derecho, entonces un valor de 1 en la intersección de la fila **A** y la columna **C** indica que **A** $\prec_{favour}$ **C**. Considerando la matriz de relaciones anterior, la Figura 4.3 muestra el grafo de la relación *favour* y la clasificación producida por el algoritmo SCO [CLR90]. En este ejemplo, utilizando SCO, las soluciones A, B, y E tienen la primera ubicación en el ranking, la solución C tiene la segunda posición, mientras la solución D se ubica en el tercer lugar.

Fig. 4.2 table
Relación *favour* [FA02] entre pares de soluciones

$\prec_{favour}$	**A**	**B**	**C**	**D**	**E**
A = (1, 4, 2, 3)	0	0	1	1	0
B = (4, 3, 4, 2)	0	0	0	0	0
C = (2, 4, 1, 7)	0	0	0	1	0
D = (2, 6, 5, 1)	0	0	0	0	0
E = (5, 2, 1, 6)	0	0	1	0	0

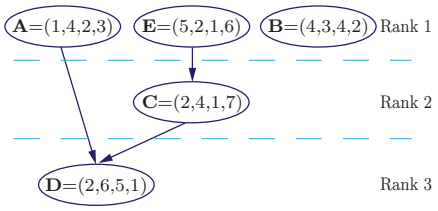

Fig. 4.3 figure
Grafo de la relación *favour* y ranking de soluciones SCO

Drechsler et al. [DDB01] probaron la clasificación utilizando SCO basado en la relación *favour* con dos instancias de un problema de aprendizaje heurístico para el diseño de un VLSI CAD con la necesidad de optimizar simultáneamente 6 y 7 objetivos. Los resultados, para instancias pequeñas, fueron comparados directamente con aquellos obtenidos por un método exacto, mostrando una calidad similar. Además, para las instancias donde no es posible computar los resultados exactos, el método basado en *favour* superó a un algoritmo evolutivo que utiliza una suma ponderada en un 50% de los casos de prueba.

4.1.1.3 Ranking por clases de satisfacibilidad basado en la relación $\epsilon-$preferred

A partir de la relación *favour* [DDB99, DDB01], en [SDD07] se define una nueva relación llamada ϵ*-preferred*. En esta relación, la comparación entre dos soluciones se basa en contar el número de objetivos en los cuales una solución es mejor que otra dentro de un umbral predefinido. En caso de empate, se utiliza la relación *favour* para establecer cuál es la mejor. Para construir la relación ϵ*-preferred*, se definen adicionalmente las siguientes relaciones:

Definición 4.4 *Relación* $\epsilon-$*exceed*: sean $\mathbf{x}, \mathbf{x}' \in \mathcal{X}_f$, $\mathbf{y} = \mathbf{F}(\mathbf{x})$, $\mathbf{y}' = \mathbf{F}(\mathbf{x}')$; para un vector de límites dado $\epsilon = (\epsilon_1, \ldots, \epsilon_m)$, se dice que $\mathbf{x}$ $\epsilon-$*exceed* $\mathbf{x}'$, denotado por $\mathbf{x} \prec_{\epsilon-exceed} \mathbf{x}'$ si y solo si:

$$|\{i \mid y_i < y'_i \wedge |y_i - y'_i| > \epsilon_i, i \in [1,m]\}| > |\{i \mid y'_i < y_i \wedge |y'_i - y_i| > \epsilon_i, i \in [1,m]\}| \quad (4.5)$$

□

Las definiciones 4.3 y 4.4 se combinan en la relación $\epsilon-$*preferred* como sigue [SDD07]:

Definición 4.5 $\epsilon-$*preferred relation*: dadas dos soluciones $\mathbf{x}, \mathbf{x}' \in \Omega$, se dice que $\mathbf{x}$ es $\epsilon-$*preferred* a $\mathbf{x}'$, denotado por $\mathbf{x} \prec_{\epsilon-pref} \mathbf{x}'$, ssi:

$$\mathbf{x} \prec_{\epsilon-exceed} \mathbf{x}' \vee (\mathbf{x}' \not\prec_{\epsilon-exceed} \mathbf{x} \wedge \mathbf{x} \prec_{favour} \mathbf{x}') \quad (4.6)$$

En [SDD07], la relación $\epsilon-$*preferred* remplaza la relación *favour* en el marco algorítmico utilizado en [DDB01]. El siguiente es un ejemplo del ranking de soluciones producido al utilizar la relación $\epsilon-$*preferred*.

Ejemplo 4.3 Considerando el mismo conjunto de soluciones utilizado en los ejemplos anteriores, la Tabla 4.5 muestra la relación $\epsilon-$*preferred* para un valor de $\epsilon_i = 1.5$ para todo $i \in [1,m]$, mientras la Figura 4.6 muestra el grafo de la relación $\epsilon-$*preferred* y las diferentes posiciones (*rank levels*). Como se puede notar, el procedimiento de ranking clasifica a *A* en el mejor lugar, a *E* en la segunda posición, *C* en el tercer puesto y finalmente, *B* y *D* reciben la cuarta ubicación.

Tabla 4.5 Relación ϵ*-preferred* entre pares de soluciones

$\prec_{\epsilon-pref}$	**A**	**B**	**C**	**D**	**E**
A = (1, 4, 2, 3)	0	1	1	1	1
B = (4, 3, 4, 2)	0	0	0	0	0
C = (2, 4, 1, 7)	0	1	0	1	0
D = (2, 6, 5, 1)	0	0	0	0	0
E = (5, 2, 1, 6)	0	0	1	0	0

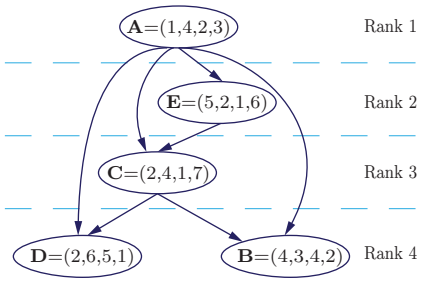

Tabla 4.6 Grafo de la relación ϵ*-preferred* y ranking SCO de las soluciones

En [SDD07] se compara el método de clasificación SCO utilizando $\epsilon-$*preferred* con el *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) [DPAM02], el método llamado *Dominates* [Kei01], y el método SCO con la relación *favour* [DDB01]. Como problema de prueba se utilizó un problema de diseño de agendas de enfermeras con 25 objetivos. De acuerdo a los resultados presentados, $\epsilon-$*preferred* produce soluciones con mejor calidad y se comporta de manera más robusta que las otras opciones utilizadas en la comparación.

4.1.1.4 Orden de preferencias (*Preference Ordering*)

A fin de mejorar la etapa de clasificación del NSGA-II [DPAM02], en [di 06, dPKS07] se presentan dos procedimientos basados en las definiciones de orden de eficiencia, y grado de eficiencia, denotados por PO_k y $PO_{k,z}$, respectivamente. La eficiencia de orden $\mathbf{k}$ y el orden de eficiencia se definen como sigue:

Definición 4.6 *Eficiencia de orden* $\mathbf{k}$ *y orden de eficiencia*: una solución $\mathbf{x}$ es eficiente de orden $\mathbf{k}$ si es Pareto óptima en los $\binom{m}{\mathbf{k}}$ subspacios del espacio objetivo tomando en cuenta solo $\mathbf{k}$ de los m objetivos por vez. El orden de eficiencia de una solución $\mathbf{x}$, denotado por $\mathsf{K}(\mathbf{x})$ es el valor mínimo de $\mathbf{k}$ para el cual $\mathbf{x}$ es eficiente. □

El procedimiento de ranking PO_k combina la Definición 4.6 con el procedimiento de ordenamiento por no-dominancia [Deb01] para clasificar las soluciones. En PO_k, una vez que todas las soluciones son clasificadas por no-dominancia, se calcula el orden de eficiencia para cada solución en $\mathcal{PF}_1$ y se obtiene el mínimo de tales valores, denotado K_{min}. Luego, el procedimiento PO_k asigna a cada solución $\mathbf{x}$ en $\mathcal{PF}_1$ un valor igual a:

$$R_{PO_k}(\mathbf{x}) = \mathsf{K}(\mathbf{x}) - \mathsf{K}_{min} + 1 \tag{4.7}$$

De acuerdo a la Ecuación (4.7), los mejores elementos en $\mathcal{PF}_1$ recibirán un valor de 1, mientras, los demás elementos valores mayores. Las soluciones que no se encuentren en $\mathcal{PF}_1$ obtienen un valor igual al peor valor asignado a las soluciones en el primer frente más su correspondiente nivel de dominancia. El conjunto de soluciones que reciben un valor de 1 por el procedimiento de ranking PO_k se etiqueta como $\mathcal{PF}_1^\star$.

El procedimiento $PO_{k,z}$ propuesto por [dPKS07] refina la clasificación PO_k para los casos donde el conjunto de soluciones en $\mathcal{PF}_1^\star$ con el mejor orden de eficiencia $\mathbf{k}^\star \neq m$ tiene más que un elemento, es decir, $|\mathcal{PF}_1^\star| > 1$. La estrategia de ranking $PO_{k,z}$ se basa en las siguiente definición:

Definición 4.7 *Eficiencia de orden* $\mathbf{k}$ *con grado* z: una solución $\mathbf{x}$ se dice eficiente de orden $\mathbf{k}$ con grado z si $\mathbf{x}$ es no-dominado por ningún miembro del conjunto Pareto en exactamente z de los $\binom{m}{\mathbf{k}}$ posibles subconjuntos de subespacios con $\mathbf{k}$ objetivos.

Una vez que PO_k se aplica, el procedimiento $PO_{k,z}$ usa la Definición 4.7 para computar el grado de eficiencia de orden $\mathbf{k}^\star - 1$ para todo $\mathbf{x} \in {\mathcal{PF}_1}^\star$. Entonces, el valor del ranking computado por $PO_{k,z}$ para los elementos $\mathbf{x} \in {\mathcal{PF}_1}^\star$ es:

$$R_{PO_{k,z}}(\mathbf{x}) = Z_{max}^{\mathbf{k}^\star - 1} - Z^{\mathbf{k}^\star - 1}(\mathbf{x}) + 1 \tag{4.8}$$

donde $Z^{\mathbf{k}^\star - 1}(\mathbf{x})$ es el grado de eficiencia de las soluciones $\mathbf{x}$ y $Z_{max}^{\mathbf{k}^\star - 1}$ es el máximo grado de eficiencia de orden $\mathbf{k}^\star - 1$. Finalmente, para aquellas soluciones que no se encuentran en ${\mathcal{PF}_1}^\star$ su posición final en el ranking se obtiene agregando $Z_{max}^{\mathbf{k}^\star - 1}$ a la posición obtenida con PO_k.

Ejemplo 4.4 Considere los vectores objetivos utilizados en los ejemplos previos: $\mathbf{A} = (1, 4, 2, 3)$, $\mathbf{B} = (4, 3, 4, 2)$, $\mathbf{C} = (2, 4, 1, 7)$, $\mathbf{D} = (2, 6, 5, 1)$, y $\mathbf{E} = (5, 2, 1, 6)$,

estos son eficientes de orden 4 puesto que todos ellos son no-dominados. Tomando en cuenta tres objetivos por vez, existen $\binom{4}{3} = 4$ subespacios posibles. La Tabla 4.7 muestra las relaciones de dominancia que se cumplen en los distintos subespacios con tres objetivos. Las soluciones C y D no son eficientes de orden 3 puesto que son dominadas en al menos un subespacio de 3 objetivos, mientras que las soluciones A, B, y E son eficientes de orden 3 puesto que no son dominadas por ningún vector para todas las combinaciones de tres objetivos. Sin embargo, considerando 2 objetivos por vez, A, B, y E son dominados en al menos un subespacio con 2 objetivos. Por tanto, los valores de $\mathsf{K}(A)$, $K(B)$, y $K(E)$ son iguales a 3, mientras $K(C)$, y $K(D)$ son iguales a 4 y el mínimo orden de eficiencia para las soluciones consideradas es $\mathbf{k}^\star = 3$. Usando la Ecuación (4.7), el procedimiento PO_k asigna una posición en el ranking de 1 para A, B, y E, mientras, C y D están en la posición 2.

Para mejorar el ranking de soluciones, el procedimiento $PO_{k,z}$ calcula el grado de eficiencia de orden 2 para las soluciones con el mejor orden de eficiencia $\mathcal{PF}_1^\star = \{A, B, E\}$. Para determinar estos valores, en la Tabla 4.8 se muestran las relaciones de dominancia existentes en los subespacios de dos objetivos. Como se puede notar, **A** y **B** no son dominados por ninguna solución en 4 de los 6 subespacios, mientras que **E** no es dominado por ninguna otra solución en 5 de los 6 subespacios; entonces, $Z^2(A) = 4$, $Z^2(B) = 4$, $Z^2(E) = 5$. De esta forma, de acuerdo a la Ecuación (4.8), la clasificación asignada por el procedimiento $PO_{k,z}$ para E es 1, mientras que para A y B es 2. La clasificación de $PO_{k,z}$ para las soluciones C y D es 7. Este resultado se obtiene al sumar el máximo grado de eficiencia de orden 2, $Z^2_{max} = 5$, a los valores previos del ranking PO_k de las soluciones. □

Tabla 4.7 Soluciones dominadas en subespacios con 3 objetivos

Subespacio	Relaciones de dominancia
(f_1, f_2, f_3)	$A \prec D$
(f_1, f_2, f_4)	$E \prec C, A \prec C$
(f_1, f_3, f_4)	–
(f_2, f_3, f_4)	$E \prec C$

Tabla 4.8 Soluciones que dominan a otras en cada subespacio con 2 objetivos

Subspacio	**A**	**B**	**E**
(f_1, f_2)	–	–	–
(f_1, f_3)	–	$A \prec B$	–
(f_1, f_4)	–	–	$A, B \prec E$
(f_2, f_3)	$E \prec A$	$E \prec B$	–
(f_2, f_4)	$B \prec A$	–	–
(f_3, f_4)	–	–	–

Para verificar la viabilidad de la definición de orden de preferencia como alternativa a la dominancia Pareto, en [dPKS07] se remplaza el procedimiento de clasificación del algoritmo NSGA-II con PO_k y $PO_{k,z}$ para comparar los tres métodos utilizando las funciones de prueba DTLZ1, DTLZ2, DTLZ3 y DTLZ5 [DTLZ02], considerando 4, 5, 6, 7 y 8 objetivos. La evaluación experimental se realizó utilizando las métricas: *Generational distance* (GD) [Vel99], *Hypervolume* [ZT99a], *Diversity Metric* 1 y 2 (DM1 y DM2) [KYD03] y *Coverage* [ZDT00, ZTL+03]. Los resultados muestran que, de acuerdo a las métricas evaluadas, los métodos PO_k y $PO_{k,z}$ son capaces de conseguir una mejor convergencia al conjunto Pareto verdadero que el NSGA-II original para los problemas considerados; sin embargo, son menos efec-

tivos que el NSGA-II original para mantener una buena diversidad de soluciones sobre la superficie Pareto completa.

4.1.1.5 $-\epsilon$-DOM ranking

[KY07a] propusieron cuatro procedimientos para reemplazar el método de asignación de distancia de hacinamiento (*crowding*) del NSGA-II [DPAM02] para problemas de optimización *many-objective*. En el NSGA-II, la distancia de *crowding* sirve como una métrica de selección secundaria para promover la diversidad entre soluciones en diferentes frentes; sin embargo, como en un problema *many-objective* prácticamente todas las soluciones se clasifican en el mismo frente (el primero), la distancia de *crowding* se vuelve el principal criterio de selección que sirve de guía para la búsqueda evolutiva. Entre los procedimientos presentados en [KY07a], según una comparación experimental realizada en el mismo trabajo, el método llamado $-\epsilon$-DOM es el que provee el mejor compromiso para un conjunto dado de métricas y problemas; por tanto este es el que se presenta a continuación.

En [KY07a] se utiliza una función auxiliar, llamada $mepsd$, para definir la función de ranking $-\epsilon$-DOM. Tal función, recibe un par de soluciones $(\mathbf{x}, \mathbf{x}')$ y retorna el valor más pequeño, tal que, si se resta de $\mathbf{F}(\mathbf{x}')$ esto haga que $\mathbf{x}'$ domine débilmente a $\mathbf{x}$. El $-\epsilon$-DOM *rank* de la solución $\mathbf{x}$ se define formalmente como

Definición 4.8 $-\epsilon$-DOM *rank*: el $-\epsilon$-DOM *rank* de una solución $\mathbf{x}$ respecto a una población P es:

$$-\epsilon\text{-DOM}(\mathbf{x}) = \min_{\mathbf{x}' \in P}\{mepsd(\mathbf{F}(\mathbf{x}), \mathbf{F}(\mathbf{x}'))\}, \tag{4.9}$$

donde,

$$mepsd(\mathbf{F}(\mathbf{x}), \mathbf{F}(\mathbf{x}')) = \begin{cases} 0 \;\; si \;\; f_i(\mathbf{x}') \leq f_i(\mathbf{x}) \; \forall i \in [1, \ldots, m] \\ \max_{i=1}^{m}\{f_i(\mathbf{x}') - f_i(\mathbf{x}) : f_i(\mathbf{x}) < f_i(\mathbf{x}')\} \end{cases} \tag{4.10}$$

Como se puede ver en la Definición 4.8, un valor alto de $-\epsilon$-DOM para una solución $\mathbf{x} \in P$ indica un mayor esfuerzo para que otra solución en P domine a $\mathbf{x}$.

Tabla 4.9 Valores de $mepsd$ y $-\epsilon$-DOM para las soluciones en el Ejemplo 4.5

	Valores de $mepsd$ para los pares					$-\epsilon$-DOM	
	(1, 4, 2, 3)	(4,3,4,2)	(2,4,1,7)	(2,6,5,1)	(5,2,1,6)	Valor	Orden
A=(1,4,2,3)	-	3	4	3	3	3	1°
B=(4,3,4,2)	1	-	5	3	4	1	3°
C=(2,4,1,7)	1	3	-	4	3	1	3°
D=(2,6,5,1)	2	2	6	-	5	2	2°
E=(5,2,1,6)	2	3	2	4	-	2	2°

Ejemplo 4.5 Para el mismo conjunto de soluciones de los ejemplos previos, la Tabla 4.9 muestra los valores de $mepsd$ para los distintos pares de soluciones, y el valor de ranking $-\epsilon$-DOM. En dicha tabla, el valor $mepsd$ correspondiente a la fila $\mathbf{A} = (1, 4, 2, 3)$ y la columna $\mathbf{B} = (4, 3, 3, 2)$, indica que 3 es el valor más pequeño que debe sustraerse de cada elemento de $\mathbf{B}$ para que B domine a A; ya que $mepsd(\mathbf{A}, \mathbf{B}) = \max\{4 - 1, 0, 4 - 2, 0\} = 3$, entonces, si $\mathbf{B} = \mathbf{B} - (3, 3, 3, 3)$, se tendrá que $B \prec A$. Las dos últimas columnas en la Tabla 4.9 indican los valores de $-\epsilon-$DOM para las soluciones del ejemplo actual (valor $mepsd$ mínimo) y el ranking correspondiente. En este caso $-\epsilon$-DOM(A)=3, $-\epsilon$-DOM(B)=1, $-\epsilon$-DOM(C)=1, $-\epsilon$-DOM(D)=2, $-\epsilon$-DOM(E)=2. El mejor (mayor) valor de $-\epsilon$-DOM es para la solución A la cual, por tanto, está en la posición 1; la ubicación 2 es para las soluciones D y E, mientras que finalmente la posición 3 es para las soluciones B y C.

Las cuatro alternativas a la distancia de *crowding* del NSGA-II propuestas en [KY07a] fueron evaluadas en el mismo trabajo utilizando DTLZ2, DTLZ3, DTLZ6 [DTLZ02], y el problema Pareto-Box [KVGN05] con 2, 8, y 15 objetivos. Para las comparaciones, se utilizaron las métricas *Convergence* [DMM03, KVGN05] y *Coverage* [ZDT00, ZTL+03]. Los experimentos en [KVGN05] concluyen que los métodos propuestos proveen una mejor convergencia que el procedimiento de *crowding* estándar, y que el $-\epsilon-$DOM también es capaz de proveer una buena cobertura del frente Pareto. Además, [KY07a] señala que puede ser beneficioso utilizar diferentes procedimientos de ranking conforme la corrida de optimización se ejecuta.

4.1.1.6 Relación de expansión de dominancia

A fin de inducir un ranking apropiado de soluciones, en [SAT07] se propone un método que expande y contrae el área de dominancia alterando la ubicación de cada solución en el espacio objetivo. Utilizando un parámetro definido por el usuario $S = [S_1, ..., S_m]$, la relación de expansión de dominancia (*Expansion dominance relation*) puede definirse como sigue:

Definición 4.9 *Relación de expansión de dominancia:* una solución $\mathbf{x}$ domina a otra solución $\mathbf{x}'$ bajo la relación de expansión si y solo si:

$$\forall i : f_i'(\mathbf{x}) \leq f_i'(\mathbf{x}') \quad \wedge \quad \exists i \quad \text{tal que} \quad f_i'(\mathbf{x}) < f_i'(\mathbf{x}') \tag{4.11}$$

donde:

$$f_i'(\mathbf{x}) = \frac{norm(F(\mathbf{x})) \cdot sin(\omega_i + S_i \cdot \pi)}{sin(S_i \cdot \pi)} \quad \forall \; i \in [1, m] \tag{4.12}$$

$norm(F(\mathbf{x}))$ es una norma de $\mathbf{F}(\mathbf{x})$, ω_i es el ángulo entre $\mathbf{F}(\mathbf{x})$ y un vector con $f_i(\mathbf{x})$ como el único componente distinto de cero. □

Si $S_i = 0.5$, la relación de expansión es equivalente a la dominancia. Si $S_i < 0.5$, entonces, $f_i'(\mathbf{x}) < f_i(\mathbf{x})$ y el área de dominancia se reduce. Por otra parte, si $S_i > 0.5$,

el área de dominancia se incrementa. La expansión en el área de dominancia produce una clasificación más fina de las soluciones, lo cual puede mejorar la selección.

Cuando todas las soluciones en el espacio objetivo se mueven a su nueva ubicación, utilizando la Definición 4.9, el ranking de dominancia Pareto [Deb01], brevemente descrito en la Subsección 4.1.1.1 puede utilizarse para clasificar las soluciones utilizando estos nuevos valores de los objetivos. Un ejemplo de este método se muestra a continuación.

Ejemplo 4.6 Considere otra vez el mismo conjunto de soluciones que en los ejemplos previos. Utilizando $S_i = 0.25$ para $i \in [1, m]$ la Tabla 4.10 muestra los valores $\mathbf{F}'$ correspondientes y el nuevo ranking de soluciones. Estos valores de S_i corresponden al valor máximo de expansión del área de dominancia [SAT07]. Entonces, utilizando la norma euclidiana ($||\cdot||$), para obtener el primer componente de $\mathbf{F}'(A)$, primeramente se calcula el valor de ω_i:

$$\omega_i = arcos\left(\frac{(1,4,2,3)\cdot(1,0,0,0)}{||1,4,2,3||\cdot||1,0,0,0||}\right) = arcos\left(\frac{1}{5.477\cdot 1}\right) = 1.39$$

Luego, usando la Ecuación 4.12 obtemos un valor de 6.39. Cuando se calculan todos los valores $\mathbf{F}'(x)$, estos valores se usan para determinar el ranking de soluciones utilizando el método de no-dominancia. En este caso el mejor valor es para la solución A, el cual es no-dominado, en la segunda posición están las soluciones B, C y E las cuales son dominadas por A, finalmente, el último lugar es para la solución D, dominada por B y D.

Tabla 4.10 Valores de $\mathbf{F}'$ y ranking para el Ejemplo 4.6 de la relación expansión

Solución	$\mathbf{F}(x)$	$\mathbf{F}'(x)$	Orden
A	$(1, 4, 2, 3)$	$(6.39,\ 7.74,\ 7.10,\ 7.58)$	1°
B	$(4, 3, 4, 2)$	$(9.39,\ 9.00,\ 9.39,\ 8.40)$	2°
C	$(2, 4, 1, 7)$	$(10.12,\ 11.35,\ 9.31,\ 11.58)$	2°
D	$(2, 6, 5, 1)$	$(9.87,\ 11.48,\ 11.40,\ 9.06)$	3°
E	$(5, 2, 1, 6)$	$(11.40,\ 9.87,\ 9.06,\ 11.48)$	2°

En [BKS01], se presenta el *Guided Multi-objective Evolutionary Algorithm* (G-MOEA) el cual, al igual que el método propuesto por [SAT07], se basa en modificar la región dominada de cada solución. Sin embargo, en el G-MOEA el tomador de decisiones tiene que especificar los valores de compromiso para cada par de objetivos para definir un conjunto de funciones objetivo lineales auxiliares que se utilizan para producir la modificación de la dominancia. Entonces, si bien ambos métodos tienen una idea básica similar, el número de relaciones de compromiso que el usuario debe especificar resulta un impedimento severo para escalar el G-MOEA a un gran número de objetivos en problemas *many-objective*

En [SAT07] usaron el problema de la mochila (*Knapsack problem*) [ZT99a] con 2, 3, 4 y 5 objetivos. Los resultados de ambos métodos fueron evaluados considerando las siguientes métricas: *Spread* [Deb01], *Hypervolume* [ZT99a] e *Inverse Genera-*

tional Distance (IGD) [Vel99]. Los autores concluyeron que la relación de expansión puede ser útil en combinación con otros métodos puesto que la relación puede mejorar ya sea la convergencia o la diversidad pero no ambos de manera simultánea. En un trabajo más reciente [LJCC09], el NSGA-II modificado con la relación de expansión ha mostrado un desempeño superior en la obtención de soluciones en la región *knee* de los problemas DTLZ [DTLZ02] cuando se comparan con otras relaciones tales como el ranking promedio [BW97], PO_k [dPKS07], y la relación *favour* [DDB99].

4.1.2 Alternativas difusas

4.1.2.1 Relación de dominancia $(1 - k_F)$

A fin de extender la relación de dominancia $(1 - k)$ (Definición 4.1) al dominio difuso, en [FA02] se propone *fuzzificar* la cuenta de los objetivos que pueden ser considerados mejores, iguales, o peores entre dos soluciones utilizando funciones de membresía μ_b^i, μ_e^i, y μ_w^i para cada función objetivo i. Entonces, la relación de dominancia $(1 - k_F)$ ($(1 - k_F)$-*dominance*) se define como sigue:

Definición 4.10 $(1 - k_F)$-*dominance*: Sean $\mathbf{x}, \mathbf{x}' \in \mathcal{X}_f$, $0 \leq k_F \leq 1$, $\mathbf{y} = \mathbf{F}(\mathbf{x})$, $\mathbf{y}' = \mathbf{F}(\mathbf{x}')$, entonces $\mathbf{x}$ $(1 - k_F)$-domina a $\mathbf{x}'$ si y solo si:

$$n_e^F(\mathbf{y}, \mathbf{y}') < m \text{ y } n_b^F(\mathbf{y}, \mathbf{y}') \geq \frac{m - n_e^F(\mathbf{y}, \mathbf{y}')}{k_F + 1} \tag{4.13}$$

donde:

$$n_b^F(\mathbf{y}, \mathbf{y}') = \sum_{i=1}^{m} \mu_b^i(y_i - y_i') \tag{4.14}$$

$$n_e^F(\mathbf{y}, \mathbf{y}') = \sum_{i=1}^{m} \mu_e^i(y_i - y_i')) \tag{4.15}$$

$$n_w^F(\mathbf{y}, \mathbf{y}') = \sum_{i=1}^{m} \mu_w^i(y_i - y_i')) \tag{4.16}$$

$$n_b^F + n_e^F + n_w^F = m \tag{4.17}$$

□

Dependiendo de la función de membresía seleccionada, puede ser necesario proveer parámetros adicionales. Por ejemplo, para la función de membresía trapezoidal [FA02], el tomador de deciciones debe proveer, para cada objetivo i, una 4-tupla (a_i, b_i, c_i, d_i) que define la forma del trapezoide (ver Figura 4.4), es decir, el intervalo de valores para los cuales la diferencia entre valores objetivos puede ser considerada insignificante, o no.

Usando la Definición 4.10, la optimalidad k_F (k_F-*optimality*) se define como:

Definición 4.11 Optimalidad k_F: un vector $\mathbf{x}$ es k_F-óptimo si y solo si $\nexists\, \mathbf{x}' \in \Omega$ tal que $\mathbf{x}'$ $(1 - k_F)$-domine $\mathbf{x}$. El conjunto k_F-óptimo y el frente k_F-óptimo se definen como el conjunto de soluciones k_F-óptimo en el dominio de las funciones y su correspondiente imagen en el espacio objetivo, respectivamente.

Ejemplo 4.7 A fin de clasificar el mismo conjunto de soluciones que en los ejemplos previos utilizando la relación (k_F)-dominance, se considera para todos los objetivos

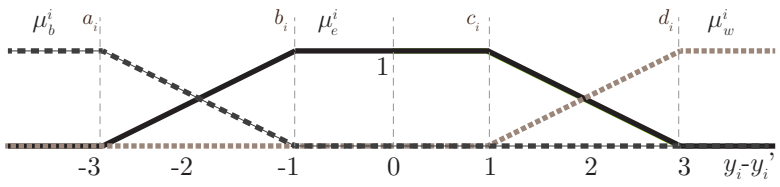

Fig. 4.4 Funciones de membresía μ_e^F, μ_b^F, μ_w^F para el Ejemplo 4.7

Tabla 4.11 Valores de $n_e(\mathbf{y}, \mathbf{y}')$ para el Ejemplo 4.7 de la relación $(1 - k_F)$-dominance

n_e^F	**A**	**B**	**C**	**D**	**E**
A	4	2.5	3	2	1.5
B	2.5	4	1.5	2.5	2
C	3	1.5	4	1.5	2.5
D	2	2.5	1.5	4	0
E	1.5	2	2.5	0	4

Tabla 4.12 Valores de $n_b^F(\mathbf{y}, \mathbf{y}')$ para el Ejemplo 4.7 de la relación $(1 - k_F)$-dominance

n_b^F	**A**	**B**	**C**	**D**	**E**
A	0	1.5	1	1.5	2
B	0	0	1	1	1
C	0	1.5	0	1.5	1
D	0.5	0.5	1	0	2
E	0.5	1	0.5	2	0

Tabla 4.13 Valores de $(m - n_e^F(\mathbf{y}, \mathbf{y}'))/(k_F + 1)$ para el Ejemplo 4.7, $k_F = 0.5$

	A	**B**	**C**	**D**	**E**
A	0	1	0.67	1.33	1.67
B	1	0	1.67	1	1.33
C	0.67	1.67	0	1.67	1
D	1.3	1	1.67	0	2.67
E	1.67	1.33	1	2.67	0

Tabla 4.14 Tabla de la relación 0.5_F-dominance para el Ejemplo 4.7

	A	**B**	**C**	**D**	**E**
A	0	1	1	1	1
B	0	0	0	1	0
C	0	0	0	0	1
D	0	0	0	0	0
E	0	0	0	0	0

reglas de membresía trapezoidal μ_b^i, μ_f^i y μ_w^i con parámetros $(a_i, b_i, c_i, d_i) = (-3, -1, 1, 3)$, como se muestra en la Figura 4.4. La Tabla 4.11 y la Tabla 4.12 muestran los valores de n_e^F y n_b^F, respectivamente, para todos los posibles pares de soluciones. Usando estos valores, la Tabla 4.13 muestra los resultados de $(m - n_e^F(\mathbf{y}, \mathbf{y}'))/(k_F + 1)$ para $k_F = 0.5$. Con los valores de dichas tablas, es posible aplicar la Definición 4.10 a cada par de soluciones y obtener la tabla de relación para 0.5_F-*dominance*, presentada en la Tabla 4.14. Luego, remplazando la relación de dominancia Pareto por la relación 0.5_F−*dominance* en el proceso de ordenamiento por no-dominancia [Deb01], la clasificación de las soluciones es: **A** en la posición 1, **B** y **C** en la posición 2, y los vectores **D** y **E** en la posición 3.

4.1.2.2 *Fuzzy-Dominance-Driven Genetic Algorithm*

En [KVGN05] se propuso el *Fuzzy-Dominance-Driven Genetic Algorithm* (FDD-GA), un algoritmo basado en una extensión difusa de la relación de dominancia Pareto (*Fuzzy Pareto dominance relation*) para calcular los grados de dominancia entre cada par de soluciones en la población. En [KVGN05] se establecen dos grados

de dominancia entre cada par de soluciones $(\mathbf{x}, \mathbf{x}')$ utilizando las funciones μ_a y μ_p, definidos como sigue:

Definición 4.12 *Relación de dominancia Pareto difusa*: sean $\mathbf{x}, \mathbf{x}' \in \mathcal{X}_f$, $\mathbf{y} = \mathbf{F}(\mathbf{x})$, $\mathbf{y}' = \mathbf{F}(\mathbf{x}')$; se dice que $\mathbf{x}$ domina a $\mathbf{x}'$ con grado μ_a, denotado por $\mathbf{x} \prec_{\mu_a} \mathbf{x}'$, donde

$$\mu_a(\mathbf{y}, \mathbf{y}') = \frac{\prod_{i=1, y_i, y_i' \neq 0}^{m} \min(y_i, y_i')}{\prod_{i=1, y_i \neq 0}^{m} y_i} \tag{4.18}$$

y que $\mathbf{x}$ es dominado por $\mathbf{x}'$ con grado μ_p, denotado $\mathbf{x} \prec_{\mu_p} \mathbf{x}'$, donde

$$\mu_p(\mathbf{y}, \mathbf{y}') = \frac{\prod_{i=1, y_i, y_i' \neq 0}^{m} \min(y_i, y_i')}{\prod_{i=1, y_i' \neq 0}^{m} y_i'} \tag{4.19}$$

En caso de tener un y_i o y_i' igual a 0, el índice correspondiente en el producto se excluye del numerador y del denominador. □

Note que si $\mathbf{x} \preceq \mathbf{x}'$, los elementos para multiplicar en el numerador provienen de $\mathbf{y}$ puesto que los elementos correspondientes en $\mathbf{y}'$ son mayores o igual a $\mathbf{y}$; por tanto, $\mu_a(\mathbf{y}, \mathbf{y}') = 1$ mientras $\mu_p(\mathbf{y}', \mathbf{y}) \leq 1$.

La clasificación difusa de una solución $\mathbf{x}$ dada en una población P es el grado máximo que tiene de ser dominado por cualquier otro elemento, cuanto más pequeño mejor [KVGN05]:

$$r_P(\mathbf{x}) = \max_{\mathbf{x}' \in \mathbf{P}, \{\mathbf{x} \neq \mathbf{x}'\}} \mu_p(\mathbf{F}(\mathbf{x}), \mathbf{F}(\mathbf{x}')) \tag{4.20}$$

Tabla 4.15 Valores de μ_p y r_P para las soluciones consideradas en el Ejemplo 4.8

μ_p	**A**	**B**	**C**	**D**	**E**	r_P	Orden
A = (1, 4, 2, 3)	–	0.125	0.214	0.133	0.100	0.214	1°
B = (4, 3, 4, 2)	0.500	–	0.214	0.400	0.267	0.500	4°
C = (2, 4, 1, 7)	0.500	0.125	–	0.133	0.400	0.500	4°
D = (2, 6, 5, 1)	0.333	0.250	0.143	–	0.067	0.333	2°
E = (5, 2, 1, 6)	0.250	0.167	0.429	0.067	–	0.429	3°

Ejemplo 4.8 La Tabla 4.15 presenta los valores de μ_p y r_P para el mismo conjunto de soluciones y objetivos considerados en los ejemplos previos. El valor 0.125 que corresponde a $\mu_p(\mathbf{F}(A), \mathbf{F}(B))$ se calcula dividiendo $1 \cdot 3 \cdot 2 \cdot 2$ entre $4 \cdot 3 \cdot 4 \cdot 2$. Como se puede notar, en este caso, A es el mejor, D está en la segunda posición, E en la tercera, mientras que las soluciones B y C están en la peor posición.

En [KVGN05], la utilidad del enfoque FDD-GA fue demostrado por medio de una comparación experimental con respecto al NSGA-II [DPAM02] utilizando el problema *Pareto-Box* con 20 objetivos.

4.1.3 Resumen sobre la utilización de MOEAs basados en relaciones de preferencias para problemas de optimización *many-objective*

Los métodos basados en las relaciones de preferencia consideran que la proporción creciente de soluciones no-dominadas constituye el reto principal para resolver problemas de optimización con cuatro objetivos o más. Por lo tanto, en general, las relaciones analizadas usan información adicional para proveer una clasificación de soluciones más fina. Las relaciones a ser utilizadas dependen del objetivo del proceso de optimización. Algunas de estas relaciones no poseen parámetros y pueden ser utilizadas para conseguir mejorar el conocimiento sobre el problema que está siendo atacado en las primeras etapas del proceso de optimización o implementadas en un método de optimización *a posteriori*, es decir, para seleccionar una solución luego de la obtención de la aproximación al conjunto Pareto [Mie99]. Sin embargo, relaciones como $\epsilon-$*preferred* [SDD07], *expansion dominance* [SAT07], y $(1-k_F)$-*dominance* [FA02] requieren que el tomador de decisiones defina parámetros más específicos para el problema de forma a guiar la búsqueda hacia la región de su interés; por lo tanto, estas relaciones se ajustan mejor a implementaciones con un enfoque interactivo. Estudiar cómo los diferentes valores de estos parámetros influyen en la búsqueda, puede servir para desarrollar técnicas que ajusten de manera automática sus respectivos parámetros conforme la evolución avanza, así como para desarrollar metodologías interactivas que permitan aplicar de mejor manera estos algoritmos a problemas *many-objective* de la vida real.

En esta sección, se ha ejemplificado cómo cada una de las relaciones de preferencia clasifica a un conjunto de cinco soluciones no-dominadas. Usando solo la relación de dominancia, todas estas soluciones son igualmente buenas; sin embargo, utilizando relaciones alternativas se puede discriminar entre éstas. A fin de visualizar la diferencia entre la optimalidad de Pareto y la inducida por las relaciones presentadas, la Figura 4.5 muestra las posiciones de las soluciones en los rankings de los ejemplos presentados. Como se puede notar, puesto que cada relación representa una preferencia distinta, una solución no-dominada dada puede ubicarse en una posición distinta para algunas de las relaciones consideradas. A su vez, al clasificar las soluciones de manera diferente, cada relación de preferencia posiblemente guíe el proceso evolutivo hacia una región diferente del espacio de búsqueda; por tanto, comparar los resultados de los métodos basados en diferentes preferencias requiere el desarrollo de metodologías que incorporen consideraciones adicionales [LJCC09].

Aún más, algunas de las relaciones proveen un ranking de soluciones más fino que otras. Cuando se incrementa la presión de selección se puede mejorar la convergencia; sin embargo, esto también puede llevar a una disminución en la diversidad de las soluciones y otros efectos no deseados. Analizar cómo las tendencias introducidas por las relaciones de preferencia afectan el compromiso entre la convergencia y la diversidad en diferentes tipos de problemas podría ser un tópico interesante para estudios futuros. Puede resultar particularmente interesante determinar las características de los problemas para los cuales una información adicional dada (como la cuenta del número de mejores objetivos) es capaz de orientar la búsqueda hacia soluciones en el conjunto Pareto global o si se quedan estancadas en un óptimo local.

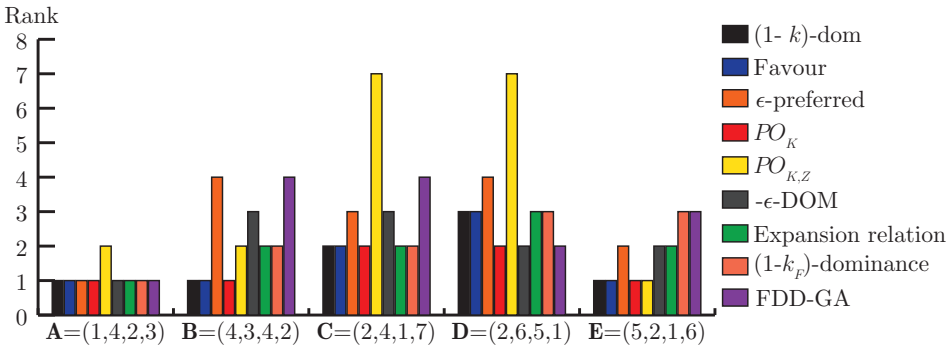

Fig. 4.5 Ranking de las soluciones de los ejemplos para las relaciones estudiadas

Una ventaja de la utilización de relaciones alternativas a la relación de dominancia Pareto es que estas son relativamente fáciles de implementar en el marco de MOEAs existentes basados en Pareto. En casos tales como las relaciones k-dominance, $(1-k_F)$-dominance [FA02] y la relación expansión [SAT07] estas pueden ser implementadas en el NSGA-II [DPAM02] con solo remplazar la dominancia Pareto con la nueva relación, mientras que otras partes del algoritmo permanecen sin modificaciones. Puesto que la relación *favour* [DDB01] y la relación ϵ-preferred [SDD07] son no transitivas, las mismas requieren que se reemplace el procedimiento de clasificación por no dominancia del NSGA-II por el SCO. Por último, los métodos basados en ordenamiento por preferencia [dPKS07] se añaden al método de ranking basado en Pareto original. En caso del FDD-GA [KVGN05], una vez que cada solución se compara con cada una de las otras, no existe necesidad de un procedimiento adicional de clasificación para obtener el valor que será asignado a una solución ya que este es el valor máximo obtenido en las comparaciones. Finalmente, implementar ϵ-DOM en el NSGA-II solo afecta el procedimiento de asignación de *crowding*. Si bien varios trabajos proponen modificaciones a MOEAs existentes, faltan estudios concluyentes que analicen cuáles son las características que hacen a un MOEA específico la mejor elección para incorporar un nuevo ranking, basados en una relación de preferencia. Además, puesto que diferentes relaciones pueden compartir el mismo marco de implementación, puede ser útil analizar cómo combinarlos en diferentes momentos del proceso evolutivo a fin de mejorar la búsqueda.

En general, los procedimientos para diferenciar y clasificar soluciones utilizando las relaciones estudiadas en esta sección pueden ser divididos en dos etapas: en la primera, las soluciones son comparadas unas con otras, un objetivo a la vez; en una segunda etapa, se utilizan los resultados de la comparación para producir un orden parcial de las soluciones. En la mayoría de los casos, la primera etapa tiene el mismo costo computacional que el procedimiento de ordenamiento rápido por no-dominancia [DPAM02], el cual es $O(m|P|^2)$, donde $|P|$ representa el tamaño de la población y m el número de objetivos; sin embargo, los métodos de ordenamiento por preferencia [dPKS07] requieren comparar las soluciones en diferentes subespacios

de objetivos hasta que las soluciones puedan diferenciarse; no existe una forma eficiente para calcular esta relación por lo que se vuelve no aplicable en problemas con un gran número de objetivos.

De acuerdo a [Deb01], conforme el tamaño de la población se incrementa, la proporción de soluciones no-dominadas decrece; por tanto, reducir la complejidad temporal de los métodos que clasifican soluciones puede ser un tópico interesante a fin de manejar poblaciones de mayor tamaño. Además, cabe destacar que, para algunas relaciones, es posible obtener un resultado de comparación contradictorio si son comparadas solo unas con otras o si son comparadas utilizando la posición en el ranking considerando toda la población. En la Tabla 4.15, por ejemplo, se puede ver que si el FDD-GA solo compara las soluciones C con E, y así C será considerada como la mejor, mientras que de acuerdo al ranking producido considerando las cinco soluciones E es mejor que C. Este comportamiento debe ser tenido en cuenta al utilizar la relación en los casos donde no se considera el conjunto completo de soluciones, por ejemplo durante la selección por torneo.

4.2 Algoritmos basados en transformaciones del problema original

4.2.1 Métodos basados en escalarización: basados en agregación

4.2.1.1 Agregación de objetivos utilizando índices de deseabilidad

En varios problemas, el tomador de decisiones puede agrupar los objetivos de un MOP en diferentes categorías. Para estos casos, Kruisselbrink et al. [KEB+09] han propuesto el uso de las llamadas funciones de deseabilidad (*desirability functions*) para combinar grupos de objetivos a fin de transformar el problema *many-objective* original a uno con un número moderado de objetivos. Básicamente, las funciones de deseabilidad pueden ser combinadas de acuerdo a categorías predefinidas para obtener un único valor para cada categoría, llamado su índice de deseabilidad. Entonces, el conjunto de estos índices forman el nuevo problema multi-objetivo a ser optimizado. Además, en [KEB+09] se explica cómo mapear problemas con un número pequeño de objetivos m' y un número c de restricciones *fuzzy* o débiles en un problema *many-objective* correspondiente con $m = m' + c$ objetivos, mapeando, tanto objetivos como restricciones, en funciones de deseabilidad.

Para probar el método de agregación de objetivos utilizando índices de deseabilidad, en [KEB+09] los autores implementaron una versión adaptada del algoritmo evolutivo presentado en [KBIvdH08] el cual utiliza el método de selección del NSGA-II [DPAM02] y un algoritmo genético mono-objetivo para resolver un problema de diseño automático de drogas donde se busca encontrar un receptor antagonista del estrógeno. El problema utilizado tiene 3 objetivos y seis restricciones que fueron mapeados a funciones de deseabilidad para obtener un problema con 9 obje-

tivos. Luego, se consideraron las siguientes alternativas de agregación: (i) funciones de deseabilidad para todos los objetivos y agregación de todas las restricciones modeladas por las funciones de deseabilidad; (ii) agregación de las funciones de deseabilidad correspondientes a las restricciones pero no aquellas correspondientes a los objetivos; y (iii) agregación de todas las funciones de deseabilidad en un único índice. El resultado obtenido por las ejecuciones de estas tres opciones de agregación se comparan utilizando un enfoque visual y una aproximación de Monte-Carlo para el cálculo de la métrica *Hypervolume*. En el trabajo de Kruisselbrink et al. [KEB+09] concluyen que el uso del índice de deseabilidad es una buena alternativa para reducir el número de objetivos en problemas *many-objective*. La parte negativa de utilizar índices de deseabilidad es que conducen a una distribución mucho menor de las soluciones obtenidas que la que puede obtenerse utilizando optimización basada en Pareto.

4.2.1.2 Agregación de objetivos utilizando correlación entre objetivos

Murata et al. [MT09] presentaron un método basado en suma ponderada (*weighted-sum*) para reducir un problema *many-objective* con m objetivos en un problema relacionado con m' objetivos que puede ser resuelto utilizando un MOEA. A diferencia de [KEB+09], en [MT09] los objetivos a ser combinados no se especifican por el tomador de decisiones sino que se determinan a partir de las correlaciones entre las funciones objetivo.

El método propuesto en [MT09] recibe como entrada el número de objetivos m' en los que se desea combinar los objetivos originales, y el número de generaciones cuando tal combinación se realizará. Entonces, luego de un número especificado de generaciones, se muestrea un conjunto de soluciones para recalcular con éstas las correlaciones entre los objetivos. Las correlaciones serán utilizadas para formar m' grupos de objetivos, tratando de maximizar la correlación promedio de los objetivos en cada grupo. Cuando se determinan los grupos de objetivos, cada grupo se utiliza para calcular, para cada solución, un valor de adaptación correspondiente al promedio de la suma de objetivos en el grupo correspondiente. De esta manera, el problema se transforma en uno con $m' < m$ funciones objetivo que serán utilizados por un MOEA para buscar soluciones.

Para evaluar experimentalmente su propuesta, en [MT09] se compararon los resultados de la versión original del NSGA-II, y los obtenidos por la versión modificada del mismo algoritmo con el método de agregación utilizando correlaciones entre objetivos considerando casos del problema de la mochila (*knapsack*) [ZT99a] con 10 y 40 objetivos. Los resultados de la comparación, utilizando como medidas de desempeño la suma normalizada y el rango de objetivos [ITN08], muestran que el NSGA-II con el enfoque de agregación propuesto puede encontrar mejores soluciones no-dominadas que la versión original del NSGA-II.

4.2.2 MOEAs basados en técnicas de reducción de la dimensionalidad

4.2.2.1 Método de reducción basado en análisis de componentes principales

Existen problemas que, aunque tienen muchos objetivos, tienen un frente Pareto de baja dimensión. Esto significa que, para dos soluciones tomadas de manera aleatoria, puede existir un conflicto entre dos objetivos dados y, sin embargo, los mismos objetivos pueden no estar en conflicto uno con otro cerca del frente Pareto. Deb y Saxena [DS05] combinaron el NSGA-II [DPAM02] con una técnica de reducción de la dimensionalidad basada en Análisis de Componentes Principales (*Principal Component Analysis* - PCA) para lidiar con aquellos problemas con muchos objetivos que tienen un frente Pareto de baja dimensión. En [DS05], el NSGA-II ejecuta un número dado de iteraciones obteniendo un frente Pareto aproximado considerando el conjunto inicial de objetivos $\mathcal{F}'_0 = \mathcal{F}$, luego, la salida intermedia (frente aproximado) se pasa al PCA para obtener un nuevo conjunto de objetivos $\mathcal{F}'_1 \subseteq \mathcal{F}'_0$ a ser utilizado en la siguiente iteración del NSGA-II.

El método de reducción basado en el PCA almacena los valores de los objetivos de la población en una matriz $\mathbf{O}$ de tamaño $m \times |P|$ donde cada fila $\mathbf{O}_i$ representa los valores de las funciones objetivo f_i, mientras que cada columna representa los valores objetivo de una solución. Los valores de $\mathbf{O}$ son utilizados para obtener la matriz estandarizada $\hat{\mathbf{O}}$, es decir un conjunto de datos que tiene cero como su centroide. La matriz estandarizada $\hat{\mathbf{O}}$ puede obtenerse restando la media de cada valor objetivo. Luego, la matriz $\hat{\mathbf{O}}$ se utiliza para calcular la matriz de covarianza $\hat{\mathbf{O}}_{covar}$ y la matriz de correlación $\hat{\mathbf{O}}_{corr}$.

Luego, se calculan los vectores propios (*eigenvectors*) y valores propios (*eigenvalues*) de $\hat{\mathbf{O}}_{corr}\hat{\mathbf{O}}^T_{corr}$. Los vectores propios se consideran como los componentes principales (*principal components* - PC). El vector propio con el mayor valor propio corresponde al primer PC, el segundo *eingenvalue* al segundo PC, y así por delante. El primer componente de un PC (un eigenvector dado) denota la contribución del primer objetivo en este vector, el segundo denota la contribución del segundo objetivo, etc. El elemento más negativo y más positivo de un componente principal dado correspondería a sus dos objetivos conflictivos más importantes. Entonces, [DS05] usan un umbral de corte predefinido para examinar los PC en un orden incremental hasta que la contribución acumulada de los PC analizados exceda el umbral de corte.

Para hacer viable el método basado en PCA, en [DS05] se propone un procedimiento adicional que selecciona, para el primer componente principal, aquellos objetivos que contribuyen de manera más positiva y más negativa. Luego, los otros componentes principales son considerados en orden. Los detalles de este método pueden encontrarse en [DS05].

Para validar el método basado en el PCA, Deb y Saxena [DS05] experimentaron con una versión modificada del problema DTLZ5 original [DTLZ02] que permite especificar la dimensionalidad deseada del frente Pareto. Los resultados experimentales mostraron que el PCA-NSGA-II puede ser utilizado efectivamente para resolver problemas redundantes y de grandes dimensiones. Además, el método PCA-NSGA-II usa una fracción del tiempo de computación del que necesita el NSGA-II

estándar para realizar una buena estimación del frente Pareto para algunos problemas de prueba. Sin embargo, el método ha demostrado dificultades para encontrar la combinación correcta de objetivos en problemas con un frente Pareto de grandes dimensiones.

En base a [DS05], en [SD07] se proponen dos nuevos algoritmos de reducción de la dimensionalidad no-lineales para optimización evolutiva multi-objetivo. Uno de estos métodos se basa en el concepto de "correntropia" PCA (*correntropy PCA*) [XPPP06], mientras que el otro implementa el principio de despliegue máximo de varianza (*maximum variance unfolding principle*) [WS06]. En [SD07] una comparación experimental demuestra que los nuevos enfoques propuestos superan al procedimiento básico basado en PCA tanto en términos de precisión como de tiempo de computación cuando se resuelven problemas DTLZ2 y DTLZ5 de hasta 50 objetivos.

4.2.2.2 Reducción voraz

En el trabajo de Brockhoff y Zitzler [BZ06b], se presenta una nueva noción de conflicto entre objetivos basado en la estructura de dominancia, de acuerdo a la cual dos conjuntos de objetivos $\mathcal{F}_1$ y $\mathcal{F}_2 \subseteq \mathcal{F}$ se consideran conflictivos si $\preceq_{\mathcal{F}_1} \neq \preceq_{\mathcal{F}_2}$. Una reducción en el número de objetivos puede cambiar la relación de dominancia, en este sentido existe un error. A fin de cuantificar tal error, [BZ06b] definieron la relación de conflicto δ (δ-*conflict relation*) entre objetivos, de la siguiente forma:

Definición 4.13 *Relación ϵ-dominance*: sea un conjunto de objetivos $\mathcal{F}' \subseteq \mathcal{F}$, la relación ϵ-dominance sobre $\mathcal{F}'$ se define como:

$$\preceq^{\epsilon}_{\mathcal{F}'} = \{(\mathbf{x}, \mathbf{x}') | \mathbf{x}, \mathbf{x}' \in \mathcal{X} \wedge \forall f_i \in \mathcal{F}',\ f_i(\mathbf{x}) - \epsilon \leq f_i(\mathbf{x}')\} \tag{4.21}$$

Definición 4.14 Sean $\mathcal{F}_1, \mathcal{F}_2 \subseteq \mathcal{F}$ dos conjuntos de objetivos, se dice que:

- $\mathcal{F}_1$ es δ-no conflictiva con $\mathcal{F}_2$ si y solo si $(\preceq_{\mathcal{F}_1} \subseteq \preceq^{\delta}_{\mathcal{F}_2}) \wedge (\preceq_{\mathcal{F}_2} \subseteq \preceq^{\delta}_{\mathcal{F}_1})$
- $\mathcal{F}_1$ es δ-conflictiva con $\mathcal{F}_2$ si y solo si ($\mathcal{F}_1$ es no δ-no conflictiva con $\mathcal{F}_2$) □

Un conjunto de objetivos δ-mínimo se define como un subconjunto de $\mathcal{F}$ que no puede ser de vuelta reducido sin cambiar la estructura de dominancia asociada con un error de como mucho δ. Un conjunto $\mathcal{F}$ de objetivos se llama δ-redundante si y solo si existe un conjunto $\mathcal{F}' \subseteq \mathcal{F}$ que es δ-mínimo con respecto a $\mathcal{F}$.

Luego, Brockhoff y Zitzler [BZ06b] propusieron un método exacto y un algoritmo voraz para resolver el problema de encontrar el mínimo conjunto de objetivos correspondientes a un error dado (δ-MOSS) así como el problema de encontrar un subconjunto de objetivos de tamaño $\mathbf{k}$ con el mínimo error posible (k-EMOSS). Como ambos problemas son $\mathcal{NP}$-difícil, el método exacto solo es aplicable a problemas de tamaño pequeño. El algoritmo voraz se basa en construir de manera iterativa un subconjunto de objetivos $\mathcal{F}'$ que es δ-no conflictivo con $\mathcal{F}$.

El método de reducción de objetivos propuesto en [BZ06b] fue comparado con los resultados del IBEA sobre 9 casos de un problema *Knapsack* [ZT99a] y casos del

DTLZ2, DTLZ5, y DTLZ7 [DTLZ05]. Un análisis experimental de los resultados demostró que, al menos en los problemas considerados, una reducción en el número de objetivos es posible sin alterar la estructura de dominancia. Además, el algoritmo exacto conduce a subconjuntos de objetivos más pequeños que el algoritmo voraz, mientras que los tiempos de ejecución son considerablemente menores para el algoritmo voraz. Finalmente, se demostró de manera experimental que el método basado en PCA [DS05] produce conjuntos de objetivos con mayor error en la estructura de dominancia que aquellos computados por el algoritmo propuesto en [BZ06b].

4.2.2.3 Reducción del conjunto de objetivos basados en la selección no supervisada de características

En [LJCCC08], la técnica de reducción originalmente desarrollada en [MMP02] fue integrada en un MOEA. La idea básica del método de reducción es similar al presentado en [DS05], es decir, las soluciones no-dominadas obtenidas por un MOEA se utilizan de manera iterativa para estimar una matriz de correlación indicando conflictos entre cada par de objetivos y determinando los objetivos más conflictivos de un problema a fin de reducir el conjunto de objetivos. Basados en la idea anteriormente señalada, en [LJCCC08] se presenta un algoritmo que encuentra el mínimo subconjunto de objetivos no redundantes con el mínimo error posible y otro algoritmo que obtiene el mínimo conjunto de k objetivos no redundantes que lleva al mínimo error. Estos dos algoritmos solo difieren en el criterio de parada y los parámetros de entrada. En un caso el número de objetivos deseados se especifica mientras que en el otro se calcula.

Luego de computar la matriz de correlación de un conjunto de soluciones no-dominadas, la técnica de reducción opera siguiendo los siguientes pasos básicos [LJCCC08]:

1. Usar el conflicto entre objetivos como distancia a fin de dividir el conjunto de objetivos en vecindades homogéneas de tamaño q en torno a cada objetivo.
2. Seleccionar el vecindario más compacto. Esto es, el vecindario con la distancia mínima a su q-ésimo vecino.
3. Mantener el centro de tal vecindad y descartar sus q vecinos (los objetivos con el menor conflicto en el conjunto actual). En este proceso, la distancia al q-ésimo vecino puede verse como un error cometido al remover los q objetivos vecinos.

De acuerdo al caso específico, el procedimiento itera los pasos 2 y 3 mientras el número de objetivos deseados no alcanza el valor k especificado o mientras que no hayan más vecinos para ser considerados.

Las técnicas de reducción del número de objetivos propuestas en [LJCCC08] fueron incorporadas en el NSGA-II [DPAM02] y comparadas con el método de reducción basado en PCA [DS05] y el método de reducción en [BSDZ08]. Para los experimentos se consideraron tres problemas: una variación del DTLZ5 con 3, 5 y 10 objetivos [DS05], una variación del DTLZ2 [BSDZ08] y el problema *Knapsack* [ZT99a] con 10 y 20 objetivos. Las métricas utilizadas para evaluar los resultados

fueron IGD [Vel99] y $\delta-$error [BZ06b]. Los experimentos resultantes muestran que los métodos resultan competitivos cuando son comparados a los algoritmos descritos en [DS05] y en [BSDZ08].

A fin de eliminar la necesidad de parámetros adicionales que deben proveerse al algoritmo presentado en [LJCCC08], en López et al. [LJCCAT11] proponen una técnica de reducción modificada.

4.2.3 MOEAs basados en indicadores

Los MOEAs basados en indicadores utilizan un indicador o métrica de calidad para asignar el valor de adaptación a las soluciones; así, estos algoritmos transforman el problema original con muchos objetivos en un problema con un único objetivo, el problema de optimizar el indicador escogido. Un indicador unario toma un conjunto de soluciones no dominadas y retorna un número real asociado con un criterio de desempeño dado; mientras que, los indicadores de calidad binarios sirven para comparar la calidad relativa de dos conjuntos de soluciones no dominadas.

En [ZK04], se propuso el *Indicator-based Evolutionary Algorithm* (IBEA) como un marco general o *framework* para incorporar indicadores en la búsqueda evolutiva. En dicho *framework*, las soluciones se comparan de a pares utilizando un indicador binario arbitrario; sin embargo, se requiere que estos indicadores preserven la dominancia, es decir, una solución no será evaluada mejor que otra que la domina [ZK04].

En IBEA, la selección para el cruzamiento se realiza utilizando torneo binario; además, un procedimiento de selección ambiental remueve de manera iterativa los peores individuos de la población y actualiza los valores de adaptación de los individuos restantes. En [ZK04], los autores propusieron dos variantes del IBEA basadas en los siguientes indicadores: el indicador aditivo ϵ (*additive ϵ-indicator* - I_{ϵ^+}) [ZTL+03], y el indicador I_{HD}, el cual se basa en el concepto de Hipervolumen [ZTL+03]. Estos indicadores pueden ser computados en tiempo lineal respecto a la dimensión del espacio objetivo cuando se comparan dos soluciones por vez. Ambas variantes del IBEA fueron comparadas una contra otra en varios problemas de prueba con muchos objetivos [BZ11, SAT10, SAT12, HR12, WBN07]. Entre estos trabajos, vale destacar los resultados de [WBN07], donde ambas variantes del IBEA convergieron en un frente Pareto para el problema DTLZ2 con 6 objetivos [DTLZ05]; mientras solo el IBEA con I_{ϵ^+} fue capaz de converger en el caso del problema DTLZ1 utilizando el mismo número de objetivos. Sin embargo, ambas variantes fallaron en producir una distribución adecuada de soluciones para DTLZ1. Cuando Sato et al. [SAT10] utilizaron el IBEA para el problema de la mochila, también encontraron que estos consiguen una buena convergencia pero con una baja diversidad.

Emmerich et al. [EBN05] desarrollaron el *$\mathcal{S}$-metric Selection-EMOA* (SMS-EMOA), un algoritmo evolutivo de estado estacionario (*steady-state evolutionary algorithm*) que intenta maximizar el Hipervolumen [ZTL+03]. En este caso, solo se

genera un nuevo individuo en cada iteración por medio de operadores de variación aleatorios. En el SMS-EMOA, el procedimiento de ordenamiento por no-dominancia sirve como un criterio de selección primario, mientras que el Hipervolumen se utiliza como un criterio de selección secundario aplicado al último frente. Si al remplazar un miembro de la población por el nuevo individuo generado se incrementa el valor del Hipervolumen cubierto por la población, el nuevo individuo es incorporado a la población. El SMS-EMOA ha mostrado buenos resultados en problemas con dos y tres objetivos [EBN05, NBE05]; sin embargo, puesto que los algoritmos que calculan de manera exacta el Hipervolumen para un conjunto de soluciones tienen tiempo de ejecución exponencial con respecto al número de objetivos, la utilización de la métrica Hipervolumen es una dificultad severa para la aplicación del SMS-EMOA en problemas con muchos objetivos.

Basseur y Zitzler [BZ06a] propusieron varias técnicas para integrar manejo de incertidumbre en el IBEA con el indicador I_{ϵ^+}. En [BZ06a] se consideran casos donde un proceso estocástico determina el valor de las funciones objetivo; entonces, cada solución se asocia con una distribución de probabilidad sobre el espacio objetivo. Para estos casos, desarrollaron un método que computa el valor exacto de $I_{\epsilon+}$, además, se proponen varios algoritmos para aproximar el valor de $I_{\epsilon+}$. Estos métodos fueron estudiados empíricamente en varios problemas, obteniendo resultados satisfactorios con un número creciente de objetivos.

La carga computacional del cálculo del Hipervolumen, en general, evita que este indicador se utilice en problemas con muchos objetivos. Es por ello que, Brockhoff y Zitzler [BZ07] propusieron el *Simple Indicator Based Evolutionary Algorithm* (SIBEA) como un *framework* para combinar estrategias de reducción del número de objetivos y búsqueda basada en Hipervolumen. Usando este *framework*, en [BZ07] se analizaron las técnicas de reducción de la dimensionalidad δ-MOSS y k-EMOSS propuestas en [BZ06b] para problemas con hasta 9 objetivos. Los experimentos realizados en [BZ07] demostraron que, considerando el mismo tiempo de ejecución, el SIBEA sin técnica de reducción puede ser mejorado con respecto a su convergencia al combinarlo con el k-EMOSS. Este resultado se explica debido a que la reducción de objetivos produce una reducción del tiempo de computación del Hipervolumen lo cual permite evaluar más soluciones en un mismo tiempo. La posible mejora que puede obtenerse gracias a la reducción de objetivos está limitada debido a la sensibilidad que tiene el cómputo del Hipervolumen con respecto al número de objetivos, además, para ser viable, los problemas deben ser reducidos a menos de 6 objetivos. Así mismo, mostraron que el SIBEA obtiene mejores valores de Hipervolumen que el NSGA-II [DPAM02] y el SPEA2 [ZLT01b].

A fin de lidiar con el esfuerzo computacional que requiere calcular el Hipervolumen, en vez de reducir el número de objetivos, Bader y Zitzler [BZ11] proponen el uso de un algoritmo Monte Carlo para aproximar el valor exacto del Hipervolumen. En base a esta idea, en [BZ11] se presentó el *Hypervolume Estimation Algorithm for Multi-objective Optimization* (Hype). Una característica importante del Hype es que permite establecer una relación de compromiso entre la precisión de la aproximación y los recursos computacionales. Hype es similar a otros algoritmos evolutivos pero utiliza el concepto de selección medioambiental para crear una nueva población a

partir de las mejores soluciones en el conjunto unión formado por el padre y las poblaciones hijo y estima el valor del Hipervolumen muestreando las soluciones en diferentes frentes. En [BZ11], se comparan dos variantes del Hype contra varios MOEA del estado del arte utilizando 17 problemas de prueba con hasta 50 objetivos. La calidad de los resultados de los métodos comparados fueron evaluados utilizando el cálculo exacto del Hipervolumen en problemas con menos de 6 objetivos, y, para un número mayor de objetivos, valores de Hipervolumen aproximados utilizando muestreo de Monte Carlo [BDZ10]. Para el conjunto de funciones de prueba y parámetros utilizados en los experimentos realizados en [BZ11], Hype mostró ser muy competitivo, consiguiendo el mejor valor de Hipervolumen para la mayoría de los casos.

4.2.4 MOEAs basados en particionamiento del espacio

En el ϵ *Ranking-Evolutionary Multi-objective Optimizer* (ϵR-EMO) presentado en [AT09] la idea central es alternar entre una iteración que considera el conjunto completo de objetivos, y un conjunto de iteraciones que toman en cuenta un subespacio diferente cada vez.

El ϵR-EMO fue desarrollado en base al NSGA-II [DPAM02]. Entonces, este empieza generando una población inicial vacía P y una población aleatoria Q. Luego, un procedimiento iterativo considera todos los objetivos del problema *many-objective* para clasificar las soluciones en $(P \cup Q)$ usando el procedimiento rápido de clasificación por no-dominancia. Las soluciones clasificadas se almacenan en una memoria adicional $\mathcal{M}$, y a estos elementos se le asigna su correspondiente distancia de *crowding*. Posteriormente, el espacio objetivo se particiona en un conjunto m_s de subespacios no solapados con la misma dimensión. Luego, las poblaciones originales P y Q se vacían. El ϵR-EMO continúa evolucionando $\mathcal{M}$ una generación para cada subespacio s utilizando el procedimiento ϵ-ranking [AT08]. Un conjunto truncado de soluciones en $\mathcal{M}$ se utiliza para obtener una nueva población P_s utilizada para generar una Q_s. Al final de cada iteración con subespacios, las soluciones en P_s y Q_s se unen en P y Q, respectivamente. El procedimiento continua hasta alcanzar el criterio de parada. De esta forma, la búsqueda utiliza diferentes combinaciones de conjuntos reducidos de objetivos en cada generación.

Antes de su ejecución, el ϵR-EMO requiere que se establezcan como se muestrearán los subespacios. En [AT09], se analizan tres estrategias: aleatorio, rotación y fijo. Además, el ϵR-EMO requiere que el tomador de decisiones determine el número de soluciones que desea considerar en cada partición, así como el número de generaciones antes de crear una nueva partición. En [LJCCAT11], el ϵR-EMO original fue modificado a fin de determinar de manera automática estos parámetros. En este caso, el número de soluciones en cada partición se define de acuerdo a la contribución de cada subespacio al conflicto total del problema, mientras que un método de detección de convergencia se utiliza para determinar cuándo una nueva partición es necesaria.

En [AT09] el método ϵR-EMO fue evaluado experimentalmente en el problema *MNK-Landscapes* [AT04] utilizando 4 a 10 objetivos. Como métricas de comparación, el trabajo utilizó Hipervolumen y Cobertura [ZT99a]. De acuerdo a los resultados obtenidos, la convergencia y la diversidad de las soluciones encontradas se mejoran con el ϵR-EMO en todos los casos. Sato et al. [SAT10] compararon un método similar al presentado en [AT09] con un muestreo aleatorio de objetivos, el cual se llama en ese trabajo *Partial Pareto Dominance MOEA* - (PPD-MOEA), contra el IBEA [ZK04], un método basado en la relación de expansión [SAT07], el NSGA-II [DPAM02] y el MSOPS [Hug05] para el problema de la mochila 0/1 con 4, 6, 8 y 10 objetivos. Además, se consideró una variación del PPD-MOEA utilizando la relación de expansión en vez de la relación de dominancia Pareto. En [SAT10], el PPD-MOEA consiguió mayor diversidad que el IBEA, MSOPS y el método basado en la relación de expansión, así como una mejor convergencia que el NSGA-II original.

4.2.5 Resumen sobre los MOEAs para problemas con muchos objetivos basados en transformaciones del problema original

Como se señaló previamente, las dificultades para resolver los problemas *many-objective* utilizando MOEAs incluyen: (i) la proporción creciente de soluciones no-dominadas, (ii) el costo computacional, y (iii) la visualización de los resultados. Los métodos basados en relaciones de preferencia se centran en la primera de estas dificultades buscando mejores esquemas para diferenciar las soluciones (ver Sección 4.1); por otra parte, los métodos basados en transformaciones del problema original también consideran, con menor o mayor grado, las otras dos cuestiones.

Los enfoques de agregación se encuentran entre las técnicas más simples que podrían utilizarse para transformar un problema de muchos objetivos en un problema de optimización multi-objetivo o incluso con un único objetivo. En general, estos métodos requieren menos carga computacional que otras alternativas. Por otra parte, teniendo en cuenta las combinaciones de objetivos, los enfoques basados en agregación pueden facilitar la visualización y toma de decisiones. Sin embargo, la principal desventaja de este tipo de MOEAs es que su capacidad de búsqueda depende de la elección de la función de escalarización, así como de la selección de objetivos a ser combinados. Ambas cuestiones se relacionan con las características del problema que se optimiza, por lo tanto, representan un inconveniente grave para la robustez de estas técnicas. De hecho, para el método basado en las funciones de deseabilidad [KEB+09], el tomador de decisiones tiene que definir a priori los objetivos que serán combinados. Por lo tanto, dos aspectos importantes para futuras investigaciones sobre los MOEAs basados en agregación son: cómo identificar los objetivos a combinar, y qué función de escalarización utilizar.

Una desventaja significativa de los métodos basados en indicadores es el excesivo tiempo de ejecución para el cálculo del indicador de calidad, especialmente para los

métodos basados en Hipervolumen. Para aliviar este impedimento, se propusieron los siguientes enfoques: (i) combinar la búsqueda basada en indicadores con técnicas de reducción objetivos [BZ07], y (ii) usar aproximaciones a los valores del indicador en lugar de un cálculo exacto [BZ11]. El análisis de otras posibilidades para mejorar el costo computacional de estos algoritmos es claramente un tema de investigación interesante para ser explorado.

Las técnicas de reducción de la dimensionalidad se pueden combinar con otros métodos como una forma de superar las dificultades que surgen en problemas con muchos objetivos, tales como el costo computacional y la dificultad de visualización. De hecho, la principal ventaja de este enfoque es que mediante la reducción de un problema de optimización con muchos objetivos en uno multi-objetivo, éste puede ser resuelto con MOEAs existentes. Sin embargo, en problemas donde el conjunto de objetivos es mínimo, el método no es viable. Hay varias técnicas de reducción del número de objetivos que se pueden utilizar, pero determinar cuál de ellas se ajusta mejor para una clase dada de problemas es todavía un área de investigación prometedora. En los casos estudiados, la eliminación de los objetivos depende de las soluciones actuales en la población. En la mayoría de los MOEAs, la población evoluciona a partir de un muestreo aleatorio del espacio de búsqueda hacia una aproximación al conjunto de Pareto, por lo tanto, dependiendo del momento en el que se ejecuta la reducción del número de objetivos, se puede eliminar un conjunto diferente de objetivos. Por consiguiente, la determinación de los criterios para ejecutar el método de reducción de objetivos se convierte en un problema importante a ser analizado.

Por último, a pesar de los resultados promisorios presentados en [AT09, LJCCAT11, SAT10], son todavía escasos los trabajos que estudian los enfoques de partición del espacio de objetivos. Estos métodos parecen ser muy atractivos para ser aplicados en combinación con relaciones de preferencia como en [SAT10]. Además, podrían ser implementados junto con los métodos basados en indicadores, lo que proporcionaría una forma sencilla de reducir la carga en la ejecución de estos métodos mediante el muestreo de objetivos.

Capítulo 5
MOEA basados en descomposición

Comparado con las otras categorización de MOEA, los basados en descomposición tienen como principales ventajas: la posibilidad de una ejecución más rápida, la posibilidad de paralelización y la posibilidad de mejorar la diversidad de la población utilizando vectores de peso o puntos de referencia distribuidos adecuadamente [XXM20]. Como se ha señalado en años recientes, los MOEA basados en descomposición han atraído a la comunidad de investigadores con el fin de aprovechar sus ventajas en la aplicación de estos algoritmos en problemas de optimización con muchos objetivos.

Existen dos métodos básicos para la descomposición de problemas: (i) el uso de funciones de agregación y (ii) el uso de puntos de referencia [vLBB19]. Entre los algoritmos que utilizan funciones de agregación destacan el *Multiple Single Objective Pareto Sampling* (MSOPS) [Hug03] y el *Multi-objective Evolutionary Algorithm based on Decomposition* (MOEA/D) [ZL07]. Mientras que el MOEA/D-M2M [LGZ14], el *Non-dominated Sorting Genetic Algorithm III* (NSGA-III) [DJ14a] y el *Reference Vector Guided EA* (RVEA) [CJOS16] son algoritmos basados en puntos de referencia.

En la siguiente sección presentamos los principales algoritmos basados en descomposición. Finalmente, presentaremos las conclusiones y analizaremos algunos temas de investigación futura.

5.1 Métodos basados en escalarización: *Multiple Single Objective Pareto Sampling (MSOPS)*

El *Multiple Single Objective Pareto Sampling* (MSOPS) [Hug03, Hug05] puede considerarse como el primer algoritmo evolutivo utilizado para resolver problemas con muchos objetivos. Este algoritmo requiere que el usuario especifique a priori un conjunto de vectores de pesos, también llamados vectores destino (*target vectors*). Una vez definido este conjunto de $\mathbf{T}$ vectores de pesos $\{W^1, \ldots, W^{\mathbf{T}}\}$ estos se utilizan para evaluar cada solución utilizando un método min-max ponderado (*weighted min-*

max). Se obtiene así, luego de la evaluación, que cada una de las soluciones **x** tienen **T** valores, uno para cada vector de pesos.

El puntaje del min-max ponderado de la solución **x** y vector de pesos W^j, denotado por $\mathbf{S}^j(\mathbf{x})$ se calcula utilizando la siguiente ecuación:

$$\mathbf{S}^j(\mathbf{x}) = \max_{i=1}^{m} \{W_i^j \cdot f_i(\mathbf{x})\} \tag{5.1}$$

donde W_i^j es el i-ésimo componente del vector de pesos W^j.

Los puntajes de todas las soluciones se guardan en filas de una matriz **S** de tamaño $|P| \times \mathbf{T}$ ($|P|$: tamaño de la población); entonces, la columna j de **S**, denotada por $\mathbf{S}^j$, corresponde al puntaje del vector W^j. Cada columna $\mathbf{S}^j$ se ordena en forma ascendente para producir un ranking de soluciones $\mathbf{R}^j$ para el correspondiente vector de pesos W^j; así, la posición 1 es para la mejor solución, mientras que la peor solución tiene un valor igual al de la población que está siendo clasificada. Los valores de las posiciones se guardan en una matriz **R**, la cual se utiliza para obtener el ranking final de la población. Se considera como la mejor solución a aquella que tiene más puntajes en la posición 1, y así por delante.

Hughes [Hug05] comparó el desempeño del MSOPS y del NSGA-II [DPAM02] con respecto al *Hypervolume* [ZT99a] utilizando dos problemas con 4 y 6 objetivos definidos en [VL98]. Los resultados en [Hug05] mostraron que el MSOPS supera al NSGA-II para la métrica y los problemas considerados.

Luego, [Hug07] presentó el MSOPS-II que considera dos variaciones con respecto al MSOPS original. La primera modificación fue un método que utiliza la población actual como entrada para generar un nuevo conjunto de vectores de pesos mientras el proceso de optimización se ejecuta. La segunda variante fue una reducción de la complejidad temporal del método de asignación del valor de adaptación del MSOPS considerando la adaptación de un miembro de la población como la mejor puntuación obtenida luego de escalar los resultados de cada vector objetivo. En [Hug07], se comparó el MSOPS-II con el MSOPS original y un procedimiento de búsqueda aleatoria utilizando un problema de 2 objetivos definido en [TWFT95], y un nuevo problema definido en [Hug07] con 5 objetivos. Para evaluar los resultados obtenidos, se utilizaron las métricas *Hypervolume* [ZT99a] y *Multi-Objective Equivalent Random Search* (MOERS) [Hug06]. Los resultados experimentales mostraron que el método de ranking del MSOPS-II puede ser aplicado efectivamente como un algoritmo de propósito general en problemas con muchos objetivos requiriendo una mínima configuración inicial.

5.2 Multi-objective Evolutionary Algorithm based on decomposition (MOEA/D)

Este algoritmo, al igual que un enfoque previamente presentado en [MIG01], se basa en la utilización de una función de escalarización en donde se tiene un vector de peso

diferente para cada individuo. De esta manera, el problema original se descompone en una colección de problemas de optimización escalares, es decir, un problema para cada individuo de la población evolutiva.

La característica sobresaliente del MOEA/D es que dicho algoritmo establece una relación de vecindad entre los diferentes subproblemas de acuerdo a la distancia que existe entre los vectores de los coeficientes de agregación así como el apareamiento y reemplazo local. De esta forma, cada uno de los subproblemas en los que se divide el problema original puede ser optimizado utilizando solo información sobre los subproblemas vecinos con lo que se consigue una mejora importante en la velocidad de convergencia del algoritmo.

En el Algoritmo 12 se presenta la secuencia de pasos del MOEA/D. Como datos de entrada este algoritmo requiere que el usuario defina un conjunto de vectores de peso $\lambda^1 \ldots \lambda^N$ de tamaño igual a la población. Los puntos de este conjunto deben estar uniformemente distribuidos y satisfacer ciertas condiciones [ZL07]. La generación correcta de tales pesos es un tema importante que ha sido tratado en varios trabajos; el lector interesado puede consultar a [ZYZJ15, HHH17, SZJYS11, QML+14, GPF14]. En este sentido, se han realizado propuestas para ajustar el vector de pesos de manera adaptativa [DLH20, MT20]. Además, otro parámetro necesario específico es el tamaño T de la vecindad a considerar.

Independientemente del proceso utilizado para generarlos, cada vector de pesos λ^j se asocia con un único individuo j de la población P. En la sentencia 5 del Algoritmo 12 se calcula para cada uno de los vectores de peso la distancia euclidiana a los demás vectores de peso. Luego, estas distancias se utilizan para definir el conjunto de vecinos de cada uno de los vectores λ^i, denotado por $B(i)$, compuesto por los T vectores más cercanos. Se genera entonces la población de tamaño N y se calcula el valor objetivo de cada elemento de la población. Posteriormente, se inicializa el valor de $z^* = z_1^*, ..., z_m^*$ que está definido como $z_i^* = \min\{f_i(x) \mid x \in \Omega\}$. Es decir, es el mejor valor del objetivo i-ésimo encontrado entre todas las evaluaciones de la función objetivo con cada individuo perteneciente al conjunto de soluciones (considerando minimización) y representa el valor ideal para cada objetivo del problema para ser utilizado en el método de escalarización de Tchebycheff, como se explica más adelante.

Luego de la etapa de inicialización se ejecutan los pasos de la evolución. Al tener asignando a cada elemento de la población i un único vector de pesos λ^i y con su correspondiente vecindad $B(i)$, se define la vecindad de i como el conjunto de soluciones con subproblemas en $B(i)$. La selección y el cruzamiento se ejecutan considerando cada vecindad. Existen diferentes propuestas que se han desarrollado para mejorar tanto la selección [LZ08, CL11, WZGZ14] como el cruzamiento en MOEA/D [LZ08]. En el modelo básico de [ZL07], para cada elemento i se hace la selección aleatoria de un par de índices j y l de $B(i)$, estos índices se utilizan para el procedimiento de cruzamiento con los elementos correspondientes de la población y obtener un nuevo vector x. Para obtener este nuevo vector se utiliza un mecanismo inspirado en la evolución diferencial. Luego, de ser necesario, se aplica un proceso de reparación de la solución específico del problema. El nuevo individuo obtenido se evalúa y en caso de que corresponda se actualiza el valor de z^*. Si alguno de los

valores objetivo de la nueva solución generada tiene un valor mejor que alguno de los valores del punto de referencia actual, z^* debe ser actualizado con el nuevo valor (línea 22).

En la línea 25 se da inicio a la fase de actualización de cada subproblema. Aquí se debe determinar si la nueva solución generada es mejor que algún individuo perteneciente a la vecindad del subproblema actual. Esto se realiza por medio de la función de escalarización, que en este caso se calculará con el método Tchebycheff. Existen diferentes alternativas para la escalarización, en el trabajo de [ZL07] se evaluaron tres opciones de escalarización para el MOEA/D: suma ponderada, Tchebycheff ponderado (*weighted Tchebycheff*) y el enfoque de intersección de límites (*boundary intersection approach*) [ISTN09, ZL07].

En el enfoque de Tchebycheff, la escalarización se realiza de la siguiente forma:

$$\min g^{te}(\, x, \lambda, z^*) \;=\; \max_{1 \le i \le m} \{\lambda_i \mid f_i(x) - z_i^*\} \forall x \in \Omega \tag{5.2}$$

En [MZT+17] se estudian los enfoques de descomposición de Tchebycheff para los problemas de optimización multi-objetivo. Por su parte, en [ISTN09], el MOEA/D con suma ponderada [Mie99] fue evaluado favorablemente en problemas *many-objective* con frente Pareto convexo; sin embargo, el MOEA/D con el método de Tchebycheff ponderado [Mie99] tiene ventaja cuando el frente Pareto es no-convexo; entonces, puesto que la elección de la función de escalarización apropiada para utilizar en el MOEA/D depende de la forma del frente Pareto, Ishibuchi et al. [ISTN09] propusieron alternar de manera automática entre suma ponderada y Tchebycheff ponderado.

El método para alternar entre las opciones de escalarización supone que, al usar un método de suma ponderada, varios individuos en un mismo vecindario tendrán el mismo valor del vector objetivo en la búsqueda en áreas no convexas. Por tanto, el procedimiento considera que si existen al menos un número dado, definido por el tomador de decisiones, de soluciones con igual valor del vector objetivo en una vecindad, se debe utilizar el método ponderado de Tchebycheff, de otra forma, se utiliza el método de suma ponderada. Además de esto, más recientemente, se han estudiado diferentes alternativas de mejora para los métodos de descomposición y otras maneras de combinarlos [PRC18].

Siguiendo con el Algoritmo 12, luego de la aplicación de la función de escalarización se utiliza el valor obtenido para comparar cada una de las demás soluciones de la vecindad. Cuando el el valor de aptitud de la solución nueva es mayor que el de alguna solución de la vecindad, la solución nueva la reemplaza. De esta manera, MOEA/D mantiene siempre dentro de la población las mejores soluciones encontradas para cada subproblema. Al terminar de procesar los n subproblemas, se pasa a una nueva iteración del algoritmo.

Previamente al método propuesto en [ZL07], Ishibuchi y Morima [IM98] propusieron un método de descomposición llamado *Multi-objective Genetic Local Search* (MOGLS), el cual luego fue mejorado por Jaszkiewicz [Jas02]. Básicamente, MOGLS genera en cada iteración un vector de pesos aleatorio para evaluar la población actual y utiliza una población externa para almacenar soluciones no

Algoritmo 12: Pseudocódigo MOEA/D

Datos: $\lambda^1, ..., \lambda^n$ = vector de pesos distribuidos de manera uniforme
T = número de vecinos de cada λ^i a considerar
N = tamaño de la población P
t_{max} = número máximo de generaciones

```
/* Inicialización */
Hacer EP =
/* Calcular los T vectores mas cercanos de cada vector λ */
para i ← 0 a N − 1 hacer
    para j ← 0 a N − 1 hacer
        si i ≠ j entonces
            distancia_euclidiana(λ^i, λ^j) ;
        fin
    fin
fin
para cada i ≤ N hacer
    Definir el conjunto de vecinos de i como B(i) = {i^1 ... i^T } tal que cada λ_{i^1} ... λ_{i^T}
      son los T elementos más cercanos a λ^i;
fin
Generar una población inicial P de tamaño N aleatoriamente ;
Calcular la función objetivo para cada elemento de la población ;
Inicializar el valor de z* = z*_1, ..., z*_m, donde cada elemento de z* corresponde al mínimo
  valor del objetivo i-ésimo de todos los elementos de P ;
/* Evolución */
mientras t < t_max hacer
    para i ← 0 a N − 1 hacer
        seleccion de pares: seleccionar aleatoriamente dos índices k y l de B(i) ;
        reproducción: Obtener una nueva solución x a partir de P[k] y P[l] ;
        reparación: Reparar la solución x utilizando heuristicas específicas del problema;
        para j ← 1 a m hacer
            si z_j < F_j(x) entonces
                z_j = F_j(x) ;
            fin
        fin
        /* Actualizar subproblema */
        para cada j ∈ B(i) hacer
            si g^te(x | λ^j, z) ≤ g^te(y^j | λ^j, z) entonces
                y^j = x;
                F(y^j) = F(x);
            fin
        fin
    fin
    t = t + 1;
fin
```

dominadas. En [ZL07], se demostró que el MOEA/D produce mejores aproximaciones que el MOGLS en el problema de la mochila con 4 objetivos. Por otra parte, en [IHN08] se mostró que el MOGLS puede trabajar bien en problemas con muchos objetivos.

Los inconvenientes principales del MOEA/D son: una posible reducción de la diversidad en la población y la necesidad de una selección adecuada de valores de vectores de peso. Para lidiar con ambos problemas, en [LGZ14] se propone el MOEA/D-M2M. Este algoritmo tiene como idea principal de descomponer el MaOP original en un conjunto de problemas multiobjetivos con un número dado de K objetivos en lugar de problemas con un solo objetivo. En este método, no se utilizan métodos de agregación sino vectores de dirección para dividir el espacio objetivo en los subespación de K objetivos. Un vector está en un subespacio dado, si tiene el ángulo más pequeño con el vector unitario correspondiente a un subespacio dado entre todos los demás vectores de dirección K. Este algoritmo utiliza un conjunto de subpoblaciones para cada uno de los subproblemas considerados. El tamaño de cada subpoblación es un parámetro del algoritmo. Estas subpoblaciones se establecen inicialmente de manera aleatoria. Luego, mientras no se cumpla el criterio de detención, cada subpoblación se considera a la vez para la aplicación de los operadores evolutivos de forma a crear una nueva descendencia. Al final de cada iteración, toda la población se divide en las subpoblaciones; sin embargo, en este caso los vectores se asignan a cada subpoblación considerando la división del problema del subespacio. Finalmente, el conjunto de soluciones no dominadas en todo el conjunto de poblaciones se obtiene como salida del algoritmo. En[LGZ14], el MOEA/D-M2M se comparó con MOEA/D y NSGA-II [DPAM02], mostrando un rendimiento mejorado en un conjunto de instancias de prueba.

5.3 Nondominated Sorting Genetic Algorithm - III (NSGA-III)

El algoritmo NSGA-III [DJ14b] es un MOEA basado en descomposición que utiliza puntos de referencia bien distribuidos en el espacio de soluciones con el fin de proporcionar la diversidad necesaria en las soluciones a fin de guiar la búsqueda de manera eficiente. Los pasos básicos de este algoritmo son los mostrados en el Algoritmo 13. Como puede observarse, en términos generales los pasos que sigue el NSGA-III son similares al NSGA-II [DPAM02], presentado en detalle en la Sección 3.2.2. Las diferencias fundamentales entre ambas versiones son la necesidad de calcular puntos de referencia para colocar en el hiperplano, la necesidad de un mecanismo para normalizar los valores de la función objetivo, la asociación de los miembros de la población a los puntos de referencia y la técnica de cálculo del valor de nicho.

En el Algoritmo 14 se presenta el una versión más completa del del NSGA-III [MKSOvLt20, DJ14b]. Como puede verse, se toma como entrada un conjunto de puntos de referencia V. Estos puntos de referencia pueden estar generados de forma automática o manual. En el caso de la generación automática, en [DJ14b] se calculan H puntos distribuidos en el espacio de un hiperplano utilizando el enfoque de Das y Dennis el cual ubica los puntos de manera uniforme [DJ14b]. Si se consideran p divisiones, el número de puntos de referencia H en un problema con m objetivos es:

$$H = \binom{m + p - 1}{p} \tag{5.3}$$

En el NSGA-III primeramente se genera la población padre inicial de forma aleatoria. Esta población padre es utilizada en el ciclo iterativo para obtener a partir de ella una población hijo Q_t a partir de la aplicación de los algoritmos de recombinación y mutación. Típicamente se utiliza el cruzamiento por SBX (*Simulated Binary Crossover*). Seguidamente, se calculan los frentes $\{\mathcal{F}^1, \mathcal{F}^2 \dots\}$ de no-dominancia de $P(t) \cup Q_t$ y se llena la población $P(t+1)$ con los elementos de cada frente. Como en el NSGA-II, se utiliza un mecanismo de nicho para determinar los elementos a ser incorporados en la población siguiente en caso que un desempate sea necesario. Para esto, se requieren los siguientes pasos: normalización de los valores objetivos, asociación de los elementos de la población a los vectores de referencia y ejecución del procedimiento de nicho.

El método de normalización de los objetivos, presentado en el Algoritmo 15 computa el punto ideal Z^{min}, que almacena los mínimos valores objetivos calculados hasta el momento. Luego, se procede a realizar una translación de objetivos f_i' restando a todos los vectores objetivos f_i de $S(t)$ el vector Z^{min}. Luego, para cada función objetivo, se calculan vectores extremos a partir de los f_i' más cercanos a cada eje objetivo. Estos vectores formarían un hiperplano que se interseca con los ejes objetivo en puntos a_i respectivos a cada eje.

Algoritmo 13: Algoritmo NSGA-III básico

```
1  Calcular el número de puntos de referencia (H) para colocar en el hiperplano utilizando 5.3
2  Generar una población inicial de manera aleatoria teniendo en cuenta los recursos
     asignados
3  Ejecutar la clasificación de la población P por no-dominancia
4  para t = 1 hasta t_max hacer
5  |  Aplicar selección, cruzamiento y mutación sobre la población P para generar la
   |    población hijo Q
6  |  Ejecutar la clasificación de la población por no-dominancia sobre P ∪ Q
7  |  Incluir los elementos de cada frente hasta el frente l antes de llenar la población
   |    P(t + 1)
8  |  si El número de elementos a incluir el frente l supera el tamaño deseado entonces
   |  |  /* Utilizar un procedimiento especial basado en los puntos de
   |  |     referencia para determinar la distancia de nicho */
9  |  |  Normalizar los valores objetivos de cada uno de los elementos de la población
10 |  |  Asociar los miembros de la población con los puntos de referencia
11 |  |  Aplicar el mecanismo de preservación del nicho
12 |  |  Determinar los elementos de F^l para completar P(t + 1) sin superar el tamaño
   |  |    deseado
13 |  fin
14 |  en otro caso
15 |  |  Incluir el frente l en la población P(t + 1)
16 |  fin
17 |  t = t + 1
18 fin
```

Algoritmo 14: Pseudocódigo NSGA-III

Datos: $P(t)$= población de la generación t
V = conjunto de puntos de referencia
$A_{1,\ldots,|V|}$= conjunto de miembros de la población asociados a los vectores de V

$t = 0$ Generar una población $P(t)$ de tamaño N en forma aleatoria;
mientras $t < t_{max}$ **hacer**
  $Q(t) = generar\_poblacion\_hija(P(t))$;
  $R(t) = P(t) \cup Q(t)$; // combinar la población padre e hijo
  $\mathcal{F}$ =**fast_non_dominated_sort**$(R(t))$; // Usar Alg. 9 para obtener los frentes de $R(t)$
  mientras $S(t) \leq | N |$ **hacer**
    $S(t) = S(t) \cup \mathcal{F}^i, \quad i = i + 1$;
  fin
  $\mathcal{F}^l = \mathcal{F}^i$ es el último frente a ser incluído
  si $| S(t) | = N$ **entonces**
    $P(t+1) = S(t)$;
    continuar en la siguiente iteración;
  fin
  $P(t+1) = \bigcup_{j=1}^{l-1} \mathcal{F}^j$;
  $k = N - | P(t+1) |$; /* cantidad de individuos a seleccionar */
  $normalizar(S(t))$; /* normalizar valores objetivos */
  /* Asociar cada elemento s de $S(t)$ a un punto de referencia, el más cercano */
  $W = computar\_linea\_referencia(V)$; /* asociar cada puntos a vectores */
  para $i \leftarrow 0$ **a** $| S(t) |$ **hacer**
    para $j \leftarrow 0$ **a** $| W |$ **hacer**
      $d_{i,j} = computar\_distancia\_perpendicular(S(t)[i], W_j)$;
    fin
    $k = \{j \,|\, \arg\min_{j \in \{1,\ldots,|W|\}} d_{i,j}\}$;
    $A_k = A_k \cup \{I_i\}$; /* miembro de la población se asocia al V_k */
  fin
  /* Elegir los k elementos faltantes para construir $P(t+1)$ */
  $niching(P(t+1), A, \mathcal{F}^l, k)$;
  $t = t + 1$;
fin

Una vez que se obtienen estos valores, los vectores f_i' se normalizan haciendo la división de cada dividiendo cada posición por el valor a_i respectivo. Ahora, el hiperplano intercepta los ejes en el valor 1 y los puntos de referencia calculados al

inicio coinciden con este plano. Con este mecanismo, se busca mantener la diversidad de las soluciones cuando se trata con objetivos en escalas desiguales.

Algoritmo 15: Procedimiento de Normalización NSGA-III

Datos: $S(t)$= población de individuos actual
`/* Traducción de vectores objetivo y cálculo de vec. extremos */`
para $i \leftarrow 1$ **a** M **hacer**
  $z_i^{min} = min_{s \cup S(t)} f_i(s)$;
  $f_i'(s) = f_i(s) - z_i^{min}$;
  $P_{1,\ldots,M} = calcular\_extremos(S(t))$;
fin
`/* Hallar puntos de intersección */`
para $i \leftarrow 1$ **a** M **hacer**
  $a_i = calcular\_interseccion(P, i)$
fin
`/* Normalizar los vectores objetivo */`
para cada $s \cup S(t)$ **hacer**
  $f^n_{1,\ldots,M} = f^n_{1,\ldots,M} / a_{1,\ldots,M}$
fin

La asociación de cada uno de los individuos a los puntos de referencia se realiza determinando para cada individuo el punto de referencia más cercano utilizando la distancia perpendicular entre cada vector objetivo y las líneas que unen los puntos de referencia con el origen (línea 18).

Note que un punto de referencia puede no estar asociado a ningún elemento de la población, a uno sólo o a varios. El algoritmo de *niching* utiliza esta información para seleccionar los K elementos faltantes para llenar $S(t)$. Este procedimiento se describe en el Algoritmo 16, y consiste primeramente en verificar cuáles V_j presentan la menor cantidad de individuos (p_j) ya seleccionados para formar parte de la siguiente generación. Se selecciona luego uno aleatorio entre dichos V_j (lineas 3 y 4 del Algoritmo 16) y, con el ciclo de la línea 5 se seleccionan solo los individuos asociados a V_j que sean también miembros de $\mathcal{F}^l$, formando el conjunto I'. Una vez realizado este proceso, pueden darse tres situaciones:

a. Si p_j es 0 (linea 11), se selecciona el individuo de $\mathcal{F}^l$ más cercano al punto de referencia para agregar a $P(t+1)$.
b. Si p_j es ≥ 1 (linea 13), se elige un individuo (aleatorio o con algún criterio) de $\mathcal{F}^l$ que esté asociado al punto de referencia para agregar a $P(t+1)$.
c. En cambio, si no existen individuos de $\mathcal{F}^l$ asociados (linea 19), entonces el punto de referencia ya no se considera en los próximos ciclos de este proceso.

Este proceso se repite K veces, hasta que la población $P(t+1)$ alcance los N individuos.

Algoritmo 16: Procedimiento de Niching NSGA-III

Datos: K = número de miembros a ser seleccionados
$\mathcal{F}^l$ = último frente de Pareto
V = conjunto de puntos de referencia
A_j = conjunto de individuos asociados a un punto V_j
ρ_j = número de miembros de un punto V_j que ya pertenecen a $P(t+1)$
$d(s)$ = distancia entre un individuo s y su punto de referencia asociado

```
k = 0;
mientras k < K hacer
    j_min = {j : arg min_{j∈V} ρ_j};
    j = random(j_min);
    para cada s ∈ A_j hacer
        si s ∈ F^l entonces
            I' = I' ∪ {s};
        fin
    fin
    si I' ≠ 0 entonces
        si ρ_j = 0 entonces
            P(t+1) = P(t+1) ∪ {s : arg min_{s∈I'} d(s)}
        en otro caso
            P(t+1) = P(t+1) ∪ random(I')
        fin
        ρ_j = ρ_j + 1;
        F^l = F^l \ {s}, A_j = A_j \ {s};
        k = k + 1;
    en otro caso
        V = V \ {V_j}
    fin
fin
```

5.4 Reference Vector Guided Evolutionaty Algorithm (RVEA)

El Reference Vector Guided Evolutionaty Algorithm (RVEA) propuesto por Cheng et al. [CJOS16] es otro algoritmo que, como el NSGA-III, se basa principalmente en una estrategia de descomposición y en el procedimiento elitista del NSGA-II [DPAM02] para dividir el problema en subproblemas, formando parte de un subproblema cada subespacio de soluciones que se encuentra cerca de uno de dichos vectores de referencia.

A diferencia de otros algoritmos, el RVEA propone un enfoque de escalarización llamado *Angle-Penalized Distance* (APD). Este procedimiento esta diseñado para

Algoritmo 17: Pseudocódigo básico del RVEA [MKSOvLt20, CJOS16].

Datos: el número máximo de generaciones t_{max}, un conjunto de vectores de referencia de la iteración inicial $V_0 = \{V_{0,1}, V_{0,2} \dots V_{0,N}$
M = número de objetivos
f_r = frecuencia de adaptación de los vectores de referencia
N = cantidad de vectores de referencia

Inicializar la población P_0 aleatoriamente con N individuos;
mientras $t < t_{max}$ **hacer**
  Aplicar selección, cruzamiento y mutación sobre la población P para generar la población hijo Q;
  Hacer $P(t) = P(t) \cup Q(t)$;
  `/* Utilizar la selección guiada por vectores los de referencia` V_t `para obtener` $P(t+1)$ `*/`
  para $i \leftarrow 0$ **a** M **hacer**
    para $j \leftarrow 0$ **a** $\mid P(t) \mid$ **hacer**
      $z_{t,i}^{min} = min(f_{i,j}, z_{t,i}^{min})$
    fin
  fin
  para $i \leftarrow 0$ **a** $\mid P(t) \mid$ **hacer** `/* Trasladar objetivos */`
    $f'_{t,i} = f_{t,i} - z_t^{min}$;
  fin
  $seleccion\_por\_vectores\_de\_referencia(V_t, P(t), N)$;
  `/* Utilizar el procedimiento de adaptación de para obtener` $Vt+1$ `a partir del conjunto` V_t `y` $P(t+1)$ `*/`
  si $\frac{t}{t_{max}} \mod f_r == 0$ **entonces**
    $V_{t+1} = adapatacion\_de\_V\,(P(t+1), V_t, V_0)$
  en otro caso
    $V_{t+1} = V_t$;
  fin
  $t = t + 1$
fin

balancear la convergencia y la diversidad de acuerdo al número de objetivos y la generación. Además, el RVEA propone una estrategia para ajustar de manera adaptativa los vectores de referencia para lidiar con aquellos vectores objetivo que no pueden ser correctamente normalizados. Adicionalmente, permite utilizar un método de articulación de preferencias que es capaz de generar distribuciones Pareto óptimas con valores objetivo en regiones específicas del espacio objetivo. Por último, en [CJOS16] se plantea el uso de una estrategia de regeneración del vector de referencia.

El Algoritmo 17 presenta el pseudocódigo básico del RVEA. El algoritmo toma como entrada un conjunto de vectores de referencia unitarios V_0, y una población P_0 con N individuos aleatorios. Los vectores de referencia se obtienen a partir de un conjunto de puntos uniformemente distribuidos en el espacio de un hiperplano que interseca con todos los objetivos. Estos puntos u_i se generan matemáticamente, y a partir de éstos se obtienen los vectores de referencia unitarios con $v_i = \frac{u_i}{\|u_i\|}$.

Como puede verse en el Algoritmo 17, el procedimiento de elitismo es similar al del NSGA-II y NSGA-III. Sin embargo, una vez que se forma la población combinada $P \cup Q$, los elementos que se incluirán en la siguiente población evolutiva se determinan mediante un procedimiento de selección guiado por vector de referencia. El primer paso del procedimiento de selección guiado por vectores de referencia algoritmo es la translación de los valores objetivos de la población combinada de padres e hijos al cuadrante de los vectores de punto de referencia ideal. Para ello se calcula z_t^{min} (línea 7), un vector que contiene los valores objetivo mínimos para la generación t actual. Ese vector se utilizar para realizar la traslación de los valores objetivo. Esta translación consiste en restar los valores objetivo de cada individuo se restan con z_t^{min} para pasarlos al primer cuadrante, y permitir el uso del origen de coordenadas como punto de referencia ideal. El conjunto de vectores de referencia requeridos por el algoritmo se puede generar usando un enfoque introducido en [CJNS15], o ser definido por el usuario para representar sus preferencias.

Después de la translación de los valores objetivos, la población se divide en subpoblaciones que asocian a cada individuo con sus vectores de referencia angularmente más cercanos. Luego, un procedimiento, denominado cálculo de distancia con penalización de ángulo, analiza las propiedades de convergencia y diversidad de cada solución en las diferentes poblaciones, es decir, con respecto al vector de referencia con el que están asociadas las soluciones candidatas. Esta función busca que en las primeras generaciones, la aptitud de los individuos se guíe principalmente por el vector objetivo de un individuo $\|f'_{t,i}$, fomentando una convergencia rápida. Sin embargo, luego, la aptitud será guiada principalmente por los ángulos entre los individuos y sus vectores de referencia, ayudando a la diversificación de los resultados. La función llamada distancia de penalización de ángulo *Angle-Penalized Distance* (Angle-Penalized Distance (APD)) se define como:

$$d_{t,i,j} = (1 + P(\theta_{t,i,j})) \cdot \|f'_{t,i}\| \tag{5.4}$$

donde

$$P(\theta_{t,i,j}) = M \cdot (\frac{t}{t_{max}})\alpha \cdot \frac{\theta_{t,i,j}}{\gamma_{v_{t,j}}} \tag{5.5}$$

$\|f'_{t,i}\|$ = distancia de $f'_{t,i}$ al punto ideal (origen de coordenadas) $\theta_{t,i,j}$ = ángulo agudo entre $f'_{t,i}$ y $v_{t,j}$ $\gamma_{v_{t,j}}$ = menor ángulo entre el V_{ref} $v_{t,j}$ y todos los demás V_{ref} α = variable definida por el usuario que controla la velocidad de cambio del énfasis de la convergencia hacia la distribución

Finalmente, según el valor otorgado por la función APD, el individuo en cada subpoblación que tenga la menor distancia es seleccionado para formar parte de la población en la siguiente generación.

Los experimentos muestran que en caso de tener objetivos escalados a diferentes rangos, los vectores de referencia distribuidos uniformemente no producirán soluciones distribuidas uniformemente [CJOS16]. Así, al final de algunas iteraciones, RVEA propone una estrategia para ajustar los vectores de referencia a los rangos de valores objetivos con el fin de obtener soluciones distribuidas uniformemente, incluso si las funciones objetivo no están normalizadas en el mismo rango, por ello,

Algoritmo 18: Selección guiada por vectores de referencia [MKSOvLt20, CJOS16].

Datos: $V_t = v_{t,1}; \ldots; v_{t,N}$ = conjunto de vectores de referencia
$P_{t,1..N}$ = población de individuos de la generación t
N = cantidad de vectores de referencia
I_t = conjunto de individuos de la generación t

para $i \leftarrow 0$ **a** $|\ P(t)\ |$ **hacer** `/* Descomposición de la población */`
  para $j \leftarrow 0$ **a** N **hacer**
    $\cos\theta_{t,i,j} = \frac{f'_{t,i} \cdot v_{t,j}}{\|f'_{t,i}\|}$;
  fin
fin
para $i \leftarrow 0$ **a** $|\ P(t)\ |$ **hacer**
  $k = \{j \,|\, \arg\max_{j \in 1,\ldots,N} \cos\theta_{t,i,j}\}$;
  $P_{t,k} = P_{t,k} \cup I_{t,i}$
fin
para $i \leftarrow 0$ **a** N **hacer** `/* Cálculo de APD */`
  para $j \leftarrow 0$ **a** $|\ P(t)\ |$ **hacer**
    $d_{t,i,j} = (1 + P(\theta_{t,i,j})) \cdot \|f'_{t,i}\|$;
  fin
fin
para $i \leftarrow 0$ **a** N **hacer** `/* Selección elitista */`
  $k = \{j \,|\, \arg\min_{j \in 1,\ldots,P(t)} d_{t,i,j}\}$;
  $P(t+1) = P(t+1) \cup I_{t,k}$;
fin

con el fin de distribuir más uniformemente las soluciones óptimas. La fórmula que se aplica para ello es la siguiente:

$$v_{t+1,i} = \frac{v_{0,i} \circ (z^{max}_{t+1} - z^{min}_{t+1})}{\|v_{0,i} \circ (z^{max}_{t+1} - z^{min}_{t+1})\|} \tag{5.6}$$

Donde $v_{0,i}$ indica el vector de referencia i inicial, es decir, en su versión de cuando recién fue inicializado al inicio del algoritmo, z_{max} y z_{min} representan los valores máximos y mínimos de cada función objetivo en la generación siguiente, y cada posición del vector $v_{0,i}$ se multiplica por cada posición del vector (z_{max}-z_{min}).

5.5 Oportunidades de investigación y conclusiones

Los temas de investigación principales en el área de algoritmos basados en descomposición se enfocan en:

- desarrollo de nuevos algoritmos
- mejorar los métodos de generación de vectores de peso [ZYZJ15, HHH17, SZJYS11, QML+14]
- mejorar y adaptar nuevas técnicas de descomposición para su utilización en el marco de la optimización evolutiva [ISTN09, WZZ+15, MZT+17, JYWL18],
- incorporación de criterios de preferencia en la búsqueda [PN15, LCL+18]
- desarrollar nuevos algoritmos basados en descomposición [CJOS16, DJ14b],
- utilización de diseños cooperativos [PRC18],
- balancear la diversidad y la convergencia utilizando vectores de peso o puntos de referencia adaptativos [YXW+15],
- adaptar y utilizar los algoritmos basados en descomposición a nuevos y diferentes tipos de problemas,
- mejorar la diversidad de las soluciones [LJM+17, YXW+15, WLZ+18, WEDB19],
- hibridización con métodos basados en indicadores. [WEDB19, LYL+18, TCZ+17, MSJ+16]

Con respecto a la mejora de las técnicas de descomposición, uno de los temas que consideramos más promisorios es la posibilidad de incorporación de criterios de preferencia en la búsqueda. Esto implica, entre otros aspectos, mejorar la capacidad de dividir el problema considerando las preferencias entre los objetivos, además de la posibilidad de realizar un enfoque mixto que incorpore las funciones de preferencia presentadas en el Capítulo 4. Creemos que en este punto la utilización de la técnica presentada en [vBB15] puede ser una oportunidad que debe ser explorada.

En la misma línea de la utilización de preferencias, consideramos interesante la utilización del método propuesto en [vBB15] con algoritmos basados en descomposición. En [vBB15], la población se divide en múltiples subpoblaciones, cada una representando soluciones cercanas o similares entre sí. A diferencia de otros algoritmos de particionamiento, el método propuesto en ese trabajo, llamado shape, se considera el orden relativo de los valores de los objetivos en las soluciones y vectores de orden seleccionados de manera aleatoria. Una posibilidad es reemplazar los vectores de orden aleatorios por vectores que correspondan a preferencias entre los objetivos establecidas por un tomador de decisión.

Igualmente la técnica de aplicación de *clustering* para la división de la población, ha demostrado ser útil para realizar el *niching* de una mejor manera y puede servir para clasificar mejor a los individuos, así como a los vectores de peso, produciendo una alternativa de división del espacio de búsqueda sin necesidad de vectores de referencia. Otra área de interés es la de analizar técnicas de *clustering* alternativas para resolver la dificultad que implica la utilización de vectores de referencia, así como el cálculo de ángulos y penalidades. Esto ha sido tratado de manera inicial en [MKSOvLt20] con resultados promisorios.

Otra cuestión relevante es la introducción de mejores técnicas para adaptar la búsqueda conforme esta se ejecuta, realizando un balance adecuado entre convergencia y diversidad como se ha venido trabajando en los distintos algoritmos. Creemos también que acá la técnica planteada en [vBB15] puede permitir regular estos aspectos por medio del ajuste de los parámetros que regulan la ejecución del procedimiento de división.

Como línea de continuidad para trabajos futuros se proponen los siguientes temas:

- Implementación de diferentes MOEAs basados en descomposición utilizando el framework presentado en [vBB15] y la comparación con las versiones originales
- Diseño e implementación de un método basado en preferencias que haga uso de las ideas de clustering para su aplicación
- Adaptar el marco de trabajo en [vBB15] para incorporar la idea de adaptabilidad durante el proceso de búsqueda a fin de conseguir soluciones más cercanas y mejor distribuidas con respecto al Frente Pareto

Referencias

[AC17] Luis Miguel Antonio and Carlos A Coello Coello. Coevolutionary multiobjective evolutionary algorithms: Survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 22(6):851–865, 2017.

[AT04] Hernán Aguirre and Kiyoshi Tanaka. Insights on properties of multiobjective MNK-Landscapes. In *2004 IEEE Congr. on Evol. Comput.*, 2004. doi:10.1109/CEC.2004.1330857.

[AT08] Hernán Aguirre and Kiyoshi Tanaka. Robust optimization by ϵ-ranking on high dimensional objective spaces. In *Simulated Evolution and Learning*. Springer, 2008. doi:10.1007/978-3-540-70928-2_54.

[AT09] Hernán Aguirre and Kiyoshi Tanaka. Many-objective optimization by space partitioning and adaptive ϵ-ranking on MNK-Landscapes. In *Proc. of the 5th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2009*, volume 5467 of *Lect. Notes in Comput. Sci.*, pages 407–422. Springer, Berlin, 2009.

[BDZ10] J. Bader, K. Deb, and E. Zitzler. Faster hypervolume-based search using Monte Carlo sampling. *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems*, pages 313–326, 2010.

[BES17] Slim Bechikh, Maha Elarbi, and Lamjed Ben Said. Many-objective optimization using evolutionary algorithms: A survey. In *Recent advances in evolutionary multi-objective optimization*, pages 105–137. Springer, 2017.

[BFH+07] Dimo Brockhoff, Tobias Friedrich, Nils Hebbinghaus, Christian Klein, Frank Neumann, and Eckart Zitzler. Do additional objectives make a problem harder? In *GECCO 2007*, pages 765–772. ACM Press, 2007.

[BFM97] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, 1997.

[BKS01] Jürgen Branke, Thomas Kaußler, and Harmut Schmeck. Guidance in evolutionary multi-objective optimization. *Adv. in Eng. Software*, 32:499–507, 2001.

[BKST78] J.L. Bentley, HT Kung, M. Schkolnick, and C.D. Thompson. On the average number of maxima in a set of vectors and applications. *Journal of the ACM (JACM)*, 25(4):536–543, 1978.

[BLIS17] Leonardo Bezerra, Manuel López-Ibáñez, and Thomas Stutzle. An empirical assessment of the properties of inverted generational distance on multi- and many-objective optimization. In *EMO 2017*, pages 31–45. Springer, 2017.

[Bre62] H. J. Bremermann. Optimization through evolution and recombination. In M. C. Yovits, G. T. Jacobi, and G. D. Goldstine, editors,

Self-organizing Systems, pages 93–106, Washington, 1962. Spartan Books.

[BSDZ08] Dimo Brockhoff, Kumar Saxena, Kalyanmoy Deb, and Eckart Zitzler. On handling a large number of objectives a posteriori and during optimization. In Joshua Knowles, David Corne, and Kalyanmoy Deb, editors, *Multi-Objective Problem Solving from Nature: From Concepts to Applications*, pages 377–403. Springer, Berlin, 2008.

[BW97] P. J. Bentley and J. P. Wakefield. Finding acceptable solutions in the Pareto-optimal range using multiobjective genetic algorithms. In *Soft Computing in Engineering Design and Manufacturing*, Part 5, pages 231–240, London, 1997. Springer.

[BZ06a] Matthieu Basseur and Eckart Zitzler. A preliminary study on handling uncertainty in indicator-based multiobjective optimization. In *Applications of Evolutionary Computing*, volume 3907 of *Lecture Notes in Computer Science*, pages 727–739. Springer, 2006. doi:10.1007/11732242_71.

[BZ06b] Dimo Brockhoff and Eckart Zitzler. Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In *Parallel Probl. Solving from Nat. (PPSN IX)*, volume 4193 of *Lect. Notes in Comput. Sci.*, pages 533–542. Springer, Berlin, 2006.

[BZ07] Dimo Brockhoff and Eckart Zitzler. Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. In *2007 IEEE Congr. on Evol. Comput.*, 2007. doi:10.1109/CEC.2007.4424730.

[BZ11] J. Bader and E. Zitzler. Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evol. Comput.*, 2011. doi:10.1162/EVCO_a_00009.

[CJNS15] Ran Cheng, Yaochu Jin, Kaname Narukawa, and Bernhard Sendhoff. A multiobjective evolutionary algorithm using gaussian process-based inverse modeling. *IEEE Trans. on Evol. Comput.*, 19(6):838–856, 2015.

[CJOS16] Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):773–791, 2016.

[CL11] Tsung-Che Chiang and Yung-Pin Lai. Moea/d-ams: Improving moea/d by an adaptive mating selection mechanism. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 1473–1480. IEEE, 2011.

[CLR90] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction to algorithms*. The MIT press, 1990.

[CLV07] Carlos Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems, Second Ed.* Springer, New York, 2007.

[Coe00] Carlos A. Coello Coello. An Updated Survey of GA-Based Multiobjective Optimization Techniques. *ACM Computing Surveys*, 32(2):109–143, Junio 2000.

[CSHM19] Tinkle Chugh, Karthik Sindhya, Jussi Hakanen, and Kaisa Miettinen. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computing*, 23(9):3137–3166, 2019.

[DAPM00] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.

[DDB99] Nicole Drechsler, Rolf Drechsler, and Bernd Becker. Multi-objected optimization in evolutionary algorithms using Satisfyability Classes. In Bernd Reusch, editor, *Int. Conf. on Comput. Intelligence, Theory and Appl., 6th Fuzzy Days*, volume 1625 of *Lect. Notes in Comput. Sci.*, pages 108–117, Dortmund, Germany, 1999. Springer.

[DDB01] Nicole Drechsler, Rolf Drechsler, and Bernd Becker. Multi-objective optimisation based on relation *favour*. In *First Int. Conf. on Evol. Multi-Criterion Optim.*, volume 1993 of *Lect. Notes in Comput. Sci.*, pages 154–166. Springer, Berlin, 2001.

[De 95] Kenneth A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, Ann Arbor, 1995. Dissertation Abstracts International 36(10), 5140B; UMI 76-9381.

[Deb01] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley and Sons, Inc., Chichester, 2001.

[dFFG15] Alan de Freitas, Peter Fleming, and Frederico Guimarães. Aggregation trees for visualization and dimension reduction in many-objective optimization. *Inf Sci*, 298(Suppl C):288 – 314, 2015.

[DG89] Kalyanmoy Deb and David E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J. David Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 42–50, George Mason University, June 1989. Morgan Kaufmann.

[di 06] Francesco di Pierro. *Many-Objective Evolutionary Algorithms and Applications to Water Resources Engineering*. PhD thesis, School of Eng., Comput. Sci. and Math., UK, 2006.

[DJ14a] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based non-dominated sorting approach, part i: Solving problems with box constraints. *IEEE Trans Evol Comput*, 18(4):577–601, 2014.

[DJ14b] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based non-dominated sorting approach, part i: Solving problems with box constraints. *Evolutionary Computation, IEEE Transactions on*, 18:577–601, 08 2014.

[DLH20] Cai Dai, Xiujuan Lei, and Xiaoguang He. A decomposition-based evolutionary algorithm with adaptive weight adjustment for many-objective problems. *Soft Computing*, 24(14):10597–10609, 2020.

[DMC96] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[DMM03] Kalyanmoy Deb, Manikanth Mohan, and Shikhar Mishra. Towards a quick computation of well-spread Pareto-optimal solutions. In *Proc. of the 2nd Int. Conf. on Evol. Multi-Criterion Optim. EMO 2003*, volume 2632 of *Lect. Notes in Comput. Sci.*, pages 222–236. Springer, Berlin, 2003.

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr 2002. doi:http://dx.doi.org/10.1109/4235.996017.

[dPKS07] F. di Pierro, S. T. Khu, and D. A. Savic. An investigation on preference order ranking scheme for multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 11(1):17–45, Feb 2007. doi:10.1109/TEVC.2006.876362.

[DRH20] Kalyanmoy Deb, Proteek Chandan Roy, and Rayan Hussein. Surrogate modeling approaches for multi-objective optimization: Survey, taxonomy, and results. "", 2020.

[DS05] K. Deb and D.K. Saxena. On finding Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. Kangal rep. 2005011, Indian Inst. of Technol., 2005.

[DSP+20] Gaurav Dhiman, Mukesh Soni, Hari Mohan Pandey, Adam Slowik, and Harsimran Kaur. A novel hybrid hypervolume indicator and reference vector adaptation strategies based evolutionary algorithm for many-objective optimization. *Engineering with Computers*, pages 1–19, 2020.

[DTLZ02] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 1, pages 825–830, May 2002. doi:10.1109/CEC.2002.1007032.

[DTLZ05] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multiobjective optimization. In Lakhmi Jain, Xindong Wu, Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective Op-*

timization, Adv. Inf. and Knowl. Process., pages 105–145. Springer, Berlin, 2005. doi:10.1007/1-84628-137-7_6.

[EBN05] Michael Emmerich, Nicola Beume, and Boris Naujoks. An EMO algorithm using the Hypervolume measure as selection criterion. In *Proc. of the 3th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2005*, volume 3410 of *Lect. Notes in Comput. Sci.*, pages 62–76. Springer, Berlin, 2005. doi:10.1007/978-3-540-31880-4_5.

[EG02] Matthias Ehrgott and Xavier Gandibleux. Multiple Objective Combinatorial Optimization - A Tutorial, 2002. Presentación en Fourth International Conference on Multi-Objective Programming and Goal Programming MOPGP'02, http://www.esc.auckland.ac.nz/People/Staff/Matthias/.

[FA02] M. Farina and P. Amato. On the optimal solution definition for many-criteria optimization problems. In *2002 Annual Meeting of the North American Fuzzy Information Processing Society Proceedings. NAFIPS-FLINT 2002*, pages 233–238, 2002. doi:10.1109/NAFIPS.2002.1018061.

[FCC20] Jesús Guillermo Falcón-Cardona and Carlos A Coello Coello. Indicator-based multi-objective evolutionary algorithms: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 53(2):1–35, 2020.

[FF93a] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[FF93b] Carlos M. Fonseca and Peter J. Fleming. Multiobjective Genetic Algorithms. In *IEE Colloquium on Genetic Algorithms for Control Systems Engineering*, pages 6/1–6/5. IEE, 1993.

[FF94] Carlos M. Fonseca and Peter J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. Technical report, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U. K., 1994.

[FF95] Carlos M. Fonseca and Peter J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evol. Comput.*, 1995. doi:10.1162/evco.1995.3.1.1.

[Fog62] L. J. Fogel. Autonomous automata. *Industrial Research*, 4:14–19, 1962.

[Fog06] David B Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*, volume 1. John Wiley & Sons, 2006.

[Fou85] M. P. Fourman. Compaction of Symbolic Layout using Genetic Algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 141–153. Lawrence Erlbaum, 1985.

[FPL05] Peter Fleming, Robin C. Purshouse, and Robert J. Lygoe. Many-objective optimization: An engineering design perspective. In *Proc. of the 3th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2005*, volume 3410 of *Lect. Notes in Comput. Sci.*, pages 14–32, Berlin, 2005. Springer.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[Gol94] D. E. Goldberg. Genetic and evolutionary algorithms come of age. *Communications of the ACM*, 37(3):113–119, Marzo 1994.

[GPF14] Ioannis Giagkiozis, Robin C Purshouse, and Peter J Fleming. Generalized decomposition and cross entropy methods for many-objective optimization. *Information Sciences*, 282:363–387, 2014.

[HDD+19] Dong Han, Wenli Du, Wei Du, Yaochu Jin, and Chunping Wu. An adaptive decomposition-based evolutionary algorithm for many-objective optimization. *Information Sciences*, 491:204–222, 2019.

[HHH17] Kei Harada, Satoru Hiwa, and Tomoyuki Hiroyasu. Adaptive weight vector assignment method for moea/d. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–9. IEEE, 2017.

[HN93] Jeffrey Horn and Nicholas Nafpliotis. Multiobjective Optimization using the Niched Pareto Genetic Algorithm. Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[HNG94] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, Junio 1994. IEEE Service Center.

[Hol62] John H. Holland. Outline for a logical theory of adaptive systems. *Journal of the ACM*, 9(3):297–314, Julio 1962.

[Hol75] John H. Holland. *Adaptation in natural artificial systems*. University of Michigan Press, Ann Arbor, 1975.

[Hor97] Jeffrey Horn. *The Nature of Niching: Genetic Algorithms and the Evolution of Optimal, Cooperative Populations*. PhD thesis, University of Illinois at Urbana Champaign, Urbana, Illinois, 1997.

[HR12] David Hadka and Patrick Reed. Diagnostic assessment of search controls and failure modes in many-objective evolutionary optimization. *Evol. Comput.*, 2012. doi:10.1162/EVCO_a_00053.

[Hug03] E. J. Hughes. Multiple single objective pareto sampling. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 4, pages 2678–2684 Vol.4, Dec 2003. doi:10.1109/CEC.2003.1299427.

[Hug05] Evan J. Hughes. Evolutionary many-objective optimisation: many once or one many? In *2005 IEEE Congr. on Evol. Comput.*, 2005.

[Hug06] Evan J. Hughes. Multi-objective equivalent random search. *Parallel Probl. Solving from Nat. (PPSN IX)*, 4193:463–472, 2006.

[Hug07] Evan J. Hughes. MSOPS-II: A general-purpose many-objective optimiser. In *2007 IEEE Congr. on Evol. Comput.*, 2007. doi:10.1109/CEC.2007.4424985.

[HY16] Zhenan He and Gary G Yen. Visualization and performance metric in many-objective optimization. *IEEE Transactions on Evol Comput*, 20(3):386–402, 2016.

[IAN15] Hisao Ishibuchi, Naoya Akedo, and Yusuke Nojima. Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems. *IEEE Transactions on Evolutionary Computation*, 19(2):264–283, 2015.

[ID90] Alfred Inselberg and Bernard Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *Proc. of the First conf. on Visualization '90*, VIS '90, pages 361–378, Los Alamitos, 1990. IEEE Press.

[IDN17] Hisao Ishibuchi, Ken DOI, and Yusuke Nojima. On the effect of normalization in moea/d for multiobjective and manyobjective optimization. *Complex & Intell Syst*, 3(4):279–294, Dec 2017.

[IHN08] H. Ishibuchi, Y. Hitotsuyanagi, and Yusuke Nojima. Scalability of multiobjective genetic local search to many-objective problems: Knapsack problem case studies. In *2008 IEEE Congr. on Evol. Comput.*, 2008. doi:10.1109/CEC.2008.4631283.

[IM98] H. Ishibuchi and T. Murata. 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, 1998. doi:10.1109/5326.704576.

[IRMD16] Amin Ibrahim, Shahryar Rahnamayan, Miguel Vargas Martin, and Kalyanmoy Deb. 3D-RadVis: visualization of pareto front in many-objective optimization. In *IEEE CEC 2016*, pages 736–745. IEEE, 2016.

[IRMD18] Amin Ibrahim, Shahryar Rahnamayan, Miguel Vargas Martin, and Kalyanmoy Deb. 3d-radvis antenna: Visualization and performance measure for many-objective optimization. *Swarm and evolutionary computation*, 39:157–176, 2018.

[IS19] Hisao Ishibuchi and Hiroyuki Sato. Evolutionary many-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 614–661, 2019.

[ISTN09] H. Ishibuchi, Yuji Sakane, Noritaka Tsukamoto, and Yusuke Nojima. Adaptation of Scalarizing Funtions in MOEA/D: An Adaptive Scalarizing Funtion-Based Multiobjective Evolutionary Algorithm. In *Proc. of the 5th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2009*, volume 5467 of *Lect. Notes in Comput. Sci.*, pages 438–452. Springer, Berlin, 2009.

[ITN08] H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *IEEE CEC 2008*, pages 2419–2426, June 2008. doi:10.1109/CEC.2008.4631121.

[Jas02] Andrzej Jaszkiewicz. On the performance of multiple-objective genetic local search on the 0/1 Knapsack problem—A comparative experiment. *IEEE Trans. on Evol. Comput.*, 2002. doi:10.1109/TEVC.2002.802873.

[JYWL18] Shouyong Jiang, Shengxiang Yang, Yong Wang, and Xiaobin Liu. Scalarizing functions in decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 22(2):296–313, 2018.

[KB94] Sami Khuri and Thomas Bäck. An evolutionary heuristic for the maximum independent set problem. In *IEEECEP: Proceedings of The IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, 1994.

[KB07] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3):459–471, 2007.

[KBH94] Sami Khuri, Thomas Bäck, and Jörg Heitkötter. The zero/one multiple knapsack problem and genetic algorithms. In *Proceedings of the 1994 ACM Symposium on Applied Computing*, pages 188–193, New York, 1994. ASME Press.

[KBIvdH08] Johannes W. Kruisselbrink, Thomas Bäck, Ad P. IJzerman, and Eelke van der Horst. Evolutionary algorithms for automated drug design towards target molecule properties. In *Proc. of the 10th Ann. Conf. on Genetic and Evol. Comput.*, GECCO '08, pages 1555–1562. ACM, 2008.

[KE95] James Kennedy and Russell Eberhart. Particle swarm optimization (pso). In *Proc. IEEE International Conference on Neural Networks, Perth, Australia*, pages 1942–1948, 1995.

[KEB+09] Johannes W. Kruisselbrink, Michael T.M. Emmerich, Thomas Bäck, Andreas Bender, Ad P. IJzerman, and Eelke van der Horst. Combining Aggregation with Pareto Optimization: A Case Study in Evolutionary Molecular Design. In *Proc. of the 5th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2009*, volume 5467 of *Lect. Notes in Comput. Sci.*, pages 453–467. Springer, Berlin, 2009.

[Kei01] Maarten Keijzer. *Scientific Discovery using Genetic Programming*. PhD thesis, Technical University of Denmark, Denmark, 2001.

[KSR+20] Lev Kazakovtsev, Guzel Shkaberina, Ivan Rozhnov, Rui Li, and Vladimir Kazakovtsev. Genetic algorithms with the crossover-like mutation operator for the k-means problem. In Yury Kochetov, Igor Bykadorov, and Tatiana Gruzdeva, editors, *Mathematical Optimization Theory and Operations Research*, pages 350–362, Cham, 2020. Springer International Publishing.

[KVGN05] Mario Köppen, Raul Vicente-Garcia, and Betram Nickolay. Fuzzy-Pareto-dominance and its application in evolutionary multi-objective optimization. In *Proc. of the 3th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2005*, volume 3410 of *Lect. Notes in Comput. Sci.*, pages 399–412, Berlin, 2005. Springer.

[KY07a] Mario Köppen and Kaori Yoshida. Substitute distance assignments in NSGA-II for handling many-objective optimization problems. In *Proc. of the 4th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2007*, volume 4403 of *Lect. Notes in Comput. Sci.*, pages 727–741, Berlin, 2007. Springer.

[KY07b] Mario Köppen and Kaori Yoshida. Visualization of Pareto-sets in evolutionary multi-objective optimization. In *7th Int. Conf. HIS*, pages 156–161. IEEE, 2007.

[KY17] Padmavathi Kora and Priyanka Yadlapalli. Crossover operators in genetic algorithms: A review. *International Journal of Computer Applications*, 162(10), 2017.

[KYD03] V. Khare, X. Yao, and K. Deb. Performance scaling of multi-objective evolutionary algorithms. In *Proc. of the 2nd Int. Conf. on Evol. Multi-Criterion Optim. EMO 2003*, volume 2632 of *Lect. Notes in Comput. Sci.*, pages 376–390, Berlin, 2003. Springer.

[LCL+18] Longmei Li, Hao Chen, Jun Li, Ning Jing, and Michael Emmerich. Integrating region preferences in multiobjective evolutionary algorithms based on decomposition. In *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*, pages 379–384. IEEE, 2018.

[LEDE16] Xiaodong Li, Michael G Epitropakis, Kalyanmoy Deb, and Andries Engelbrecht. Seeking multiple solutions: an updated survey on niching methods and their applications. *IEEE Transactions on Evolutionary Computation*, 21(4):518–538, 2016.

[LGZ14] Hai-Lin Liu, Fangqing Gu, and Qingfu Zhang. Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE Trans Evol Comput*, 18(3):450–455, 2014.

[LJCC09] A. López Jaimes and Carlos Coello Coello. Study of preference relations in many-objective optimization. In *Proc. of the 11th Ann. Conf. on Genetic and Evol. Comput.*, GECCO '09, pages 611–618. ACM, 2009.

[LJCCAT11] Antonio López Jaimes, Carlos Coello Coello, Hernán Aguirre, and Kiyoshi Tanaka. Adaptive objective space partitioning using conflict information for many-objective optimization. In *Proc. of the 6th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2011*, volume 6576 of *Lect. Notes in Comput. Sci.*, pages 151–165. Springer, Berlin, 2011.

[LJCCC08] Antonio López Jaimes, Carlos Coello Coello, and Debrup Chakraborty. Objective reduction using a feature selection tech-

nique. In *Proc. of the 10th Ann. Conf. on Genetic and Evol. Comput.*, GECCO '08, pages 673–680. ACM, 2008.

[LJM+17] Qiuzhen Lin, Genmiao Jin, Yueping Ma, Ka-Chun Wong, Carlos A Coello Coello, Jianqiang Li, Jianyong Chen, and Jun Zhang. A diversity-enhanced resource allocation strategy for decomposition-based multiobjective evolutionary algorithm. *IEEE transactions on cybernetics*, 48(8):2388–2401, 2017.

[LLKH19] Jianchang Liu, Fei Li, Xiangyong Kong, and Peiqiu Huang. Handling many-objective optimisation problems with r2 indicator and decomposition-based particle swarm optimiser. *International Journal of Systems Science*, 50(2):320–336, 2019.

[LLTY15] Bingdong Li, Jinlong Li, Ke Tang, and Xin Yao. Many-objective evolutionary algorithms: A survey. *ACM Comput Surv*, 48(1):13:1–13:35, 2015.

[LRLL20] Ruochen Liu, Rui Ren, Jin Liu, and Jing Liu. A clustering and dimensionality reduction based evolutionary algorithm for large-scale multi-objective problems. *Applied Soft Computing*, 89:106120, 2020.

[LSS+17] Siew Mooi Lim, Abu Bakar Md Sultan, Md Nasir Sulaiman, Aida Mustapha, and Kuan Yew Leong. Crossover and mutation operators of genetic algorithms. *International journal of machine learning and computing*, 7(1):9–12, 2017.

[LYL+18] Jianping Luo, Yun Yang, Xia Li, Qiqi Liu, Minrong Chen, and Kaizhou Gao. A decomposition-based multi-objective evolutionary algorithm with quality indicator. *Swarm and evolutionary computation*, 39:339–355, 2018.

[LZ08] Hui Li and Qingfu Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE transactions on evolutionary computation*, 13(2):284–302, 2008.

[LZS+10] Miqing Li, Jinhua Zheng, Ruimin Shen, Ke Li, and Qizhao Yuan. A grid-based fitness strategy for evolutionary many-objective optimization. In *Proc. of the 12th Ann. Conf. on Genetic and Evol. Comput.*, GECCO '10, pages 463–470. ACM, 2010. doi:10.1145/1830483.1830570.

[LZY17] Miqing Li, Liangli Zhen, and Xin Yao. How to read many-objective solution sets in parallel coordinates. *IEEE Computational Intelligence Magazine*, 12(4):88–100, 2017.

[Mah95] Samir W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, University of Michigan, Urbana, IL, USA, 1995.

[Mah97] Samir W. Mahfoud. Niching methods. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, pages C6.1:1–4. Institute of Physics Publishing and Oxford University Press, Bristol, New York, 1997.

[MDL19] Seyedali Mirjalili, Jin Song Dong, and Andrew Lewis. *Nature-inspired optimizers: theories, literature reviews and applications*, volume 811. Springer, 2019.

[Mie99] K. Miettinen. *Nonlinear multiobjective optimization*. Springer, 1999.

[Mie01] Kaisa Miettien. Some methods for nonlinear multi-objetive optimization. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[Mie03] K. Miettinen. Graphical illustration of Pareto optimal solutions. *Multi-objective programming and goal programming: Theory and applications*, pages 197–202, 2003.

[MIG01] Tadahiko Murata, Hisao Ishibuchi, and Mitsuo Gen. Specification of genetic search directions in cellular multi-objective genetic algorithms. In *First Int. Conf. on Evol. Multi-Criterion Optim.*, volume 1993 of *Lect. Notes in Comput. Sci.*, pages 82–95. Springer, Berlin, 2001. doi:10.1007/3-540-44719-9_6.

[MKSOvLt20] Erich Alexander Michel Kolm, Montserrat Seall Ovelar, and Christian von Lücken (tutor). *Enfoques de Paralelización y Particionamiento de la Población aplicados a Algoritmos Evolutivos Multi-Objetivo basados en Descomposición*. FPUNA, Facultad Politécnica, 2020. Trabajo de fin de grado.

[MLZ+18] Xiaoliang Ma, Xiaodong Li, Qingfu Zhang, Ke Tang, Zhengping Liang, Weixin Xie, and Zexuan Zhu. A survey on cooperative co-evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 23(3):421–441, 2018.

[MMP02] P. Mitra, CA Murthy, and S.K. Pal. Unsupervised feature selection using feature similarity. *IEEE Trans. on pattern analysis and machine intelligence*, 24(3):301–312, 2002.

[MOBE18] J. Maltese, B. M. Ombuki-Berman, and A. P. Engelbrecht. A scalability study of many-objective optimization algorithms. *IEEE Trans Evol Comput*, 22(1):79–96, 2018.

[MSJ+16] Wali Khan Mashwani, Abdellah Salhi, M Jan, R Khanum, and M Suliaman. Evolutionary algorithms based on decomposition and indicator functions: state-of-the-art survey. *Advanced Computer Science and Applications (IJACSA)*, 7(2), 2016.

[MSW+20] Lianbo Ma, Mingli Shi, Rui Wang, ShengMinjie Chen, Junfei Zhao, and Xiaolong Shen. Multi/many-objective optimization via a new preference indicator. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–6. IEEE, 2020.

[MT09] Tadahiko Murata and Akinori Taki. Many-Objective Optimization for Knapsack Problems Using Correlation-Based Weighted Sum Approach. In *Proc. of the 5th Int. Conf. on Evol. Multi-Criterion*

Optim. EMO 2009, volume 5467 of *Lect. Notes in Comput. Sci.*, pages 468–480. Springer, Berlin, 2009.

[MT20] Suraj S Meghwani and Manoj Thakur. Adaptively weighted decomposition based multi-objective evolutionary algorithm. *Applied Intelligence*, pages 1–23, 2020.

[MZT+17] Xiaoliang Ma, Qingfu Zhang, Guangdong Tian, Junshan Yang, and Zexuan Zhu. On tchebycheff decomposition approaches for multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 22(2):226–244, 2017.

[NBE05] Boris Naujoks, Nicola Beume, and Michael Emmerich. Multi-objective optimization using S-metric selection: Application to three-dimensional solution spaces. In *2005 IEEE Congr. on Evol. Comput.*, 2005. doi:10.1109/CEC.2005.1554838.

[OEBY20] Sherin M Omran, Wessam H El-Behaidy, and Aliaa AA Youssif. Decomposition based multi-objectives evolutionary algorithms challenges and circumvention. In *Science and Information Conference*, pages 82–93. Springer, 2020.

[OS03] Shigeru Obayashi and Daisuke Sasaki. Visualization and data mining of Pareto solutions using self-organizing map. In *EMO 2003*, pages 796–809. Springer, 2003.

[PF01] R.C. Purshouse and P.J. Fleming. The Multi-Objective Genetic Algorithm Applied to Benchmark Problems—An Analysis. Technical Report 796, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK, Agosto 2001.

[PF03] Robin Purshouse and Peter J. Fleming. Conflict, harmony, and independence: Relationships in evolutionary multi-criterion optimisation. In *Proc. of the 2nd Int. Conf. on Evol. Multi-Criterion Optim. EMO 2003*, volume 2632 of *Lect. Notes in Comput. Sci.*, pages 16–30, Berlin, 2003. Springer.

[PG16] G. Pavai and T. V. Geetha. A survey on crossover operators. *ACM Comput. Surv.*, 49(4), December 2016. doi:10.1145/3009966.

[PMN07] Andy Pryke, Sanaz Mostaghim, and Alireza Nazemi. Heatmap visualization of population based multi objective algorithms. In *EMO 2007*, pages 361–375. Springer, 2007.

[PN15] Martin Pilat and Roman Neruda. Incorporating user preferences in moea/d through the coevolution of weights. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 727–734, 2015.

[PRC18] Miriam Pescador-Rojas and Carlos A Coello Coello. Collaborative and adaptive strategies of different scalarizing functions in moea/d. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2018.

[QML+14] Yutao Qi, Xiaoliang Ma, Fang Liu, Licheng Jiao, Jianyong Sun, and Jianshe Wu. Moea/d with adaptive weight adjustment. *Evolutionary computation*, 22(2):231–264, 2014.

[Ray15] Kumar S Ray. *Soft Computing and Its Applications, Volume One: A Unified Engineering Concept*, volume 1. CRC Press, 2015.

[Rec65] I. Rechenberg. Cybernetic solution path of an experimental problem. Technical report, Royal Air Force Establishment, Farnborough, UK, 1965.

[Rot06] Franz Rothlauf. *Representations for genetic and evolutionary algorithms*. Springer Berlin Heidelberg New York, 2006.

[Sai17] Nisha Saini. Review of selection methods in genetic algorithms. *International Journal of Engineering and Computer Science*, 6(12):22261–22263, 2017.

[SAT07] Hiroyuki Sato, Hernán E. Aguirre, and Kiyoshi Tanaka. Controlling dominance area of solutions and its impact on the performance of MOEAs. In *Proc. of the 4th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2007*, volume 4403 of *Lect. Notes in Comput. Sci.*, pages 5–20, Berlin, 2007. Springer.

[SAT10] H. Sato, Hernan E. Aguirre, and Kiyoshi Tanaka. Pareto partial dominance MOEA and hybrid archiving strategy included CDAS in many-objective optimization. In *2010 IEEE Congr. on Evol. Comput.*, 2010. doi:10.1109/CEC.2010.5586247.

[SAT12] Hiroyuki Sato, Hernán Aguirre, and Kiyoshi Tanaka. Variable space diversity, crossover and mutation in MOEA solving many-objective Knapsack problems. *Annals of Math. and Artif. Intell.*, pages 1–28, 2012. doi:10.1007/s10472-012-9293-y.

[Sch65] H. P. Schwefel. *Kybernetische Evolution als Strategie der Experimentellen Forschung in der Strungstechnik*. Hermann Fttinger Institut für Strungstechnik, Berlin, 1965.

[Sch84] J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.

[Sch85] J. David Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.

[SD93] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. Technical report, Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India, 1993.

[SD94] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.

[SD95] N. Srinivas and Kalyanmoy Deb. Comparative study of vector evaluated GA and NSGA applied to multiobjective optimization. In P. K. Roy and S. D. Mehta, editors, *Proceedings of the Symposium on Genetic Algorithms*, pages 83–90, 1995.

[SD07] Dhish Kumar Saxena and Kalyanmoy Deb. Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: Employing correntropy and a novel maximum variance unfolding. In *Proc. of the 4th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2007*, volume 4403 of *Lect. Notes in Comput. Sci.*, pages 772–787, Berlin, 2007. Springer.

[SDD07] André Sülflow, Nicole Drechsler, and Rolf Drechsler. Robust multi-objective optimization in high dimensional spaces. In *Proc. of the 4th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2007*, volume 4403 of *Lect. Notes in Comput. Sci.*, pages 715–726, Berlin, 2007. Springer.

[SG85] J. David Schaffer and John J. Grefenstette. Multiobjective Learning via Genetic Algorithms. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 593–595, Los Angeles, California, 1985. AAAI.

[SGCGG19] FL Sáez-Gutiérrez, FJF Cañavate, and A Guerrero-González. Review of industrial design optimization by genetic algorithms. In *Advances on Mechanics, Design Engineering and Manufacturing II*, pages 336–346. Springer, 2019.

[SI20] Ke Shang and Hisao Ishibuchi. A new hypervolume-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 2020.

[Sin19] Hemant Kumar Singh. Understanding hypervolume behavior theoretically for benchmarking in evolutionary multi/many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 24(3):603–610, 2019.

[SK98] Bruno Sareni and Laurent Krahenbuhl. Fitness sharing and methods revisited. *IEEE Transactions on Evolutionary Computation*, 2(3), Septiembre 1998.

[SK20] Adam Slowik and Halina Kwasnicka. Evolutionary algorithms and their applications to engineering problems. *Neural Computing and Applications*, pages 1–17, 2020.

[SLCC11] O. Schütze, A. Lara, and C. Coello Coello. On the influence of the number of objectives on the hardness of a multiobjective optimization problem. *IEEE Trans Evol Comput*, 15(4):444–455, Aug 2011. doi:10.1109/TEVC.2010.2064321.

[SMO01] D. Sasaki, M. Morikawa, and S. Obayashi. Aerodynamic shape optimization of supersonic wings by adaptive range multiobjective genetic algorithms. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 639–652, Germany, Marzo 2001. Springer-Verlag. Lecture Notes in Computer Science No. 1993.

[SPM15] Anupriya Shukla, Hari Mohan Pandey, and Deepti Mehrotra. Comparative review of selection techniques in genetic algorithm. In *2015*

International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), pages 515–519. IEEE, 2015.

[Sud20] Dirk Sudholt. The benefits of population diversity in evolutionary algorithms: a survey of rigorous runtime analyses. In *Theory of Evolutionary Computation*, pages 359–404. Springer, 2020.

[SZJYS11] Jiang Siwei, Cai Zhihua, Zhang Jie, and Ong Yew-Soon. Multiobjective optimization by decomposition with pareto-adaptive weight vectors. In *2011 Seventh International Conference on Natural Computation*, volume 3, pages 1260–1264. IEEE, 2011.

[TCZ+17] Ye Tian, Ran Cheng, Xingyi Zhang, Fan Cheng, and Yaochu Jin. An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility. *IEEE Transactions on Evolutionary Computation*, 22(4):609–622, 2017.

[Tey07] O. Teytaud. On the hardness of offline multi-objective optimization. *Evol Comput*, 15(4):475–491, December 2007.

[TF15] Tea Tušar and Bogdan Filipič. Visualization of pareto front approximations in evolutionary multiobjective optimization: A critical review and the prosection method. *IEEE Trans on Evol Comput*, 19(2):225–245, 2015.

[TSSG16] Anupam Trivedi, Dipti Srinivasan, Krishnendu Sanyal, and Abhiroop Ghosh. A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation*, 21(3):440–462, 2016.

[Tur95] Allan Turing. Computing machinery and intelligence. In George F. Luger, editor, *Computation and Intelligence:Collected Readings*. AAAI Press/The MIT Press, 1995. http://www.abelard.org/turpap/turpap.htm.

[TWFT95] Masahiro Tanaka, Hikaru Watanabe, Yasuyuka Furukawa, and Tetsuzo Tanino. GA-based decision support system for multicriteria optimization. In *Proc. of the Int. Conf. on Systems, Man, and Cybernetics*, volume 2, pages 1556–1561, Piscataway, NJ, 1995. IEEE.

[TZCJ16] Ye Tian, Xingyi Zhang, Ran Cheng, and Yaochu Jin. A multiobjective evolutionary algorithm based on an enhanced inverted generational distance metric. In *2016 IEEE Congress on Evol Comput*, pages 5222–5229. IEEE, 2016.

[US15] Anant J Umbarkar and Pranali D Sheth. Crossover operators in genetic algorithms: a review. *ICTACT journal on soft computing*, 6(1), 2015.

[vBB15] C. von Lücken, C. Brizuela, and B. Barán. Clustering based parallel many-objective evolutionary algorithms using the shape of the objective vectors. In *Evolutionary Multi-Criterion Optimization*, volume 9019 of *Lecture Notes in Computer Science*, pages 50–64. Springer International Publishing, 2015. doi:10.1007/978-3-319-15892-1_4.

[Vel99] David A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Inst. of Technol., Wright-Patterson AFB, Ohio, 1999.

[VL98] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Inst. of Technol., Wright-Patterson AFB, Ohio, 1998.

[vLBB14] Christian von Lücken, Benjamín Barán, and Carlos Brizuela. A survey on multi-objective evolutionary algorithms for many-objective problems. *Comput Optim Appl*, 58(3):707–756, 2014.

[vLBB19] Christian von Lücken, Carlos Brizuela, and Benjamin Barán. An overview on evolutionary algorithms for many-objective optimization problems. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 9(1):e1267, 2019.

[von03] Christian Daniel von Lücken. Algoritmos evolutivos para optimización multiobjetivo: Un estudio comparativo en un ambiente paralelo asíncrono. Master's thesis, Universidad Nacional de Asunción, Paraguay, 2003. (In Spanish).

[von16] Christian Daniel von Lücken. *Análisis del estado del arte de algoritmos evolutivos para problemas de muchos objetivos y una propuesta de agrupamiento para su paralelización*. PhD thesis, Universidad Nacional de Asunción, Paraguay, 2016. (In Spanish).

[vZL03] David A. van Veldhuizen, Jesse B. Zydallis, and Gary B. Lamont. Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 7(2):144–173, April 2003.

[WBN07] Tobias Wagner, Nicola Beume, and Boris Naujoks. Pareto-, Aggregation-, and Indicator-based methods in many-objective optimization. In *Proc. of the 4th Int. Conf. on Evol. Multi-Criterion Optim. EMO 2007*, volume 4403 of *Lect. Notes in Comput. Sci.*, pages 742–756, Berlin, 2007. Springer.

[WEDB19] Yali Wang, Michael Emmerich, André Deutz, and Thomas Bäck. Diversity-indicator based multi-objective evolutionary algorithm: Di-moea. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 346–358. Springer, 2019.

[WFE12] David Walker, Jonathan Fieldsend, and Richard Everson. Visualising many-objective populations. In *Genet Evol Comput Conf Companion '12*, pages 451–458. ACM, 2012.

[Whi93] Darrell Whitley. A genetic algorithm tutorial. Technical Report CS-93-103, Colorado State University, 1993.

[WLZ+18] Peng Wang, Bo Liao, Wen Zhu, Lijun Cai, Siqi Ren, Min Chen, Zejun Li, and Keqin Li. Adaptive region adjustment to improve

the balance of convergence and diversity in moea/d. *Applied Soft Computing*, 70:797–813, 2018.

[WMS19] Guohua Wu, Rammohan Mallipeddi, and Ponnuthurai Nagaratnam Suganthan. Ensemble strategies for population-based optimization algorithms – a survey. *Swarm and Evolutionary Computation*, 44:695 – 711, 2019. URL: http://www.sciencedirect.com/science/article/pii/S2210650217300445, doi:https://doi.org/10.1016/j.swevo.2018.08.015.

[WS06] Kilian Q. Weinberger and Lawrence K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *Int. J. Comput. Vision*, 70(1):77–90, 2006.

[WSL+20] Jia Wang, Yuchao Su, Qiuzhen Lin, Lijia Ma, Dunwei Gong, Jianqiang Li, and Zhong Ming. A survey of decomposition approaches in multiobjective evolutionary algorithms. *Neurocomputing*, 2020.

[WZGZ14] Zhenkun Wang, Qingfu Zhang, Maoguo Gong, and Aimin Zhou. A replacement strategy for balancing convergence and diversity in moea/d. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2132–2139. IEEE, 2014.

[WZZ+15] Luping Wang, Qingfu Zhang, Aimin Zhou, Maoguo Gong, and Licheng Jiao. Constrained subproblems in a decomposition-based multiobjective evolutionary algorithm. *IEEE Transactions on Evolutionary computation*, 20(3):475–480, 2015.

[XPPP06] Jian-Wu Xu, P.P. Pokharel, A.R.C. Paiva, and J.C. Principe. Nonlinear component analysis based on correntropy. In *International Joint Conference on Neural Networks, 2006. IJCNN '06.*, pages 1851 – 1855, 2006.

[XXL19] Minghui Xiong, Wei Xiong, and Chengxiang Liu. A hybrid many-objective evolutionary algorithm with region preference for decision makers. *IEEE Access*, 7:117699–117715, 2019.

[XXM20] Qian Xu, Zhanqi Xu, and Tao Ma. A survey of multiobjective evolutionary algorithms based on decomposition: Variants, challenges and future directions. *IEEE Access*, 8:41588–41614, 2020.

[YXW+15] Yuan Yuan, Hua Xu, Bo Wang, Bo Zhang, and Xin Yao. Balancing convergence and diversity in decomposition-based many-objective optimizers. *IEEE Transactions on Evolutionary Computation*, 20(2):180–198, 2015.

[ZDT99] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Technical Report 70, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, December 1999.

[ZDT00] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 2000. doi:10.1162/106365600568202.

[ZDT+01] Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Cornde, editors. *Proceedings of the First International conference on EMOO, 2001*, Berlin, Germany, Marzo 2001. Springer-Verlag.

[ZK04] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *Parallel Probl. Solving from Nat. (PPSN VIII)*, volume 3242 of *Lect. Notes in Comput. Sci.*, pages 832–842, Berlin, 2004. Springer.

[ZL07] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, Dec 2007. doi:10.1109/TEVC.2007.892759.

[ZL08] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 11:712 – 731, 01 2008. doi:10.1109/TEVC.2007.892759.

[ZLT01a] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, Mayo 2001.

[ZLT01b] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Comput. Eng. and Netw. Lab., Swiss Fed. Inst. of Technol. Zurich, 2001.

[ZT98] Eckart Zitzler and Lothar Thiele. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, Mayo 1998.

[ZT99a] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto Approach. *IEEE Trans. on Evol. Comput.*, 1999. doi:10.1109/4235.797969.

[ZT99b] Eckart Zitzler and Lothar Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Noviembre 1999.

[ZTL+03] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Trans. on Evol. Comput.*, 2003. doi:10.1109/TEVC.2003.810758.

[ZX17] Jiawei Zhang and Lining Xing. A survey of multiobjective evolutionary algorithms. In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE Interna-*

tional Conference on Embedded and Ubiquitous Computing (EUC), volume 1, pages 93–100. IEEE, 2017.

[ZYZJ15] Ying Zhang, Rennong Yang, Jialiang Zuo, and Xiaoning Jing. Enhancing moea/d with uniform population initialization, weight vector design and adjustment using uniform design. *Journal of Systems Engineering and Electronics*, 26(5):1010–1022, 2015.

Índice Alfabético