UniversidaddeValladolid

Máster en Ingeniería Industrial

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

TRABAJO FIN DE MÁSTER

ESTUDIO DE ALGORITMOS GENÉTICOS MULTIOBJETIVO

Autor: D. Joseph Álvarez Pedrosa
Tutora: Dª María Elena Pérez Vázquez

Valladolid, septiembre 2020

UniversidaddeValladolid

Máster en Ingeniería Industrial

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

TRABAJO FIN DE MÁSTER

ESTUDIO DE ALGORITMOS GENÉTICOS MULTIOBJETIVO

Autor: D. Joseph Álvarez Pedrosa
Tutora: Dª María Elena Pérez Vázquez

Valladolid, septiembre 2020

RESUMEN

En muchas ocasiones, a lo largo de nuestra experiencia, nos encontramos ante problemas de decisión donde los objetivos a satisfacer son conflictivos entre sí: desde la decisión de comprar un coche económico y confortable, a la configuración de productos financieros que generen el máximo rendimiento con el menor riesgo. La mayoría de estos problemas se resuelven gracias al uso del intelecto, la experiencia acumulada y la intuición del que toma la decisión final. La metaheurística propone un marco normativo para implementar técnicas que ayuden a resolver estos problemas, muchas de ellas basadas en patrones similares a los que acontecen en la naturaleza para que las especies prosperen. Aquí se enmarcan los algoritmos genéticos, donde una población de individuos prospera ante las constantes evaluaciones de aptitud a las que son sometidos, formando un conjunto diverso de soluciones óptimas, dejando en manos del experto la decisión final.

El objeto del presente trabajo es hacer un estudio de las técnicas metaheurísticas presentes en la actualidad, haciendo especial énfasis en los algoritmos genéticos multiobjetivo, para después seleccionar una de estas técnicas como rival ante dos grandes del mercado. El problema a tratar es de tipo estructural, objeto de estudio actualmente en la universidad, y consiste en la sintonización de amortiguadores de masa (TMD) en un edificio esbelto para satisfacer dos objetivos conflictivos entre sí: seguridad estructural y confort de los habitantes ante la acción sísmica.

Palabras clave: *Metaheurística, Multiobjetivo, Dominancia Pareto, MOP.*

ABSTRACT

On many occasions, we are faced with problems in decision-making, where the objectives to be met are conflicting with each other: from deciding to buy a car which is both economical and comfortable, to generating maximum performance with the least risk in outperforming financial products. Most of these problems are solved by decision-makers that use their intellectual skills, accumulated experience and intuition. Metaheuristics proposes a high-level framework to implement techniques that help solve these problems, many of them based on patterns similar to those that occur in nature for species to thrive. Genetic algorithms are framed within these guidelines, where a group of individuals thrives and is subjected to a constant evaluation, resulting in a diverse set of optimal solutions, leaving the final decision up to the expert.

The purpose of this work is to carry out a research study on metaheuristic techniques in use today, with a special emphasis on multi-objective genetic algorithms. One of these techniques is selected to compete with two of the top market algorithms. The problem to be solved is of a structural nature and is currently under study at the university. It consists in the tuning of mass dampers (TMD) in slender buildings to meet two conflicting objectives between them: structural safety and comfort of the inhabitants during a seismic action.

Keywords: *Metaheuristics, Multiobjective, Pareto Dominance, MOP.*

Índice

Índice de Tablas

Índice de Figuras

1. INTRODUCCIÓN A LAS TÉCNICAS METAHEURÍSTICAS MULTIOBJETIVO

1.1. Metaheurística – Antecedentes y Definición

Los algoritmos de optimización pueden ser divididos en dos grandes categorías: los exactos o deterministas y los heurísticos. En los primeros se garantiza que se va a encontrar el óptimo en una cantidad finita de tiempo, mientras que los heurísticos no tienen estas garantías, por lo que generalmente devuelven soluciones peores al óptimo.

Precisamente por esta virtud de encontrar el óptimo, los algoritmos exactos deben examinar exhaustivamente cada solución en el espacio de búsqueda, salvo que puedan probar de forma explícita que no necesitan hacerlo, favoreciendo que sean poco eficientes ante problemas donde el tiempo de computación se incrementa con la magnitud del problema e incluso habiendo problemas de decisión reales de gran interés para la ciencia (de dificultad NP) que directamente son inabordables por estos algoritmos, haciendo prácticos a los algoritmos heurísticos por su aptitud para encontrar una gama de soluciones cercanas al óptimo en fracciones de tiempo de computación muy inferiores.

En un nivel superior a la heurística se enmarca el término *metaheurística*, acuñado por primera vez en (Glover, 1986), aunque su definición concreta fue eludida. En (Sörensen & Glover, 2013) se ha definido como sigue:

"Una metaheurística es un marco algorítmico de alto nivel e independiente del problema, que proporciona un conjunto de pautas o estrategias para desarrollar algoritmos de optimización heurística. El término también se utiliza para referirse a la implementación específica del problema de un algoritmo de optimización heurístico que sigue las pautas expresadas en dicho marco."

Por lo que, la metaheurística, en términos generales, más que un algoritmo es un conjunto de ideas, conceptos y operadores, que se utilizan para diseñar algoritmos de optimización heurísticos. La justificación de la acuñación del término viene de la necesidad de definir un marco normativo independiente del problema, debido al vínculo que existe tradicionalmente de las reglas heurísticas con el problema a resolver.

1.2. Métodos clásicos en la metaheurística

Una metaheurística será efectiva si existe un equilibrio entre sus capacidades de *exploración* de nuevos resultados en el espacio de búsqueda y la *explotación* de la experiencia acumulada, por lo que las diferencias entre métodos vendrán dadas por la forma en que tratan de lograr este balance.

Desde los comienzos de la metaheurística hace más de tres décadas, multitud de autores han publicado (y continúan publicando) nuevos marcos metaheurísticos. Es por ello por lo que elaborar una clasificación rigurosa de la metaheurística es una empresa arriesgada, principalmente por esta avalancha de nuevos métodos metaheurísticos desarrollados en la teoría combinatoria durante los últimos años.

La clasificación más común diferencia si la exploración está basada en poblaciones o en trayectorias, así como si la forma de explotar la experiencia acumulada se inspira en fenómenos de la naturaleza o no. Una clasificación más reciente es la propuesta en (Abdel-Basset et al., 2018) en su taxonomía de la metaheurística, en donde diferencia dos principales vertientes: metaheurísticas basadas en metáforas o no basadas en metáforas.

En este capítulo se van a definir las principales ramas de la metaheurística, a partir de las cuales se basan estos métodos más recientes, para después dar paso al problema multiobjetivo (MOP) y las técnicas metaheurísticas desarrolladas hasta la fecha que han sido implementadas con éxito en una gama de problemas que sirven para testear su eficacia y obtener métricas de rendimiento entre los distintos algoritmos a la hora de abordar dichos problemas multiobjetivo. Posteriormente, se abordará un

problema multiobjetivo con una selección de algoritmos genéticos y se compararán los resultados obtenidos según las métricas establecidas en este trabajo.

1.2.1. Simulated Annealing (SA) – Simulación del Recocido

Creado por Metropolis en 1953, se introdujo como método de optimización en los comienzos de la computación. Su principio se basa en el recocido de los materiales, donde tenemos una colección de átomos en equilibrio a una temperatura dada. Fue utilizado con éxito en el diseño de ordenadores (Kirkpatrick et al., 1983).

En cada paso del algoritmo se produce un pequeño desplazamiento aleatorio en un *átomo* y el estado resultante se caracteriza por la *variación de energía del sistema* ΔE, de forma que si $\Delta E \leq 0$, se acepta el desplazamiento y la configuración resultante es la que se utiliza como punto de partida en la siguiente iteración. El caso donde $\Delta E > 0$ se trata de forma probabilística, tal que la probabilidad de que el cambio sea aceptado es $P(\Delta E) = \exp(\Delta E / k_B T)$. Así, se crea un conjunto aleatorio uniformemente distribuido en el intervalo $(0, 1)$ y se selecciona un elemento $P(x)$. Si el elemento seleccionado $P(x) < P(\Delta E)$, entonces se acepta la nueva configuración. En otro caso, se utiliza la configuración original en el siguiente paso.

Repitiendo estos pasos muchas veces obtenemos una simulación del movimiento de átomos en contacto térmico con una fuente de calor a una temperatura T. La función $P(\Delta E)$ hace que el sistema evolucione conforme a la distribución de Boltzmann.

Utilizando una función de coste en lugar de una función de energía y definiendo configuraciones de estados mediante un conjunto de parámetros del problema $\{x_i\}$, se genera directamente una población de configuraciones de un problema de optimización a una *temperatura* dada. Esta temperatura será un parámetro de control en las mismas unidades que la función de coste.

El proceso consiste en "fundir" el sistema a temperaturas elevadas, donde casi cualquier cambio es posible, y después disminuir la temperatura a estados de energía inferiores hasta que el sistema "cristaliza" y ya no es posible la realización de cambios a partir de este momento. A cada temperatura, la simulación debe avanzar lo suficiente para que el sistema alcance el estado de equilibrio. La secuencia de temperaturas y el número de configuraciones obtenidas de las $\{x_i\}$ variables para lograr el equilibrio a cada temperatura definen el recocido simulado.

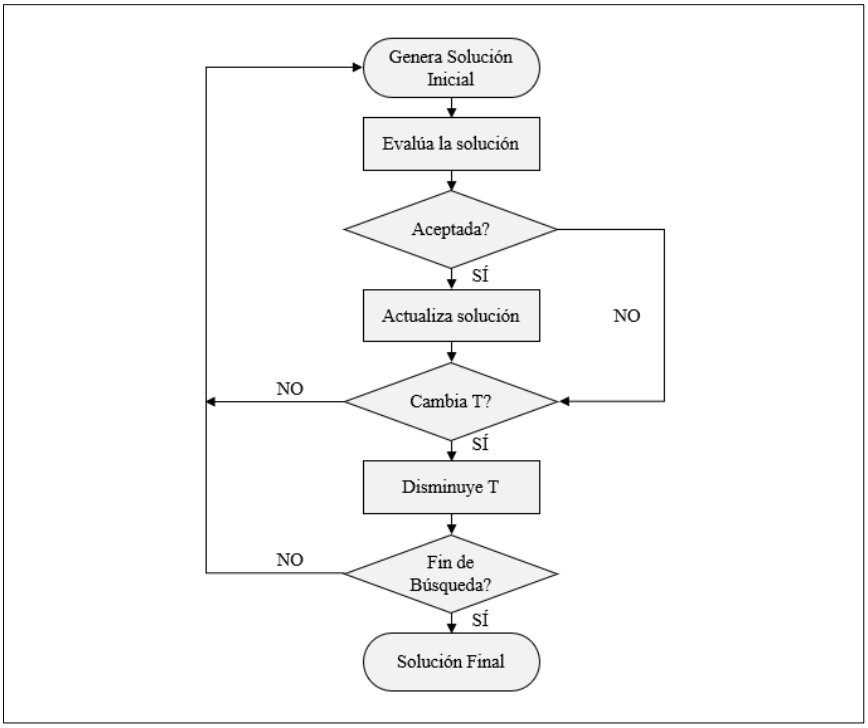

Figura 1.1. Diagrama de Flujo SA. Fuente (Pham & Karaboga, 2012)

1.2.2. Tabu Search (TS) – Búsqueda Tabú

Metaheurística propuesta por Glover en 1986 para resolver problemas de optimización en la teoría combinacional. Se basa en la búsqueda iterativa y en recordar algunos aspectos del proceso de búsqueda en el espacio de soluciones para su uso en la siguiente iteración, creando una lista tabú de cambios ya realizados que no se podrán realizar en un futuro.

Estos algoritmos se inicializan con una configuración determinada. En cada iteración, se define la estructura del vecindario para la configuración actual. A continuación, se realiza un movimiento hacia la mejor configuración en este vecindario (por ejemplo, en un problema de minimización, el movimiento será hacia valores del coste más pequeños). Normalmente solo evalúa a los vecinos más prometedores, evitando que el problema pueda llegar a ser intratable. La comunidad de vecinos se actualiza de forma dinámica, y se permiten transiciones a configuraciones con mayor coste (consiguiendo salir de los mínimos locales). La cualidad esencial de este tipo de algoritmos es la exclusión directa de alternativas de búsqueda clasificadas temporalmente como tabú (uso de memoria). Por último, posee dos mecanismos fundamentales: intensificación y diversificación. Con el primero, el algoritmo realiza una exploración comprehensiva de regiones de interés que pueden llevar a un óptimo local. Con la diversificación, la búsqueda se lleva a cabo en regiones que permanecieron sin visitar previamente, evitando mínimos locales (Monticelli et al., 2007).

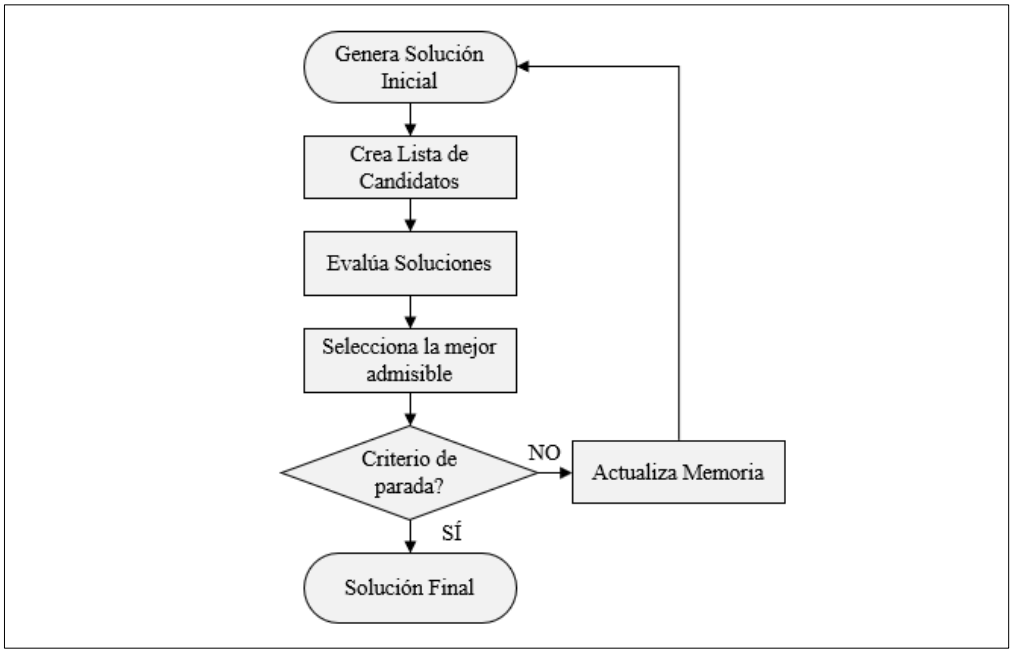

Figura 1.2. Diagrama de Flujo TS. Fuente (Pham & Karaboga, 2012)

1.2.3. Genetic Algorithms (GA) – Algoritmos Genéticos

Para satisfacer las necesidades de resolver problemas de optimización multimodales con funciones no lineales, surgen los algoritmos genéticos. En estos casos puede ser necesario un proceso de búsqueda aleatoria. Sin embargo, las técnicas de búsqueda aleatoria sin una guía son extremadamente ineficientes cuando tratamos con grandes cantidades de datos.

Un algoritmo genético (GA) es una técnica de búsqueda aleatoria guiada, inventada por Holland en 1975, capaz de encontrar el óptimo global en espacios de búsqueda multidimensionales complejos. Se modela según la evolución natural, en la cual los operadores que utiliza se inspiran en procesos de evolución. Estos operadores, conocidos como *operadores genéticos*, manipulan a los individuos en la población durante varias generaciones para mejorar su *fitness* (aptitud) de forma gradual. Los individuos en la población actúan como cromosomas, y normalmente se representan como cadenas de dígitos.

La evolución de la población se gobierna bajo el *Teorema del Esquema* (o teorema fundamental de los algoritmos genéticos), donde dicho esquema representa un conjunto de individuos, por ejemplo, un subconjunto de la población, en términos de similitud de dígitos en ciertas posiciones de estos

individuos. Así, un *esquema* se caracteriza por dos parámetros: *longitud característica* y *orden*. La longitud característica es la distancia entre el primer y el último dígito con valores fijos. El orden de un esquema es el número de dígitos con valores especificados. Según el teorema, la distribución de un esquema a través de la población de una generación a otra depende de su orden, longitud característica y fitness.

Los algoritmos genéticos no utilizan mucho el conocimiento sobre el problema a optimizar ni tratan directamente con los parámetros del problema, sino que trabajan con códigos que representan a estos parámetros. Luego, el primer paso para aplicar el GA es saber cómo codificar el problema de optimización, es decir, cómo representar los parámetros del problema. Dado que los GAs trabajan con poblaciones de soluciones posibles, el segundo paso es saber cómo se va a crear la población inicial. El tercer paso es cómo seleccionar o idear un conjunto adecuado de operadores genéticos. Por último, se debe conocer la calidad de las soluciones encontradas para mejorarlas.

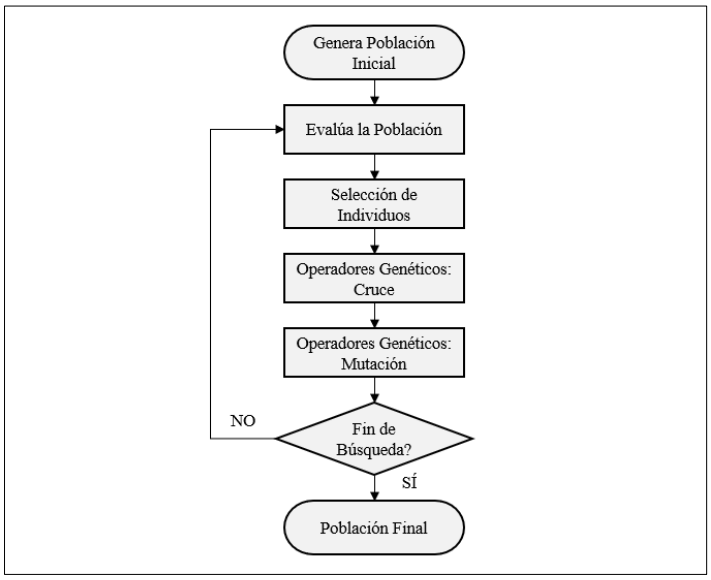

Figura 1.3. Diagrama de Flujo de un algoritmo genético. Fuente (Pham & Karaboga, 2012)

1.3. Problemas de Optimización Multi-Objetivo

Muchas situaciones y problemas a la hora de tomar decisiones en el mundo real son de tipo multiobjetivo, donde no existe un único criterio por el cual el éxito de una solución particular puede ser medida, sino que hay múltiples criterios que deben satisfacerse. Además, normalmente estos objetivos son conflictivos entre sí. En estos casos, no existe una solución ideal en la cual se satisfagan todos los criterios, por lo que habrá una solución de compromiso acorde a las preferencias del diseñador. Los procesos matemáticos para la búsqueda de tales soluciones se conocen como algoritmos multiobjetivo.

En los comienzos, el enfoque para resolver problemas de optimización multiobjetivo (en adelante MOP), estaba basado principalmente en un sistema preferencias, donde el MOP se convertía en una función objetivo escalar mediante un vector de ponderación. Aquí encontramos métodos clásicos como el método de sumas ponderadas, donde se puede escalar un conjunto de objetivos mediante la definición de este vector de ponderación por parte del diseñador para obtener un problema de un objetivo (Biswas & Suganthan, 2018).

Sin embargo, la necesidad de encontrar todo un conjunto de soluciones motivó a los investigadores a hondar en esta problemática, con el consecuente diseño de técnicas de programación multiobjetivo para tratar esta situación en donde se manejan grandes cantidades de variables de todo tipo, no lineales, con funciones complejas y en condiciones de contorno no estándar. Este tipo de factores de complejidad son especialmente relevantes en problemas de ingeniería, donde los algoritmos de optimización para un solo objetivo se han visto limitados (Jones et al., 2002).

La optimización multiobjetivo implica la optimización simultánea de más de una función objetivo. Cuando el problema no es trivial, no existe una solución que optimice simultáneamente los objetivos,

sino que son conflictivos entre sí. En este caso, existen múltiples soluciones óptimas (posiblemente infinitas) denominadas *óptimos de Pareto*.

Una solución es un óptimo de Pareto si no es posible mejorar uno de los objetivos sin degradar el valor del resto. Sin información subjetiva adicional, todas las soluciones pertenecientes a los óptimos de Pareto son consideradas igual de buenas. El objetivo es, por tanto, encontrar un conjunto representativo de soluciones óptimas o cuantificar las compensaciones necesarias para satisfacer los distintos objetivos, o directamente encontrar una solución única que satisfaga las preferencias subjetivas del diseñador.

Puede ocurrir que las funciones objetivo formen un problema multimodal Figura 1.4. En estos problemas pueden existir múltiples soluciones óptimas con la misma calidad o una solución global con múltiples óptimos locales (Khosrowpour, 2008). Una manera de testear la eficacia de los algoritmos multiobjetivo es enfrentarlos a problemas de este tipo, en donde se pueden analizar factores como el estancamiento en óptimos locales, convergencia hacia una zona concreta del espacio de búsqueda, diversidad en la población de soluciones o relación entre exploración y explotación de resultados.

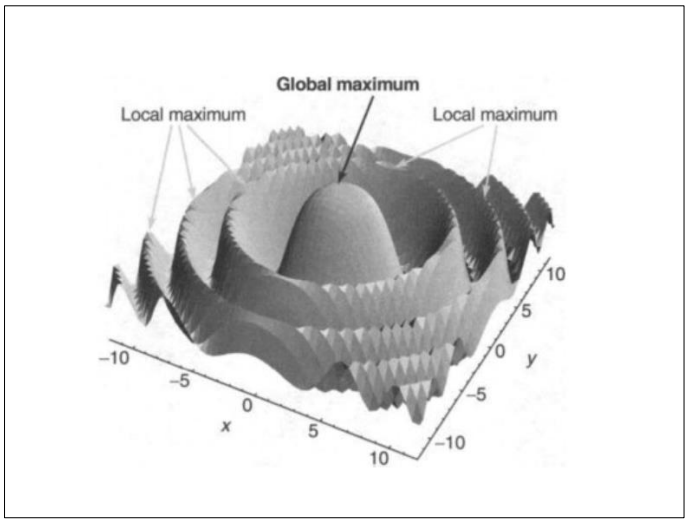

Figura 1.4. Espacio de búsqueda multimodal. Fuente (Jacob, 2001)

1.3.1. Técnicas de optimización para MOPs

Para abordar el MOP, las primeras técnicas de optimización multiobjetivo trataban de obtener óptimos de Pareto de uno en uno, siendo además especialmente sensibles a la forma del frente: no son adecuados para frentes discontinuos o de formas cóncavas. Las metaheurísticas, en cambio, resultaron particularmente adecuadas para resolver MOPs porque son menos susceptibles a la forma y continuidad de la frontera Pareto, por lo que son aptos para manejar fronteras discontinuas o no convexas (se ahondará en este punto en la Sección 1.3.5).

Metaheurísticas como Simulated Annealing (SA), visto en la Sección 1.2.1, a pesar de estar originalmente diseñadas para la optimización combinatoria, se ha extendido su uso para tratar múltiples objetivos. La clave de esta extensión reside en determinar cómo se computa la probabilidad de aceptar a un individuo $\boldsymbol{y}$ cuando $\boldsymbol{f}(\boldsymbol{y})$ es dominada con respecto a $\boldsymbol{f}(\boldsymbol{x})$ (Coello et al., 2007).

Lo mismo ocurre con Tabu Search (TS), visto en la Sección 1.2.2. La idea básica es crear un subconjunto cuyos elementos se denominan tabú. La pertenencia a este subconjunto se les confiere mediante una lista de movimientos históricos previamente catalogados como improductivos. Así, TS tiende a generar movimientos en el área circundante a la solución candidata. Luego, el principal problema cuando se quiere extender esta técnica al tratamiento multiobjetivo está en cómo mantener la diversidad para poder generar una frontera de Pareto.

Actualmente, las metaheurísticas para tratar MOPs (algoritmos evolucionarios, algoritmos de optimización por enjambre de partículas, colonias de hormigas, etc.) se basan en poblaciones, de forma que son capaces de generar varios óptimos de Pareto en una ejecución. En la Figura 1.5 se muestra una clasificación de estas técnicas.

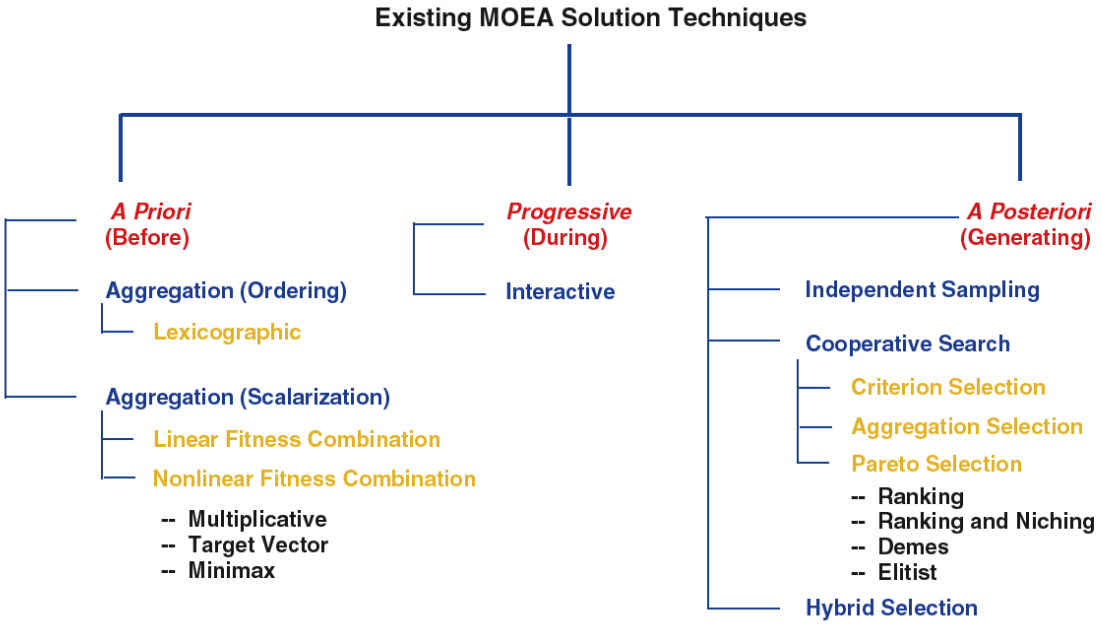

Figura 1.5. Técnicas de optimización multiobjetivo, clasificación. Fuente (Coello et al., 2007)

1.3.2. Formulación del problema multiobjetivo

Un problema de optimización multiobjetivo tiene un número determinado de funciones objetivo las cuales deben ser minimizadas o maximizadas (Deb, 2001b). También tiene una serie de restricciones, las cuales deben ser cumplidas por cualquier solución factible. En su forma general, siguiendo la nomenclatura propuesta por Deb, un problema MOP se puede representar como:

$$\begin{aligned} &Minimizar/Maximizar && f_m(\boldsymbol{x}), && m = 1,2,\dots,M; \\ &Sujeto\ a && g_j(\boldsymbol{x}) \geq 0 && j = 1,2,\dots,J; \\ & && h_k(\boldsymbol{x}) = 0 && k = 1,2,\dots,K; \\ & && x_i^{(L)} \leq x_i \leq x_i^{(U)} && i = 1,2,\dots,n\,. \end{aligned} \tag{1.1}$$

Una solución $\boldsymbol{x}$ es un vector de n variables de decisión: $\boldsymbol{x} = (x_1, x_2, \dots, x_n)^T$. El último conjunto de restricciones en la representación $x_i^{(L)}$ y $x_i^{(U)}$son los límites inferior y superior de la variable. Estos límites definen lo que se denomina el *espacio de decisión*, D. Asociadas al problema tenemos J restricciones tipo inecuación y K restricciones de igualdad. Los términos $g_j(\boldsymbol{x})$ y $h_k(\boldsymbol{x})$ son las denominadas *funciones de restricción*. Si una solución satisface todas las $(J + K)$ restricciones y está dentro de los límites, entonces esta es una solución *factible.* Al conjunto de soluciones factibles se le denomina *espacio de búsqueda* Ω.

Dado que hay M funciones objetivo $\boldsymbol{f}(\boldsymbol{x}) = \big(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \dots, f_M(\boldsymbol{x})\big)^T$consideradas en el problema, cada función puede tener el objeto de ser maximizada o minimizada. Aplicando el principio de dualidad (Deb, 1995) en el contexto de la optimización podemos convertir un problema de maximización en uno de minimización multiplicando la función objetivo por -1.

La principal diferencia a la hora de abordar problemas MOP con respecto a los de un solo objetivo es que, en MOPs, las funciones objetivo constituyen un espacio multidimensional (con tantas dimensiones como objetivos se quieran optimizar), además del conocido espacio de búsqueda. Este espacio se denomina *espacio objetivo*, Z. Para cada solución $\boldsymbol{x}$ en el espacio de decisión, existe un punto en el espacio objetivo $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{z} = (z_1,\ z_2, \dots, z_M)^T$.

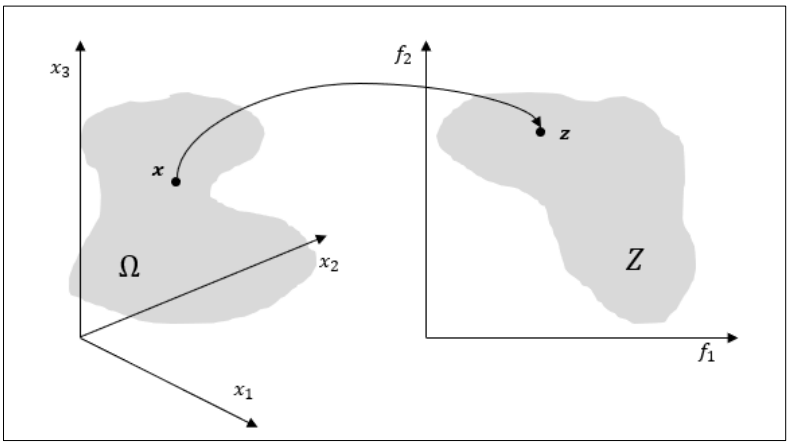

Figura 1.6. Representación del espacio objetivo y el espacio de búsqueda. Fuente (Deb, 2001b)

1.3.3. Optimización Multiobjetivo

Para poder definir cómo trabajan los algoritmos multiobjetivo es necesario primero visualizar de nuevo el espacio objetivo e identificar en él los diversos puntos (vectores) estratégicos para la búsqueda de soluciones en estos algoritmos.

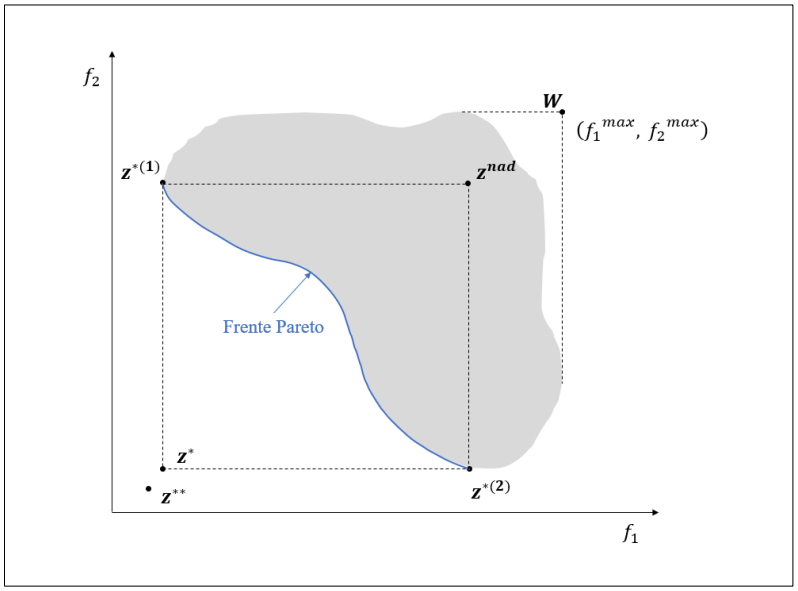

Figura 1.7. Frente de Pareto y Vectores característicos: ideal, utópico y nadir. Fuente (Deb, 2001b)

1.3.3.1. Vector Objetivo Ideal

Para cada uno de los *M* objetivos existe una solución óptima diferenciada. Un vector objetivo construido con estos valores objetivos óptimos constituye el *vector objetivo ideal* (Deb, 2001b)*:*

$$\mathbf{z}^* = \mathbf{f}^* = (f_1^*, f_2^*, \dots, f_M^*)^{\mathrm{T}} \qquad (1.2)$$

Para los problemas que se van a abordar con los algoritmos multiobjetivo esta solución es ideal, una solución no existente. Solo puede existir cuando todos los objetivos se minimizan con la misma solución, por lo que los objetivos no son conflictivos entre sí.

La utilidad de este vector es como punto de referencia: cuanto más se acerque una solución a ésta en el espacio de decisión, mejor. También es utilizado para normalizar las soluciones en un rango común, donde es necesario conocer el límite inferior de cada función objetivo.

La Figura 1.7 muestra un vector objetivo ideal.

1.3.3.2. Vector Objetivo Utópico

El vector objetivo ideal es un array formado por los límites inferiores de las funciones objetivo. Algunos algoritmos requieren una solución de comparación cuyo valor objetivo sea estrictamente superior a la de cualquier solución en el espacio de búsqueda. Así, el *vector objetivo utópico*, $\mathbf{z}^{**}$, tiene sus componentes ligeramente inferiores a los del vector objetivo ideal:

$$\begin{gathered} z_i^{**} = z_i^* - \epsilon_i \\ \text{con } \epsilon_i > 0 \text{ para todo } i = 1, 2, \ldots, M \end{gathered} \tag{1.3}$$

La Figura 1.7 muestra un vector objetivo utópico.

1.3.3.3. Vector Objetivo Nadir

A diferencia del vector objetivo ideal que representa el límite inferior de cada objetivo en el espacio de soluciones factibles *S* (espacio de búsqueda), el *vector objetivo Nadir*, $\boldsymbol{z^{nad}}$, representa el límite superior de cada objetivo en el conjunto de óptimos de Pareto, y no en el espacio de búsqueda. No se debe confundir el vector nadir con el vector de máximos $\boldsymbol{W}$.

Este vector se obtiene a partir del vector objetivo ideal. Así, para dos objetivos, si:

$$\begin{aligned} \boldsymbol{z}^{*(1)} &= \left(f_1\left(\boldsymbol{x}^{*(1)}\right), f_2\left(\boldsymbol{x}^{*(1)}\right)\right)^T \\ \boldsymbol{z}^{*(2)} &= \left(f_1\left(\boldsymbol{x}^{*(2)}\right), f_2\left(\boldsymbol{x}^{*(2)}\right)\right)^T \end{aligned} \tag{1.4}$$

son las coordenadas de los mínimos de f_1 y f_2 en el espacio objetivo, entonces vector objetivo nadir se puede estimar como:

$$\boldsymbol{z^{nad}} = \left(f_1\left(\boldsymbol{x}^{*(2)}\right), f_2\left(\boldsymbol{x}^{*(1)}\right)\right)^T \tag{1.5}$$

El vector nadir puede representar una solución existente o no, dependiendo de la concavidad y continuidad del conjunto de soluciones Pareto. Para normalizar cada objetivo en el rango de óptimos de Pareto, se utilizan los vectores ideales objetivo y nadir como sigue:

$$f_i^{norm} = \frac{f_i - z_i^*}{z_i^{nad} - z_i^*} \tag{1.6}$$

1.3.4. Dominancia y Frente de Pareto

La gran parte de los algoritmos multiobjetivo utilizan el concepto de *dominancia* en su búsqueda. En un problema de dos objetivos, tomando dos soluciones cualesquiera del espacio de decisión y comparándolas, podemos diferenciar dos casos:

- Una solución mejora a los dos objetivos con respecto a la otra.
- Una solución mejora un objetivo, pero empeora el otro con respecto a la otra.

Considerando dos vectores objetivo $\boldsymbol{x}^{(1)}$ y $\boldsymbol{x}^{(2)}$, si todos los componentes de $\boldsymbol{x}^{(2)}$ son menores o iguales a los componentes de $\boldsymbol{x}^{(1)}$, con al menos un elemento estrictamente menor, entonces el vector $\boldsymbol{x}^{(2)}$ se corresponde con una solución mejor que $\boldsymbol{x}^{(1)}$. En este caso, se dice que $\boldsymbol{x}^{(2)}$ domina a $\boldsymbol{x}^{(1)}$. De una manera formal se puede escribir como $\boldsymbol{x}^{(2)} <_{\mathcal{P}} \boldsymbol{x}^{(1)}$.

De forma general, se dice que $\boldsymbol{y} \in S$ domina a $\boldsymbol{x} \in S$ si y solo si $\boldsymbol{f}(\boldsymbol{y}) <_{\mathcal{P}} \boldsymbol{f}(\boldsymbol{x})$ y se denota como:

$$\boldsymbol{y} \prec \boldsymbol{x}$$

Los vectores objetivo que no pueden ser dominados constituyen al conjunto de valores óptimos del problema en términos de Pareto. Esto significa que, si se mejora un componente de alguno de estos vectores objetivo, al menos otro de sus componentes va a empeorar. Estos vectores pertenecen a la denominada *Frontera de Pareto*, denotada como $\mathcal{P}$:

$$\mathcal{P} = \{\mathbf{f}(\mathbf{x}) | \mathbf{x} \in \Omega, \nexists \mathbf{y} \in \Omega, \mathbf{y} \prec \mathbf{x}\} \tag{1.7}$$

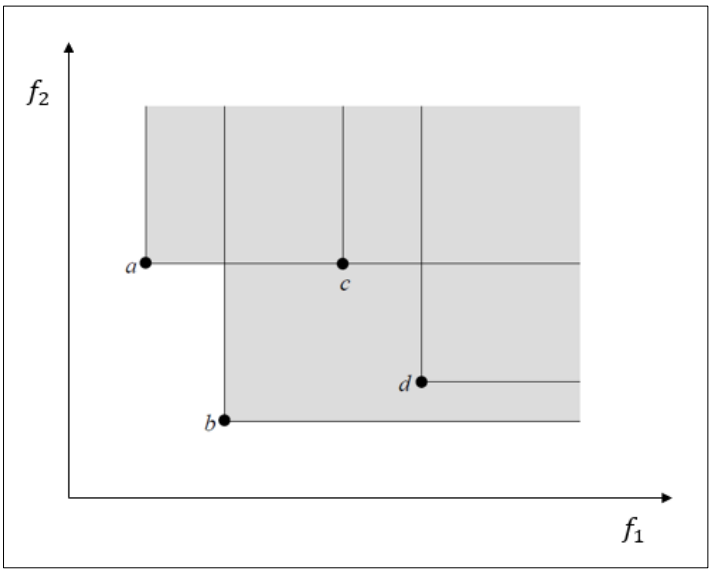

Figura 1.8. Dominancia de soluciones. Fuente (Pétrowski & Ben-Hamida, 2017)

Así, el conjunto óptimo de Pareto $\mathbf{X}^*$ se define como el conjunto de soluciones en el espacio de búsqueda $\boldsymbol{\Omega}$ cuyos vectores objetivo pertenecen a la frontera de Pareto:

$$\boldsymbol{X}^* = \{x \in \boldsymbol{\Omega} | \mathbf{f}(\mathbf{x}) \in \mathcal{P}\} \tag{1.8}$$

La optimización multiobjetivo consiste en construir el conjunto óptimo de Pareto $\boldsymbol{X}^*$. Sin embargo, este conjunto puede contener infinitas soluciones si el espacio de búsqueda Ω es continuo*. Incluso si Ω es finito, el diseñador será capaz de explotar las soluciones de forma efectiva si $\mathbf{X}^*$ no es muy grande. Por eso se espera del algoritmo produzca un conjunto o subconjunto de soluciones no dominadas, que no sea muy grande, y que esté lo más próximo posible al Frente de Pareto, cubriéndolo tan homogénea y completamente como sea posible.

1.3.5. Convexidad en problemas multiobjetivo

La convexidad de un MOP es una materia importante: existen muchos algoritmos capaces de manejar MOPs convexos pero que presentan dificultades cuando no lo son.

Una función objetivo $f: \mathbb{R}^n \rightarrow \mathbb{R}$ es convexa si para cualquier pareja de soluciones $x^{(1)}, x^{(2)} \in \mathbb{R}^n$ se cumple que:

$$f\left(\lambda x^{(1)} + (1-\lambda)x^{(2)}\right) \leq \lambda f\left(x^{(1)}\right) + (1-\lambda) f\left(x^{(2)}\right) \tag{1.9}$$

Con $0 \leq \lambda \leq 1$.

Así, una función convexa no puede tener valores mayores que los obtenidos por interpolación lineal entre $f\left(x^{(1)}\right)$ y $f\left(x^{(2)}\right)$.

Por otra parte, un conjunto de puntos o región en el espacio se define como un conjunto convexo en el espacio $\mathbb{R}^n$ si, para todos los pares de puntos $x^{(1)}$ y $x^{(2)}$ en el conjunto, el segmento que los une también pertenece al conjunto. En la Figura X se muestra una función convexa y una representación de espacios objetivo convexo y no convexo.

Los métodos que se basan en sumas ponderadas no pueden encontrar soluciones en regiones no convexas de la Frontera de Pareto, a pesar de que estas soluciones no dominadas existen con frecuencia. Esto se debe a que la estrategia de sumas ponderadas se implementa como una combinación de objetivos convexa, donde la suma de los pesos es constante y no se permiten pesos negativos (Kim & de Weck, 2005).

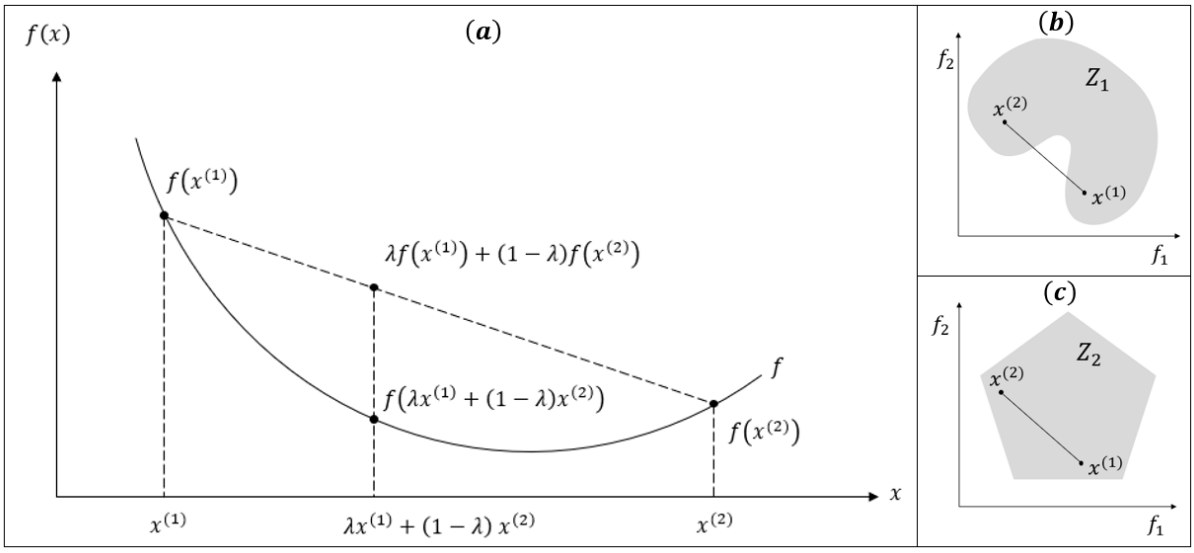

Figura 1.9. Convexidad (a) Fuente (Jahn, 2007), Espacio no convexo (b) y Espacio convexo (c). Fuente (Coello et al., 2007)

1.3.6. Indicadores de Calidad

Hay una amplia gama de elección para algoritmos de optimización multiobjetivo, cada uno con sus propias características. Estos normalmente usan parámetros cuyos valores pueden influir fuertemente en la calidad de los resultados, mientras que es difícil estimar si sus valores van a alcanzar los mejores resultados para el usuario o diseñador. Normalmente se selecciona la mejor aproximación comparando los resultados de cada algoritmo con cada conjunto de parámetros. Es por ello importante que dispongan de indicadores de calidad para facilitar el análisis de actuación de los distintos métodos considerados.

1.3.6.1. Medida del Hipervolumen ($\boldsymbol{HPV}$)

Un método interesante por sus buenas propiedades, a pesar de que requiere una elevada capacidad de computación, es la *medida del hipervolumen* o $\mathcal{S}$ – *metric*. Se basa en la medida del hipervolumen del frente obtenido a partir de un conjunto de soluciones no dominadas, donde el máximo hipervolumen se obtiene cuando dicho conjunto es propiamente el Frente de Pareto, y un valor elevado del hipervolumen indica que las soluciones no dominadas están próximas al Frente, con una buena calidad de dispersión o cobertura.

Para ello se fija un punto de referencia en el espacio objetivo $\boldsymbol{\rho} = (\rho_1, \dots, \rho_M)$. Entonces, si $\boldsymbol{a_i} = (a_1, \dots, a_M)$ es un elemento del conjunto $\boldsymbol{A}$ de soluciones no dominadas, para generar el hipervolumen se requiere que $a_m \leq \rho_m$. La expresión es:

$$HPV = v(\boldsymbol{A}, \boldsymbol{\rho}) = \prod_{m=1}^{M} (\rho_m - a_m) \tag{1.10}$$

El hipervolumen definido en el espacio objetivo consta de la unión de los hiperrectángulos asociados con los elementos del conjunto $\boldsymbol{A}$, como se muestra en la Figura X. El vector $\boldsymbol{\rho}$ se escoge tal que todos sus componentes sean un límite superior de los componentes de todos los puntos del conjunto $\boldsymbol{A}$. Así, si cogemos otro conjunto $\boldsymbol{B}$, y alguno de sus elementos es dominado por al menos un elemento del conjunto $\boldsymbol{A}$, entonces el hipervolumen $v(\boldsymbol{B}, \boldsymbol{\rho})$ será menor que el hipervolumen $v(\boldsymbol{A}, \boldsymbol{\rho})$.

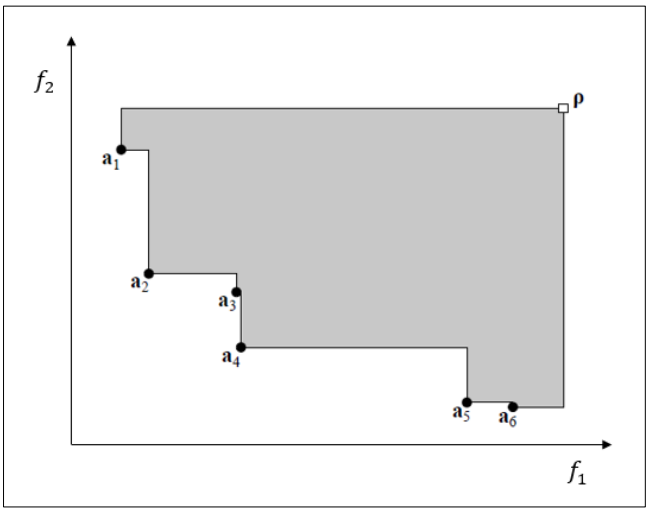

Figura 1.10. Hipervolumen $v(\boldsymbol{A}, \boldsymbol{\rho})$ con $\boldsymbol{A} = \{a_1, \dots, a_6\}$. Fuente:(Pétrowski & Ben-Hamida, 2017)

1.3.6.2. Espaciado Normalizado $(\boldsymbol{SP})$

Propuesto en (Schott, 1995), mide el espacio entre soluciones en el frente de Pareto para poder valorar su distribución a lo largo del frente.

Siendo $\boldsymbol{A}$ el conjunto final de los óptimos de Pareto obtenido por el algoritmo multiobjetivo, se define la función S:

$$SP = \sqrt{\frac{1}{|\boldsymbol{A}| - 1} \sum_{i=1}^{|\boldsymbol{A}|} \left(\bar{d} - d_i\right)^2} \tag{1.11}$$

Donde d_i es la distancia mínima entre la solución $f_m(\boldsymbol{a}_i)$ y cualquier otra considerada $f_m(\boldsymbol{a}_j)$, definida en este caso como la distancia euclídea:

$$d_i = \min_j \left\{ \sqrt{\sum_{m=1}^{k} \left(f_m(\boldsymbol{a}_i) - f_m(\boldsymbol{a}_j)\right)^2} \right\} \qquad \boldsymbol{a}_i, \boldsymbol{a}_j \in \boldsymbol{A} \qquad i, j = 1,2, \dots, |\boldsymbol{A}| \tag{1.12}$$

$\bar{d}$ es el valor medio de todos los d_i, y k es el número de funciones objetivo. Un valor nulo en esta métrica indica que las soluciones no dominadas están espaciadas de forma equidistante en el espacio objetivo.

Cuando el orden de magnitud entre soluciones dista de forma significativa, se define el Espaciado Normalizado como:

$$d_i = \min_j \left\{ \sqrt{\sum_{m=1}^{k} \left(\frac{f_m(\boldsymbol{a}_i) - f_m(\boldsymbol{a}_j)}{f_m^{max} - f_m^{min}}\right)^2} \right\} \qquad \boldsymbol{a}_i, \boldsymbol{a}_j \in \boldsymbol{A} \qquad i, j = 1,2, \dots, |\boldsymbol{A}| \tag{1.13}$$

1.3.6.3. Máximo Spread $(\boldsymbol{MS})$

Propuesto en (Zitzler et al., 2000), aborda el rango de valores de las funciones objetivo teniendo en cuenta su proximidad con la frontera de Pareto óptima, midiendo la calidad de la cobertura del frente obtenido con respecto al ideal.

$$MS = \sqrt{\frac{1}{k} \sum_{m=1}^{k} \left(\frac{\max\left(f_m(\boldsymbol{a}_i)\right) - \min\left(f_m(\boldsymbol{a}_i)\right)}{f_m^{\max\, ideal} - f_m^{\min\, ideal}}\right)^2} \qquad \boldsymbol{a}_i \in \boldsymbol{A} \qquad i = 1,2, \dots, |\boldsymbol{A}| \tag{1.14}$$

1.3.6.4. Distancia Generacional $(\boldsymbol{GD})$

Propuesto en (van Veldhuizen & Lamont, 1999), se basa también en medir la cercanía de las soluciones a la frontera óptima:

$$GD = \frac{\sqrt{\sum_{i=1}^{|\boldsymbol{A}|} d_{min,i}^2}}{|\boldsymbol{A}|} \qquad i = 1,2,\dots,|\boldsymbol{A}| \tag{1.15}$$

Donde la distancia $d_{min,i}$ es la distancia en el espacio objetivo entre el individuo $\boldsymbol{a}_i$ y el miembro $\boldsymbol{o}_j$ más cercano a la frontera óptima de Pareto, $\boldsymbol{O}$:

$$d_{min,i} = \min_j \left\| f(\boldsymbol{a}_i) - f(\boldsymbol{o}_j) \right\| \qquad \boldsymbol{o}_j \in \boldsymbol{O} \qquad j = 1,2,\dots,|\boldsymbol{O}| \tag{1.16}$$

1.3.6.5. Cobertura de conjunto – $\boldsymbol{C(AB)}$

Propuesta en (Zitzler & Thiele, 1999), considera la relación de dominancia entre dos algoritmos, esto es, cómo de bien domina una frontera obtenida de un algoritmo a otra obtenida por un algoritmo distinto y viceversa, calculando la proporción de puntos en una frontera que son dominados por al menos una solución perteneciente a la otra frontera:

$$C(AB) = \frac{\left| b^1 \in \boldsymbol{B} | \exists a^2 \in \boldsymbol{A} : a^2 \prec b^1 \right|}{|\boldsymbol{B}|}$$
$$C(BA) = \frac{\left| a^1 \in \boldsymbol{A} | \exists b^2 \in \boldsymbol{B} : b^2 \prec a^1 \right|}{|\boldsymbol{A}|} \tag{1.17}$$

Con $C(AB) \neq C(BA)$

En el extremo, si $C(AB) = 0$ y $C(BA) = 1$, ninguna solución en el conjunto $\boldsymbol{B}$ es dominada por alguna solución del conjunto $\boldsymbol{A}$ y, a su vez, todas las soluciones del conjunto $\boldsymbol{A}$ son dominadas por al menos una solución del conjunto $\boldsymbol{B}$.

1.4. Algoritmos Genéticos Multi-Objetivo (MOEAs)

Los algoritmos genéticos son apropiados para la búsqueda simultánea de una colección de soluciones óptimas porque manejan poblaciones de soluciones, donde múltiples individuos buscan múltiples soluciones en paralelo, tomando ventaja de las similitudes disponibles en la familia de posibles soluciones (Fonseca, 1995). El enfoque evolucionario requiere la implementación del archivo de soluciones no dominadas descubiertas durante una evolución completa. Con todo, no hay garantías de que al final de una evolución las soluciones que tengan una mejor aproximación hacia el Frente de Pareto prosperen en la población. Al final de cada generación, la población se copia en el archivo de soluciones y después se eliminan los individuos dominados. Sin embargo, existen otras técnicas que evitan la creación y mantenimiento de dicho archivo implementando lo que se conoce como elitismo.

1.4.1. Primera Generación de Algoritmos Genéticos Multiobjetivo

Dentro de la clasificación anterior, hay dos clases de enfoques evolucionarios que destacan por haber sido ampliamente utilizados en la literatura (Pétrowski & Ben-Hamida, 2017):

- Métodos que usan un *Ranking de Pareto* para evaluar la función fitness.
- Métodos de *agregación* que transforman un problema de optimización multiobjetivo en una colección de problemas de un único objetivo. La solución exacta (o aproximada) de cada problema de un objetivo da un punto perteneciente o cercano al Frente Pareto.

Los métodos con Ranking de Pareto fueron los primeros en mostrar su eficiencia dando una buena cobertura del frente (un conjunto de soluciones lo suficientemente dispersa y homogénea a lo largo del frente). Los individuos de la población corresponden a soluciones en el espacio de búsqueda. Para cada uno de ellos, se evalúa y asigna un vector objetivo. Después, cada individuo obtiene un valor escalar de

aptitud o fitness (una puntuación), que se computa con los vectores objetivo, tal que los individuos no dominados son seleccionados con mayor frecuencia que los otros.

La cobertura del frente se consigue mediante un mecanismo de preservación de la diversidad en la población. Las principales dificultades de estos algoritmos aparecen cuando tenemos muchos objetivos que optimizar. Cuantos más objetivos, más vasto es el frente y menos probable es que haya individuos que dominen a otros, haciendo lenta la convergencia del algoritmo.

Como primera propuesta está la de (Goldberg, 1989), consistente en dividir la población según las similitudes entre individuos. La forma de medir esta similitud es mediante el parámetro σ_{share} o radio de nicho, que actúa sobre la cantidad de información intercambiada entre dos miembros. Para favorecer la formación de nichos, existe una función de reparto de aptitudes (*fitness sharing*) donde individuos de un nicho comparten el valor de aptitud con sus vecinos.

Aún sin describir una implementación concreta del algoritmo, la idea inspiró a muchos investigadores en los siguientes años, dando lugar al nacimiento de la primera generación de algoritmos que utilizan el ranking de Pareto como MOGA, NPGA y NSGA.

1.4.1.1. VEGA – Schaffer (1985)

Vector Evaluated Genetic Algorithm (Schaffer, 1985): Fue el primer método multiobjetivo utilizado en la categoría de selección de criterio (métodos a posteriori), implementado por Parmee y Purchase en 1994 para la optimización en el diseño de una turbina de gas con un espacio de búsqueda extremadamente restringido. El principio se basa en el uso de subpoblaciones gobernadas por las diferentes funciones objetivo, optimizándolas de forma separada. El concepto de óptimo de Pareto no está directamente incorporado en el mecanismo de selección del algoritmo, sino que mediante los operadores genéticos consigue que el algoritmo permanezca en la región factible del espacio de búsqueda. Esto provoca la convergencia hacia regiones extremas en el frente de Pareto cuando tenemos un cambio de concavidad, debido a que las direcciones de búsqueda del algoritmo son paralelas a los ejes del espacio objetivo.

Schaffer sugirió dos enfoques para mejorar VEGA; uno consiste en utilizar una selección heurística para los individuos no dominados en cada generación, pero esta heurística falla cuando la población tiene pocos individuos no dominados. El otro consiste en hibridar a las especies (soluciones próximas entre sí, pero en regiones diferentes del frente) en la fase de selección, pero también falla a la hora de prevenir la participación de individuos peores en la selección por su carácter aleatorio.

1.4.1.2. MOGA – Fonseca & Fleming (1993)

Multi Objective Genetic Algorithm (Fonseca & Fleming, 1993): Basado en un Ranking de Pareto, el método busca optimizar los componentes de una función coste evaluada con un sistema vectorial mediante una población de soluciones donde el algoritmo genético puede realizar una búsqueda en paralelo de soluciones no dominadas.

Nació como solución al problema de convergencia del método VEGA con frentes de Pareto de secciones cóncavas, donde las soluciones tendían a dividirse formando especies, cada una de ellas particularmente fuerte en uno de los objetivos.

Así, definió un sistema de aptitud o fitness basado en un ranking, donde a cada individuo no dominado se le asigna un rango. Con ello se consigue ordenar la población según el rango, asignar aptitudes interpolando desde el mejor rango al peor y promediar la aptitud de los individuos con el mismo rango, para que todos sean seleccionados con la misma probabilidad.

1.4.1.3. NPGA – Horn J., Nafpliotis N., Goldberg D.E. (1994)

Niched Pareto Genetic Algorithm (Horn et al., 1994): Implementa la selección del más apto mediante una competición de individuos, donde se selecciona aleatoriamente a un conjunto de individuos de la población actual y el mejor de este subconjunto formará parte de la siguiente generación. Ajustando el tamaño del conjunto de competidores se puede ajustar la velocidad de convergencia, donde

competiciones de conjuntos de dos individuos se corresponden con la velocidad mínima de convergencia.

Para evitar el estancamiento en una zona concreta del frente y mantener la diversidad de múltiples soluciones óptimas, esta filosofía de competición se extiende también a competiciones de dominancia Pareto y competición de no dominados en donde se reparten las aptitudes para distribuir la población sobre los diferentes picos en el espacio de búsqueda, recibiendo cada pico una fracción de la población en proporción a la altura de dicho pico.

1.4.1.4. NSGA – Srinivas N., Deb K. (1994)

Nondominated Sorting Genetic Algorithm (Srinivas & Deb, 1994): En sintonía con las necesidades del momento, este método nace para continuar aportando soluciones contra el estancamiento que tiene el algoritmo VEGA para frentes de Pareto de secciones cóncavas. La principal diferencia con respecto a los dos métodos anteriores (que compartían la misma inquietud) es en cuanto a los principios de funcionamiento. NSGA implementa la noción que dio Goldberg en 1989 para ordenar las soluciones no dominadas con un método basado en los nichos de soluciones y la especiación, consiguiendo eliminar la parcialidad del método VEGA y por lo tanto distribuyendo la población entre todas las regiones óptimas del frente de Pareto.

La idea de NSGA es implementar un ranking para realzar las soluciones buenas y un método de nicho o anidado para mantener estables a las subpoblaciones de soluciones buenas. Difiere en los métodos tradicionales en cuanto al operador de selección; el cruce y la mutación funcionan igual.

Antes de realizar la selección, la población se clasifica de acuerdo con su dominancia. Los individuos no dominados son los primeros en esta clasificación. A estos se les da una calificación de fitness elevada para dar el mismo potencial reproductivo a todos estos individuos no dominados. Después se reparten las aptitudes del individuo dividiendo el valor original por una cantidad proporcional al número de individuos alrededor de éste. Después se procesa al resto de la población de la misma manera para identificar a los individuos del segundo frente de dominancia. Ahora se les asigna de nuevo un valor de aptitud, pero esta vez más pequeño que el mínimo del frente anterior. Se continúa el proceso hasta que la población completa está clasificada en diversos frentes.

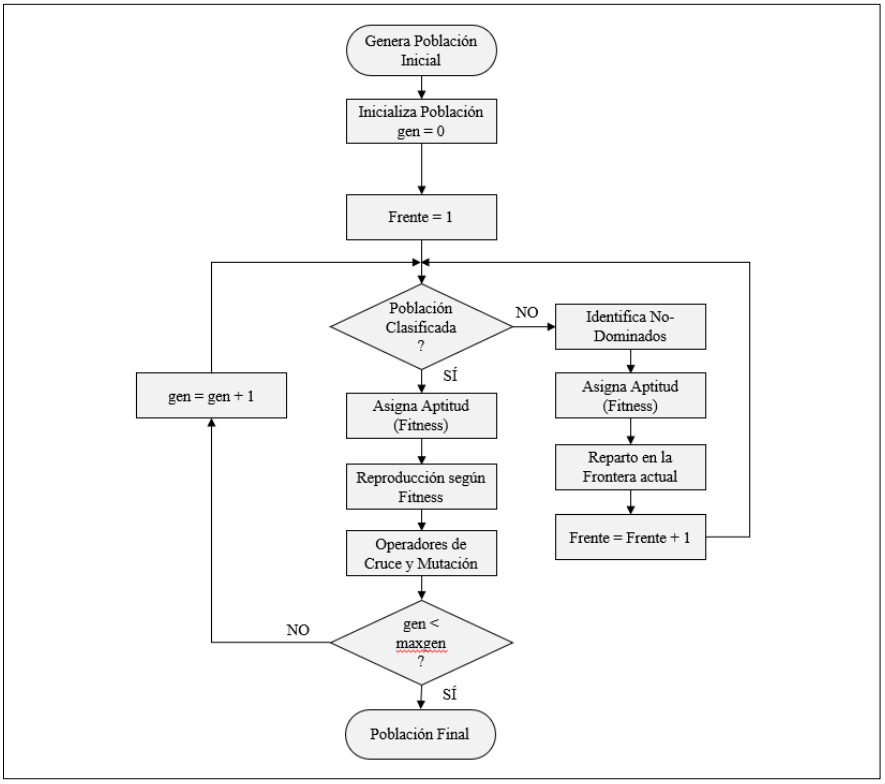

Figura 1.11. NSGA Diagrama de flujo. Fuente (Srinivas & Deb, 1994)

1.4.2. Segunda Generación de Algoritmos Genéticos Multiobjetivo

En la década del 2000, estos enfoques se mejoraron con la introducción del *elitismo*, dando paso al nacimiento de la segunda generación de algoritmos genéticos multiobjetivo. Algunos de los más utilizados son NSGA-II, SPEA, SPEA2, PAES (Pétrowski & Ben-Hamida, 2017).

1.4.2.1. SPEA – Zitzler E., Thiele L. (1998)

Strength Pareto Evolutionary Algorithm (Zitzler & Thiele, 1998): Combina varios factores de las versiones anteriores aunque de una manera diferente. Además de la población, aparte se mantiene a un conjunto de individuos que contiene las soluciones óptimas de Pareto generadas hasta el momento. Este conjunto se utiliza para evaluar la aptitud de un individuo según la relación de dominancia en el frente. En lugar de utilizar un reparto de aptitudes, ahora la diversidad de la población se preserva en base a la dominancia de Pareto en lugar de la distancia entre éstas. Se introduce también un método de agrupación o *cluster* para reducir el tamaño del conjunto independiente de Pareto sin destruir sus características.

En primer lugar, se asigna un valor real a cada solución en el conjunto de Pareto denominado *strength* (fortaleza) $s \in [0,1)$. Este valor de s representa también aptitud o fitness de una solución. Tras esto, en un segundo paso, se realiza un ranking de los individuos en la población en función de su valor de s. Esto es, para cada individuo se suman las fortalezas s de todas las soluciones Pareto por las que está cubierto. Se suma uno al valor resultante, garantizando que las soluciones Pareto tengan mayor probabilidad de reproducirse, y se obtiene el valor de fitness f, donde $f \in [1, N)$ y N es el tamaño de la población.

Si todas las soluciones de Pareto tienen el mismo valor de fortaleza, la aptitud de un individuo se determina por el número de puntos de Pareto que lo cubren. En cambio, en caso de que la población no esté equilibrada entran en juego las fortalezas, de forma que cuanto más fuerte sea un individuo, menos aptos serán los que estén cubiertos por éste.

El algoritmo comienza con la actualización del conjunto Pareto: todos los individuos no dominados en la población se copian al conjunto Pareto y, de forma consecutiva, los individuos que posiblemente han pasado a ser dominados se eliminan del conjunto. Si el número de soluciones Pareto almacenadas externamente excede un máximo determinado, se computa una representación reducida del conjunto a través del clustering. Después de la asignación de fitness o aptitud, se seleccionan individuos aleatorios del conjunto resultante de la unión del conjunto población y conjunto Pareto externo para llenar el recinto de apareamiento o *mating pool* y competir en una competición binaria. Finalmente se aplican los operadores de cruce y mutación.

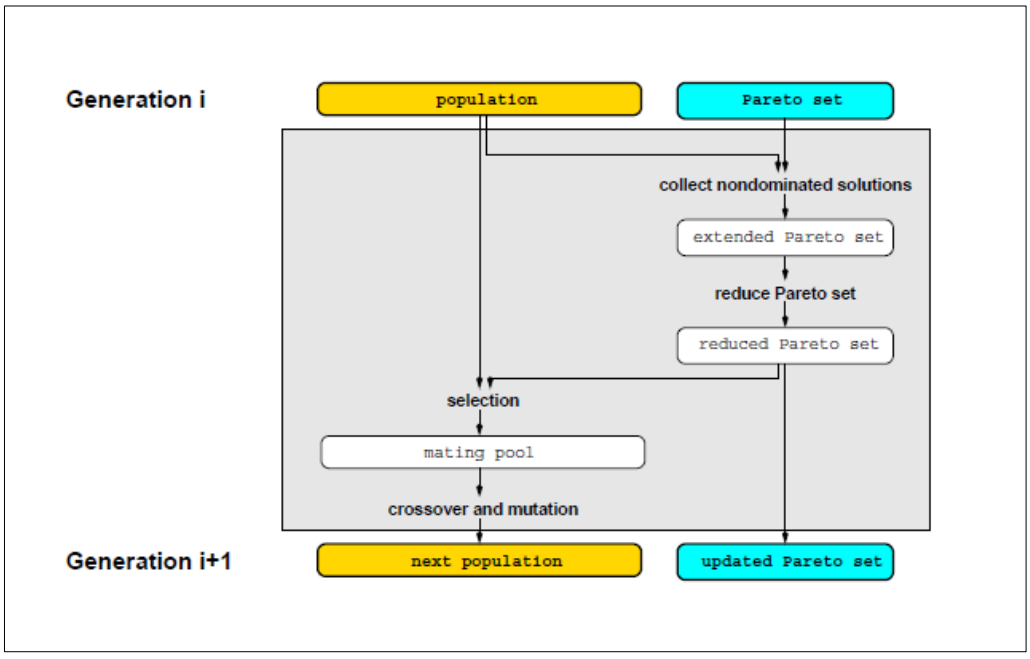

Figura 1.12. SPEA - Esquema de funcionamiento. Fuente (Zitzler & Thiele, 1998)

1.4.2.2. PAES – Knowles, J., Corne, D. (1999)

Pareto Archived Evolution Strategy (Knowles & Corne, 1999): Knowles y Corne crearon un algoritmo sencillo utilizando una estrategia evolutiva con un padre y un descendiente o hijo. En lugar de usar parámetros de tipo real, se codifica la solución mediante cadenas binarias donde la mutación se realiza bit a bit para crear al descendiente. Una vez se obtiene al descendiente, se realiza una comparación con el padre. Si el hijo domina al padre, se acepta como siguiente padre y se continúa la iteración. Si el padre domina al hijo, el descendiente se descarta y se obtiene una nueva solución mutada. Sin embargo, si el descendiente y el padre no se dominan entre ellos, la elección entre el descendiente y el padre se hace mediante una comparación con el archivo de mejores soluciones encontradas hasta el momento. Así, se compara al descendiente con el archivo para comprobar si domina a algún miembro y, si lo hace, el descendiente se acepta como nuevo padre y todas las soluciones dominadas se eliminan del archivo. Si el descendiente no domina a ningún miembro del archivo, se analiza la cercanía de ambos a soluciones del archivo. Si el descendiente reside en la región menos poblada del espacio objetivo entre los miembros del archivo, se acepta como padre y se añade una copie de él al archivo. El distanciamiento se mantiene dividiendo el espacio de búsqueda en subespacios y mediante la actualización dinámica de éstos.

1.4.2.3. NSGA-II – K. Deb et al. (2000)

Non Dominated Sorting Genetic Algorithm II (Deb et al., 2000): Los algoritmos genéticos que usaban métodos de ordenación de no dominados y reparto de aptitudes fueron criticados, según recoge Deb, principalmente por tres motivos:

1. Complejidad computacional de orden $O(MN^3)$. Con M número de objetivos y N tamaño de la población.
2. Enfoque no-elitista, perdiendo aceleración en el rendimiento del algoritmo.
3. La necesidad de especificar el valor del parámetro de reparto de aptitudes σ_{share}.

Esta versión de NSGA alivia estos problemas, reduciendo la complejidad computacional a $O(MN^2)$, implementando un operador de selección que crea una *piscina de apareamiento (mating pool)* en donde se combina al padre con las poblaciones descendientes, seleccionando las N soluciones mejores.

Este algoritmo es uno de los tres implementados en este trabajo, por lo que será descrito con mayor detalle en el siguiente capítulo.

1.4.2.4. SPEA2 – Zitzler E., Laumanns M., Thiele L. (2001)

SPEA 2: Improving Strength Pareto Evolutionary Algorithm (Zitzler et al., 2001). Aunque el algoritmo SPEA ha sido probado con eficacia en diversos estudios, se han identificado algunos aspectos como debilidades que son tratadas en esta nueva versión del algoritmo.

1. Asignación de fitness: Los individuos que son dominados por los mismos miembros del archivo del conjunto de Pareto tienen valores de fitness idénticos. Esto implica que, si el archivo contiene un solo individuo, toda la población tendrá el mismo rango independientemente de que entre ellos se dominen o no. El algoritmo se comporta en este caso como un algoritmo de búsqueda aleatoria.
2. Estimación de densidad: Si hay mucha diferencia entre los individuos de la generación actual, esto es, no son dominados entre sí, se obtiene muy poca información (o ninguna) sobre el orden parcial en la relación de dominancia. En esta probable situación, se debe utilizar una densidad de información para hacer la búsqueda más efectiva. El proceso de clustering utiliza esta información, pero sólo la correspondiente al archivo Pareto (no al resto de la población).
3. Truncamiento del archivo: La técnica de clustering es capaz de reducir el conjunto no dominado sin destruir sus características, pero puede perder soluciones externas que deberían guardarse en el archivo para obtener un buen *spread* (propagación) de soluciones no dominadas.

Así, en contraste con la versión anterior, SPEA2 implementa una estrategia de *ajuste de grano fino* que incorpora la mencionada densidad de información. Además, el tamaño del archivo Pareto se ajusta incluyendo un número de soluciones dominadas hasta completar el tamaño predefinido, sin hacer variar

su tamaño con el tiempo. Por último, en lugar de clustering, SPEA2 implementa un método de truncamiento donde no se pierden soluciones externas.

Este algoritmo es el segundo de los tres implementados en este trabajo, por lo que será descrito con mayor detalle en el siguiente capítulo.

1.4.2.5. MOEA/D – Zhang y Li (2008)

Multiobjective Evolutionary Algorithm Based on Descomposition (Zhang & Li, 2008)*:* La estrategia de este algoritmo se basa en descomponer el problema de optimización multiobjetivo MOP en un determinado número de subproblemas de optimización escalar que son optimizados de forma simultánea. Cada subproblema se optimiza utilizando información que proviene exclusivamente de sus subproblemas vecinos, consiguiendo una menor complejidad computacional en cada generación comparado con NSGA-II. MOEA/D tiene las siguientes características:

- Consigue introducir métodos de descomposición a MOPs de forma sencilla y eficiente en comparación con los algoritmos anteriores.
- Como optimiza al mismo tiempo N subproblemas en lugar de resolver el MOP como un todo, es más fácil la asignación de fitness y el mantenimiento de la diversidad.
- Tiene una complejidad computacional inferior a los algoritmos anteriores.
- Puede utilizar técnicas de normalización de objetivos para tratar con objetivos dispares.

Cuando se desarrolló este algoritmo había una tendencia a dejar de lado la descomposición, tratando el problema MOP como un todo. No se asociaba a cada solución individual un problema de optimización escalar particular. Aquí la idea es (o suele ser) buscar una única solución óptima, para después poder comparar todas las soluciones basándonos en su valor en las funciones objetivo.

Zhang defiende que la relación de dominancia en problemas multiobjetivo no define un orden completo en las soluciones del espacio objetivo y los algoritmos genéticos tratan de producir un número de soluciones Pareto tan diversas como sea posible para representar el frente completo. Por lo tanto, los operadores convencionales de selección (originalmente diseñados para la optimización escalar) no se pueden utilizar directamente en métodos que no estén basados en la descomposición del MOP. Afirma que, si hay un esquema de asignación de fitness para cada solución individual, en donde se asigna un valor relativo de fitness para reflejar la utilidad de esta selección, entonces los algoritmos genéticos de optimización escalar sí pueden extender su utilidad para tratar con MOPs, aunque requiriendo otras técnicas tales como restricciones de apareamiento, mantenimiento de la diversidad, propiedades del MOP y poblaciones externas para potenciar el rendimiento de estos algoritmos.

Por esta razón la asignación de fitness ha sido un problema de importancia en la investigación de MOEAs. Las estrategias populares de asignación de fitness incluyen la alternancia de asignación de fitness basada en objetivos como VEGA, la asignación basada en la dominancia como PAES, SPEA2 y NSGA-II.

Este algoritmo es el tercero de los tres implementados en este trabajo, por lo que será descrito con mayor detalle en el siguiente capítulo.

1.4.3. Siguientes Generaciones de Algoritmos Multiobjetivo

La Tabla 1.1 recoge una colección de algoritmos genéticos cuya eficacia ante problemas multiobjetivo ha sido testada y publicada en conferencias de la comunidad de computación evolucionaria.

Estos algoritmos evalúan a los individuos utilizando la dominancia de Pareto, que evalúa la aptitud junto con el mantenimiento de la diversidad, funcionando bien para dos o tres objetivos. Sin embargo, el rendimiento del algoritmo cae drásticamente cuando aumenta el número de objetivos, debido a que casi la totalidad de la población se convierte en solución no dominada. Aparte de los basados en la dominancia de Pareto, hay un interés creciente en los métodos basados en la descomposición del MOP en subproblemas escalares como el MOEA/D propuesto por (Zhang & Li, 2008) o NSGA-III por (Deb & Jain, 2014). Este tipo de algoritmos aumenta el rendimiento para estos problemas por el ajuste que realiza de los vectores de ponderación en los subproblemas (Li & Zhang, 2020).

En la Tabla 1.2 se recogen los algoritmos aptos para el tratamiento de problemas con mayor número de objetivos.

Algoritmo	**Año**	**Descripción**
M-PAES	2000	Memetic algorithm based on Pareto archived evolution strategy
NSGA II	2000	Non dominated sorting genetic algorithm II
PSEA II	2001	Pareto envelope-based selection algorithm II
SPEA2	2001	Strength Pareto Evolutionary Algorithm 2
MOPSO	2002	Multi-objective particle swarm optimization
E MOEA	2003	Multi-objective evolutionary algorithm based on e-dominance
IB EA	2004	Indicator-based evolutionary algorithm
ParEGO	2005	Efficient global optimization for Pareto optimization
MOEA/D	2007	Multi-objective evolutionary algorithm based on decomposition
MSOPS II	2007	Multiple single objective Pareto sampling algorithm II
SMS EMOA	2007	S-metric selection evolutionary multi-objective optimization algorithm
SMS-EGO	2008	S-metric-selection-based efficient global optimization
SMPSO	2009	Speed-constrained multi-objective particle swarm optimization
MTS	2009	Multiple trajectory search
dMOPSO	2011	Decomposition-based particle swarm optimization
AGE II	2013	Approximation guided evolutionary algorithm II
BCE IB EA	2015	Bicriterion evolution for IBEA
NSLS	2015	Non dominated sorting and local search
K-RVEA	2016	Kriging assisted RVEA
MOEA/IGD NS	2016	Multiobjective evolutionary algorithm based on an enhanced inverted generational distance metric

Tabla 1.1. Recopilación de Algoritmos Genéticos Multi-Objetivo. Fuente (Tian et al., 2017)

Algoritmo	Año	Descripción
HypE	2011	Hyper volume-based estimation algorithm
GrEA	2013	Grid-based evolutionary algorithm
PICEA-g	2013	Preference-inspired coevolutionary algorithm with goals
A-NSGA-III	2014	Adaptive NSGA-III
NSGA-III	2014	Non-dominated sorting genetic algorithm III
SPEA2+SDE	2014	SPEA2 with shift-based density estimation
BiGE	2015	Bi-goal evolution
EFR-RR	2015	Ensemble fitness ranking with ranking restriction
I-DBEA	2015	Improved decomposition based evolutionary algorithm
KnEA	2015	Knee point driven evolutionary algorithm
MaOEA-DDFC	2015	Many-objective evolutionary algorithm based on directional diversity and favorable convergence
MOEA/DD	2015	Multi-objective evolutionary algorithm based on dominance and decomposition
MOMBI -II	2015	Many-objective meta-heuristic based on the R2 indicator II
Two_Arch2	2015	Two-archive algorithm 2
i-DEA	2016	i-dominance based evolutionary algorithm
MaOEA-R&D	2016	Many-objective evolutionary algorithm based on objective space reduction and diversity improvement
RPEA	2016	Reference points-based evolutionary algorithm
RVEA	2016	Reference vector guided evolutionary algorithm
RVEA*	2016	RVEA embedded with the reference vector regeneration strategy
SPEA/R	2016	Strength Pareto evolutionary algorithm based on reference direction

Tabla 1.2. Recopilación de algoritmos many-objective. Fuente (Tian et al., 2017)

2. ALGORITMOS DE TRABAJO

Para comprobar la eficacia de los algoritmos seleccionados en este trabajo, se va a seleccionar un problema objeto de estudio actualmente en la Universidad, conocido como la sintonización de TMDs en edificios esbeltos. A continuación, se describe el funcionamiento en mayor detalle los tres algoritmos utilizados: NSGA-II, SPEA2 y MOEA/D, para después dar paso a la definición del problema del TMD.

2.1. NSGA-II – K. Deb

Quizá el algoritmo más consolidado hasta la fecha por sus buenas prestaciones, NSGA-II es una versión mejorada del clásico NSGA en donde se resuelven los problemas identificados por Deb como el de eliminar la necesidad de especificar parámetros por el usuario, disminuir la complejidad computacional e introducir el concepto de elitismo. En la mayoría de los aspectos, este algoritmo no tiene mucha similitud con el original NSGA, pero los autores conservaron el nombre para subrayar su origen (Deb, 2001a).

2.1.1. Funcionamiento

La población descendiente Q_t se crea a partir de la población progenitora P_t. Ahora, en lugar de buscar soluciones no dominadas en Q_t, primero se combinan las poblaciones para formar R_t de tamaño $2N$. Después se utiliza un proceso de ordenación de no dominados para clasificar la población R_t. A pesar de que esta operación requiere un mayor esfuerzo que ordenar la población Q_t, permite realizar una comprobación global entre los progenitores y la descendencia. Una vez se ha terminado el proceso de ordenación, la nueva población se ocupa con soluciones no dominadas de distintos frentes, de una en una. La ocupación comienza con la mejor frontera no dominada, sigue con la segunda, la tercera y así sucesivamente. Como el tamaño de R_t es $2N$, no se podrán emplazar todos los frentes en los N espacios disponibles en la nueva población P_{t+1}. Todos los individuos que no puedan acceder son eliminados. Cuando se está considerando la última frontera permitida, pueden existir más soluciones en el frente que espacios disponibles en la nueva población. En este caso, ilustrado en la Figura 2.1, en lugar de descartar arbitrariamente algunos miembros de la frontera, se utiliza una estrategia de anidado (*niching*) para elegir a los miembros, de forma que se aceptan en la nueva población los que vivan en las regiones menos pobladas. Los efectos de esta estrategia son poco apreciables en las etapas tempranas de la evolución. Esto es debido a la existencia de múltiples fronteras en la población combinada. Es probable que las soluciones de muchos frentes no dominados ya estén incluidas en la nueva población antes de completar el tamaño N. En cambio, durante las etapas más avanzadas de la evolución, es probable que la mayoría de las soluciones en la población se acomoden en los mejores frentes no dominados. Así, cuando toda la población converge al frente de Pareto óptimo, la estrategia de este algoritmo asegura una buena dispersión (*spread*) entre las soluciones.

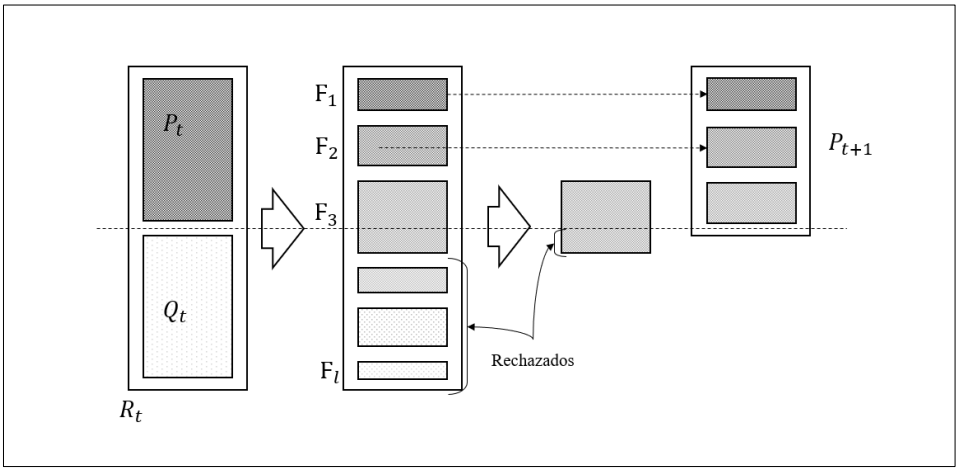

Figura 2.1. Generación de descendencia en NSGA-II. Fuente (Deb et al., 2000)

2.1.2. Bucle principal

Inicialmente se crea una población aleatoria P_o y se ordena en diferentes niveles de dominancia. A cada solución se le asigna un valor de aptitud igual a su nivel de no dominancia, siendo 1 el mejor nivel. A continuación, se utiliza la selección por torneo binario con un operador de truncamiento, la recombinación y el operador de mutación para generar la población descendiente Q_o de tamaño N.

1. Combinar las poblaciones progenitora y descendiente para generar el conjunto $R_t = P_t \cup Q_t$. Realizar un proceso de ordenación de no dominados e identificar las diferentes fronteras $\mathcal{F}_i$ con $i = 1,2, \dots, etc.$
2. Definir una nueva población como un conjunto vacío $P_{t+1} = \emptyset$. Inicializar un contador $i = 1$ y, hasta que el número de individuos $|P_{t+1}| + |\mathcal{F}_i| < N$, realiza $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$ y $i = i + 1$.
3. Iniciar el proceso de *ordenación de multitudes* $(\mathcal{F}_i, <_c)$ e incluir las $(N - |P_{t+1}|)$ soluciones con mayor dispersión, utilizando los valores de distancia entre multitudes del conjunto ordenado $\mathcal{F}_i$ a P_{t+1}.
4. Crear la población descendiente Q_{t+1} a partir de P_{t+1} utilizando la selección por torneo de multitudes, junto con los operadores de cruce y mutación.

En el Paso 3, el proceso de ordenación grupal para las soluciones del frente $\mathcal{F}_i$ (el último frente que no podría ser emplazado al completo en la nueva población) se realiza utilizando la *métrica de distancia grupal*, que se describirá más adelante. La población se ordena en orden descendiente de magnitud según sus valores en esta distancia grupal. En el Paso 4 se utiliza un operador de selección por torneo de multitudes, que también utiliza la distancia grupal.

Es importante mencionar aquí que la ordenación de no dominados en el Paso 1 y el emplazamiento de individuos en P_{t+1} se pueden realizar a la vez. De esta forma, cada vez que se encuentra un frente no dominado, su tamaño se puede utilizar para comprobar si es posible incluirlo en la nueva población. Si no es posible, no es necesario continuar con el proceso de ordenación, por lo que se reduce el tiempo de computación para el algoritmo.

2.1.3. Operador de Selección por Torneo de Multitudes

El operador de comparación de multitudes $(<_c)$ compara dos soluciones y devuelve al vencedor del torneo, asumiendo que cada solución tiene dos atributos:

- Un rango de no dominancia (r_i) en la población.
- Una distancia grupal local (d_i) en la población, que se definirá a continuación.

La distancia grupal (d_i) de la solución (i) es una medida del espacio de búsqueda no ocupado por otra solución en torno a (i). Basándonos en estos dos atributos, se define el operador de selección por torneo de multitudes de la siguiente forma:

Una solución (i) gana el torneo con otra solución (j) si se cumple alguna de las siguientes condiciones:

- La solución (i) tiene mejor rango que la solución (j).

$$r_i < r_j$$

- Si las soluciones tienen el mismo rango, la solución (i) tiene un mejor valor de distancia grupal.

$$r_i = r_j \qquad\qquad d_i > d_j$$

La primera condición asegura que la solución elegida cae en una frontera de no dominancia mejor. La segunda condición resuelve el problema en caso de que las dos soluciones pertenezcan al mismo frente, seleccionando aquella que resida en una región menos poblada, esto es, con una mayor distancia grupal (d_i). Esta distancia se puede calcular de varias formas. De estas, la métrica de contador de nichos (nc_i) y la métrica de recuento (hc_i) son de uso común (Deb, 2001a). A pesar de que se puedan utilizar, se deben de utilizar en sentido inverso. Esto es, si una solución tiene un valor de contador de nichos o recuento menor, implica una solución en una región menos poblada y es preferida por ello.

2.1.4. Distancia Grupal

Para obtener una estimación de la densidad de soluciones alrededor de una particular (i) en la población, se toma la distancia media de dos soluciones a cada lado de la solución (i) a lo largo de cada objetivo. Esta cantidad $d_i$sirve como estimación del perímetro del cuboide formado por los vecinos más cercanos a (i) como vértices. Así, la distancia grupal de la solución (i) en su frontera es la longitud media del lado del cuboide.

Procedimiento para la asignación de la distancia grupal y ordenación de multitudes $(\mathcal{F}, <_c)$

1. Denominamos l al número de soluciones en $\mathcal{F}$, esto es: $l = |\mathcal{F}|$. Para cada uno de los individuos (i) en el conjunto, inicializamos $d_i = 0$.
2. Para cada función objetivo $m = 1,2, \dots, M$, se ordena el conjunto según el peor orden de f_m o se halla el vector de índices ordenados $I^m = ordena(f_m, >)$.
3. Para cada $m = 1,2, \dots, M$, se asigna un valor elevado de distancia a las soluciones límite $d_{I_1^m} = d_{I_l^m} = \infty$. Para todo el resto de soluciones, desde $j = 2$ hasta $j = (l - 1)$:

$$d_{I_j^m} = d_{I_j^m} + \frac{f_m^{I_{j-1}^m} - f_m^{I_{j+1}^m}}{f_m^{max} - f_m^{min}} \tag{2.1}$$

El índice I_j representa el índice del j-ésimo miembro en la lista ordenada. Así, para cualquier objetivo, I_1 e I_l representan los valores más bajo y más elevado de las funciones objetivo, respectivamente. El segundo término del sumando es la diferencia en los valores de las funciones objetivo entre dos soluciones vecinas a cada lado de la solución I_j. De esta forma esta métrica denota la mitad del perímetro del cuboide circundante a la solución I_j, con las soluciones vecinas más cercanas como vértices (Figura 2.2).

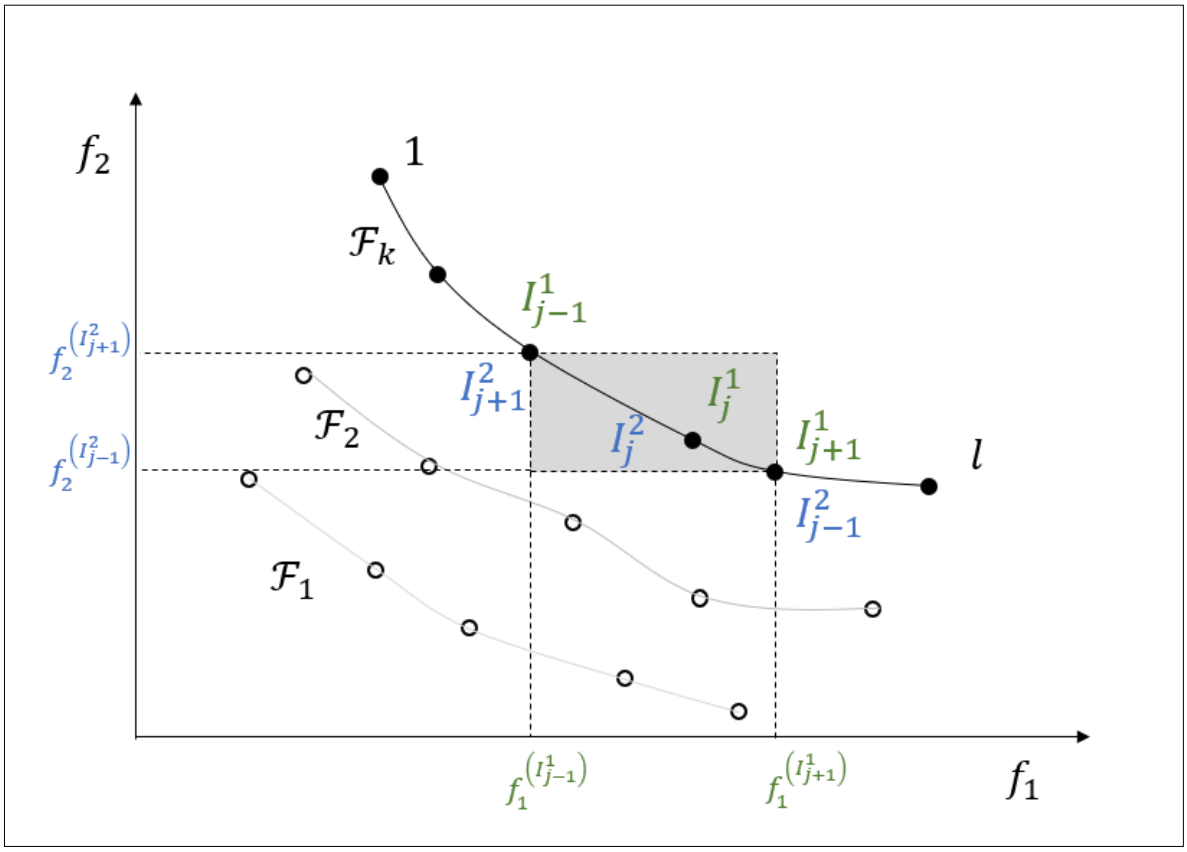

Figura 2.2. Cálculo de la distancia grupal. Cuboide de la solución I_j^m. Fuente:(Deb, 2001a)

2.1.5. Ejemplo práctico

Supongamos la siguiente población progenitora y descendiente, con sus correspondientes valores objetivo:

$Población\ P_t$					$Población\ Q_t$				
$Sol.$	x_1	x_2	f_1	f_2	$Sol.$	x_1	x_2	f_1	f_2
1	0.31	0.89	0.31	6.10	a	0.21	0.24	0.21	5.90
2	0.43	1.92	0.43	6.79	b	0.79	2.14	0.79	3.97
3	0.22	0.56	0.22	7.09	c	0.51	2.32	0.51	6.51
4	0.59	3.63	0.59	7.85	d	0.27	0.87	0.27	6.93
5	0.66	1.41	0.66	3.65	e	0.58	1.62	0.58	4.52
6	0.83	2.51	0.83	4.23	f	0.24	1.05	0.24	8.54

Tabla 2.1. Ejemplo de población progenitora y descendiente. Fuente (Deb, 2001a)

1. Combinamos las poblaciones P_t y Q_t para formar $R_t = \{1,2,3,4,5,6, a, b, c, d, e, f\}$. Después, realizamos una ordenación de no dominados en R_t, obteniendo los distintos frentes en orden de dominancia (Figura 2.3):

$$\mathcal{F}_1 = \{5, a, e\}$$
$$\mathcal{F}_2 = \{1,3, b, d\}$$
$$\mathcal{F}_3 = \{2,6, c, f\}$$
$$\mathcal{F}_4 = \{4\}$$

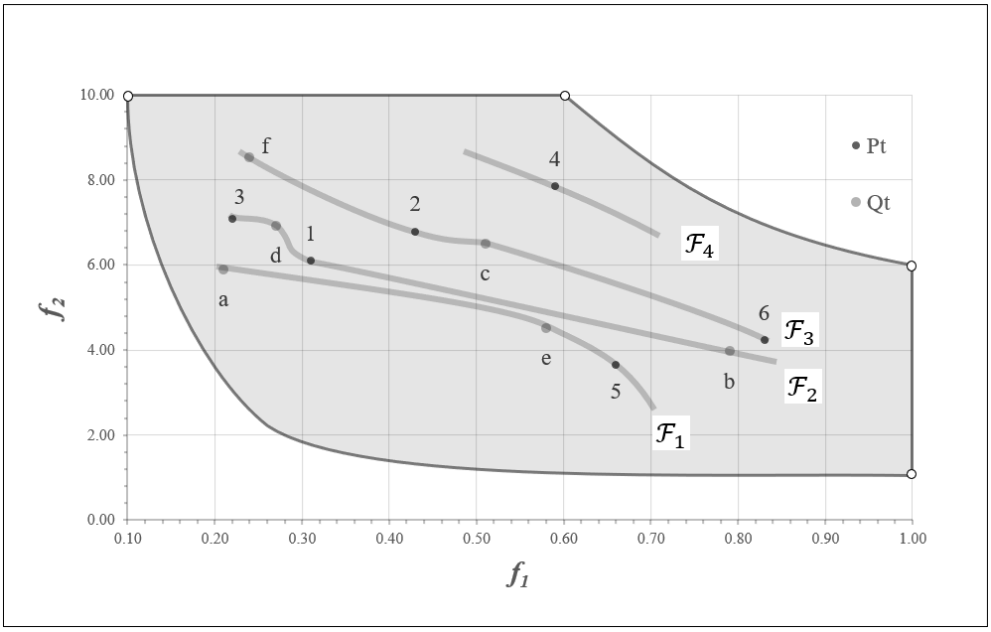

Figura 2.3. Fronteras no dominadas en la población combinada Rt. Fuente (Deb, 2001a)

2. Fijamos $P_{t+1} = \emptyset$ y $i = 1$. Ahora comienza el emplazamiento por orden de dominancia, esto es, $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$. Comenzamos con la frontera $\mathcal{F}_1 = \{a, e, 5\}$ con $|\mathcal{F}_1| = 3$. Ahora, $|P_{t+1}| = 0 + 3 = 3$. Como es un tamaño inferior al de la población, $N = 6$, continuamos con la siguiente frontera, $\mathcal{F}_2 = \{3, d, 1, b\}$ con $|\mathcal{F}_2| = 4$. Con la inclusión de este segundo frente, ahora $|P_{t+1}| = 3 + 4 = 7$. Como es un valor mayor al del tamaño de la población, paramos el proceso de inclusión de fronteras. Teniendo en cuenta que el tamaño de la población es conocido, se podría haber ahorrado el paso de ordenación de las fronteras $\mathcal{F}_3$ y $\mathcal{F}_4$.

3. Para seleccionar las soluciones a emplazar en P_{t+1} de la última frontera escogida, se inicia el proceso de ordenación de multitudes de la sección 2.1.3, utilizando el operador $<_c$.
    a. Se tiene que $l = |\mathcal{F}_2| = 4$, por lo que se inicializan las distancias $d_3, d_d, d_1, d_b = 0$. También se fijan los valores máximos $f_1^{max} = 1, f_1^{min} = 0.1, f_2^{max} = 60, f_2^{min} = 0$.
    b. Para la primera función objetivo, la ordenación de las soluciones es $I^1 = \{3, d, 1, b\}$, como se muestra en la Tabla 2.2.
    c. Como las soluciones 3 y b son soluciones límite para f_1, fijamos las distancias $d_3 = d_b = \infty$. Para las dos soluciones restantes, se tiene:

$$d_d = d_d + \frac{f_1^{(1)} - f_1^{(3)}}{f_1^{max} - f_1^{min}} = 0 + \frac{0.31 - 0.22}{1.00 - 0.10} = 0.10$$

$$d_1 = d_1 + \frac{f_1^{(b)} - f_1^{(d)}}{f_1^{max} - f_1^{min}} = 0 + \frac{0.79 - 0.27}{1.00 - 0.10} = 0.58$$

    d. Para la segunda función objetivo, fijamos $d_b = d_3 = \infty$ y las otras dos distancias son:

$$d_1 = d_1 + \frac{f_2^{(d)} - f_2^{(b)}}{f_2^{max} - f_2^{min}} = 0.58 + \frac{6.93 - 3.97}{60.00 - 0.00} = 0.63$$

$$d_d = d_d + \frac{f_2^{(3)} - f_2^{(1)}}{f_2^{max} - f_2^{min}} = 0.10 + \frac{7.09 - 6.10}{60.00 - 0.00} = 0.12$$

    La distancia grupal media para estas soluciones es:

$$d_1 = 0.63, \quad d_3 = \infty, \quad d_b = \infty, \quad d_d = 0.12$$

    Los cuboides (rectángulos para este caso) están representados en la Figura 2.4.

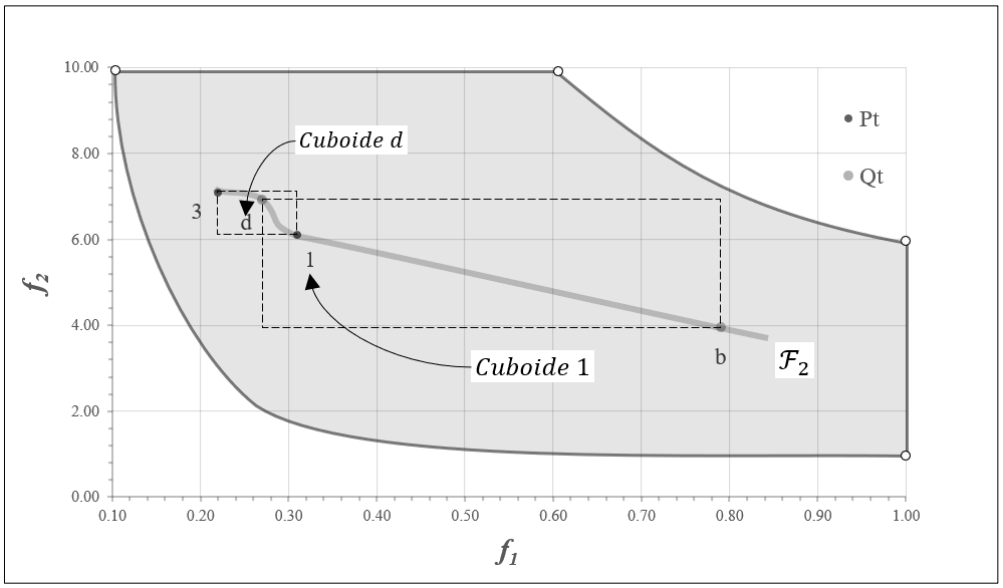

Figura 2.4. Cuboides para las soluciones 1 y d. Fuente (Deb, 2001a)

3. Ordenamos las soluciones en sentido descendente de acuerdo a sus distancias grupales, obteniendo la lista ordenada $\{3, b, 1, d\}$. Escogemos las tres primeras soluciones.
4. La nueva población es $P_{t+1} = \{5, a, e, 3, b, 1\}$. Ahora se crea la población Q_{t+1} utilizando esta población progenitora. Su configuración exacta dependerá de la elección del par de soluciones que participan en el torneo y los operadores de cruce y mutación seleccionados. Se van a emparejar las soluciones $(5, e), (a, 3), (1, b), (a, 1), (e, b)$ y $(3, 5)$, de forma que cada solución participa dos veces en el torneo. Los valores de la distancia grupal para los miembros de la primera frontera también se listan en la Tabla 2.2. En el primer torneo, las soluciones 5 y e

pertenecen a la misma frontera ($r_5 = r_e = 1$), por lo que escogemos al que tenga un mayor valor de distancia grupal, resultando como vencedora del torneo la solución 5.

Una vez finalizados los torneos, se obtiene la piscina de apareamiento siguiente: $\{5, a, a, b, b, e\}$. Estas soluciones pasarán al proceso de cruce y mutación para obtener Q_{t+1}, completando así una iteración del algoritmo NSGA-II.

	$Frontera\ \mathcal{F}_1$				$Ordenación$		$Distancia$
$Sol.$	x_1	x_2	f_1	f_2	f_1	f_2	
5	0.66	1.41	0.66	3.65	$tercera$	$primera$	∞
a	0.21	0.24	0.21	5.90	$primera$	$tercera$	∞
e	0.58	1.62	0.58	4.52	$segunda$	$segunda$	0.54
	$Frontera\ \mathcal{F}_2$				$Ordenación$		$Distancia$
$Sol.$	x_1	x_2	f_1	f_2	f_1	f_2	
1	0.31	0.89	0.31	6.10	$tercera$	$segunda$	0.63
3	0.22	0.56	0.22	7.09	$primera$	$cuarta$	∞
b	0.79	2.14	0.79	3.97	$cuarta$	$primera$	∞
d	0.27	0.87	0.27	6.93	$segunda$	$tercera$	0.12

Tabla 2.2. Asignación de fitness en NSGA-II. Fuente (Deb, 2001a)

2.2. SPEA2 – Zitzler E., Laumanns M., Thiele L.

Algoritmo elitista que nace de la mejora de su versión anterior, SPEA2 emplea una estrategia que potencia la asignación de fitness y nuevas técnicas de truncamiento de archivo y selección basada en la densidad de la población, consiguiendo mejorar el rendimiento con respecto a su predecesor.

El funcionamiento del algoritmo es el siguiente:

En la iteración inicial $t = 0$ se genera una población inicial aleatoria $\boldsymbol{P}_o$ de tamaño N y se crea un conjunto externo (vacío) $\overline{\boldsymbol{P}}_o = \emptyset$ de tamaño $\overline{N}$.

Tras generar a la población inicial entramos en la fase de evaluación donde evaluamos la aptitud o fitness de ambos conjuntos $\boldsymbol{P}_t$ y $\overline{\boldsymbol{P}}_t$, teniendo así en cuenta tanto las soluciones dominadas como las no dominadas.

Ya con la asignación de fitness, se procede con la selección ambiental. Se genera el conjunto $\overline{\boldsymbol{P}}_{t+1}$ copiando todos los individuos no dominados en $\boldsymbol{P}_t$ y $\overline{\boldsymbol{P}}_t$ Si el tamaño de $\overline{\boldsymbol{P}}_{t+1}$ excede $\overline{N}$, entonces se reduce el conjunto con el operador de truncamiento. En otro caso, si el tamaño de $\overline{\boldsymbol{P}}_{t+1}$ es menor que $\overline{N}$, entonces se completa $\overline{\boldsymbol{P}}_{t+1}$ con individuos dominados de $\boldsymbol{P}_t$ y $\overline{\boldsymbol{P}}_t$. En este punto, si t alcanza el número máximo de generaciones, $t \geq T$, o se satisface otro criterio de parada, entonces se define el conjunto $\boldsymbol{A}$ como el conjunto de vectores de decisión representados por los individuos no dominados en $\overline{\boldsymbol{P}}_t$.

Si el número de generaciones es inferior al máximo, se realiza la selección para el apareamiento. Para ello se realiza un torneo binario de selección con reemplazamiento en $\overline{\boldsymbol{P}}_t$ para ocupar la piscina de apareamiento.

Se aplican ahora los operadores de cruce y mutación en la piscina de apareamiento para producir $\boldsymbol{P}_{t+1}$. En este momento termina el bucle principal del algoritmo, por lo que se incrementa el valor de t en una unidad y se vuelve al paso de asignación de fitness, esta vez con el nuevo conjunto generado.

En la Figura 2.5 se muestra un diagrama de flujo del algoritmo descrito.

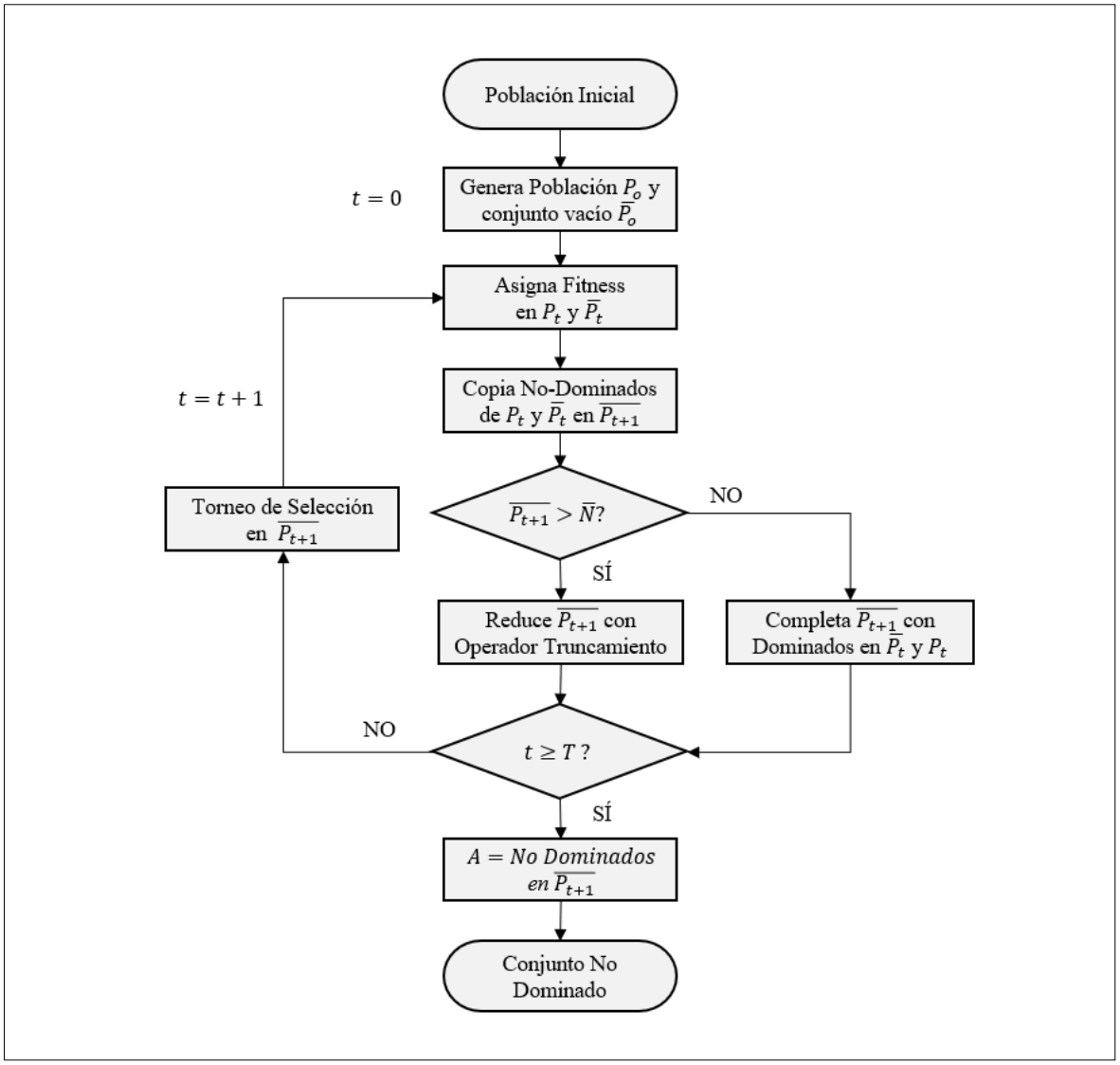

Figura 2.5. SPEA2 Diagrama de Flujo. Fuente (Zitzler et al., 2001)

2.2.1. Asignación de Fitness

A cada individuo $\boldsymbol{i}$ en el archivo $\overline{\boldsymbol{P}}_t$ y en la población $\boldsymbol{P}_t$ se le asigna un valor de fortaleza (strength value) $S(\boldsymbol{i})$, que representa el número de soluciones que domina:

$$S(N) = |\boldsymbol{j}|\boldsymbol{j} \in \boldsymbol{P}_t + \overline{\boldsymbol{P}}_t, \boldsymbol{i} \succ \boldsymbol{j}| \tag{2.2}$$

El operador $|\cdot|$ representa la cardinalidad del conjunto (sus elementos), $+$ es la unión de conjuntos y $\succ$ es la relación de dominancia. Utilizando como base el valor de la fortaleza, se obtiene la aptitud bruta $R(\boldsymbol{i})$ como:

$$R(\boldsymbol{i}) = \sum_{\boldsymbol{j} \in \boldsymbol{P}_t + \overline{\boldsymbol{P}}_t, \boldsymbol{j} \succ \boldsymbol{i}} S(\boldsymbol{j}) \tag{2.3}$$

De esta manera, un individuo con $R(\boldsymbol{i}) = 0$ corresponde a un individuo no dominado, mientras que un valor elevado en $R(\boldsymbol{i})$ denota que $\boldsymbol{i}$ está dominado por muchos, como se muestra en la Figura 2.6.

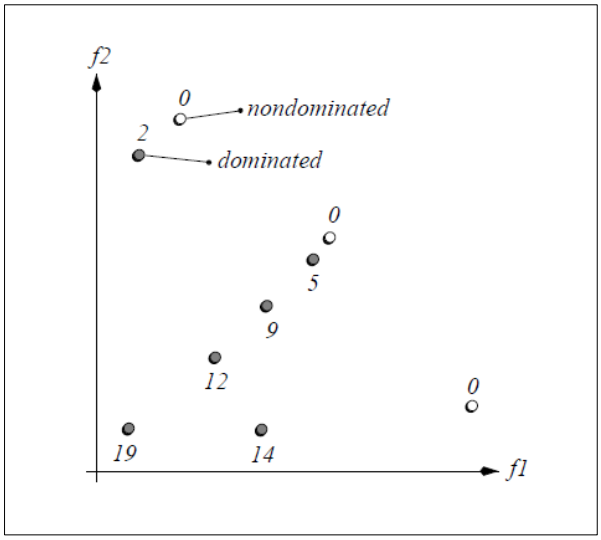

Figura 2.6. Asignación de Fitness en SPEA2. Fuente (Zitzler et al., 2001)

Para evitar fallos cuando la mayoría de los individuos son no dominados, se incorpora información de densidad para discriminar entre individuos con valores idénticos de fitness. El método para estimar la densidad en SPEA2 es una adaptación del método de Silverman del *k-ésimo vecino más cercano*, donde describe la densidad en cualquier punto mediante una función decreciente de la distancia al k-ésimo punto más cercano. SPEA2 toma simplemente el inverso de esta distancia como estimación de la densidad. Así, para cada individuo i se calculan y ordenan de forma creciente las distancias a todos los individuos $\boldsymbol{j}$ en el archivo y la población. En dicha lista, el elemento k-ésimo da la distancia buscada, denotada como $\sigma_{\boldsymbol{i}}^{k}$, utilizando un valor de $k = \sqrt{N + \overline{N}}$. Con todo, la función densidad queda:

$$D(\boldsymbol{i}) = \frac{1}{\sigma_{\boldsymbol{i}}^{k} + 2} \tag{2.4}$$

Se suma dos en el denominador para asegurar que es mayor que cero y que $D(\boldsymbol{i}) < 1$. Sumando la función densidad a la aptitud bruta de un individuo $\boldsymbol{i}$ se obtiene la función fitness $\mathrm{F}(\boldsymbol{i})$:

$$F(\boldsymbol{i}) = R(\boldsymbol{i}) + D(\boldsymbol{i}) \tag{2.5}$$

2.2.2. Selección Ambiental

La operación de actualización del archivo en SPEA2 se realiza manteniendo constante el número de individuos en el archivo a lo largo del tiempo y utilizando un operador de truncamiento para prevenir la eliminación de soluciones límite.

Durante la selección ambiental el primer paso es copiar todos los individuos no dominados del archivo y la población al archivo de la siguiente generación.

$$\overline{\boldsymbol{P}}_{t+1} = \{\boldsymbol{i} | \boldsymbol{i} \in \boldsymbol{P}_t + \overline{\boldsymbol{P}}_t, F(\boldsymbol{i}) < 1\} \tag{2.6}$$

Si el frente no dominado encaja exactamente en el archivo $\overline{\boldsymbol{P}}_{t+1}$ de tamaño $\overline{N}$, se completa el paso de selección. En otro caso puede haber dos situaciones: no se ha completado el archivo o se ha excedido el tamaño máximo permitido. En el primer caso, se seleccionan los mejores individuos dominados en el anterior archivo y se copian en el nuevo. Esto se puede conseguir ordenando el conjunto $\boldsymbol{P}_t + \overline{\boldsymbol{P}}_t$ según los valores de fitness y copiando los primeros individuos con $F(\boldsymbol{i}) \geq 1$ hasta completar el archivo. En el segundo caso, cuando el tamaño del conjunto actual de no dominados excede el tamaño $\overline{N}$ se invoca al proceso de truncamiento de archivo, que elimina individuos iterativamente del conjunto hasta que alcanza el tamaño máximo permitido. La condición para que el individuo $\boldsymbol{i}$ sea eliminado con respecto al individuo $\boldsymbol{j}$ es:

$$\boldsymbol{i} \leq_d \boldsymbol{j} \quad :\Leftrightarrow \quad \forall\, 0 < k < |\overline{\boldsymbol{P}}_{t+1}| : \sigma_{\boldsymbol{i}}^k = \sigma_j^k \;\; \exists\, 0 < k < |\overline{\boldsymbol{P}}_{t+1}| : \left[\left(\forall\, 0 < l < k \;:\; \sigma_{\boldsymbol{i}}^l = \sigma_j^l\right) \wedge \sigma_{\boldsymbol{i}}^k < \sigma_j^k\right] \tag{2.7}$$

Donde σ_i^k representa la distancia del individuo i al k-ésimo vecino más cercano en $\overline{\boldsymbol{P}}_{t+1}$. Se escoge al individuo que tiene la distancia mínima a otro en cada iteración; si hay varios con la distancia mínima se considera entonces la siguiente mayor distancia y sucesivas. El proceso se ilustra en la Figura 2.7:

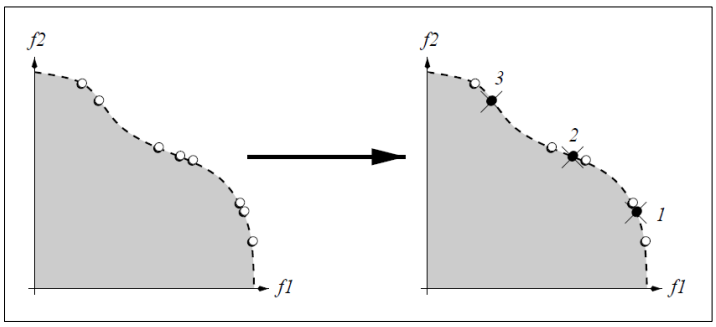

Figura 2.7. Proceso de truncamiento en SPEA2. Fuente:(Zitzler et al., 2001)

2.3. MOEA/D – Qingfu Zhang

MOEA/D descompone explícitamente el problema MOP en N subproblemas de optimización escalar y resuelve estos problemas de forma simultánea evolucionando a una población de soluciones. En cada generación, la población se compone de la mejor solución encontrada para cada subproblema desde el inicio del algoritmo y la relación de vecindad entre estos subproblemas se define mediante la distancia euclídea entre sus vectores de coeficientes de agregación, por lo que las soluciones óptimas a dos subproblemas que sean vecinos deberían ser similares. Así, cada subproblema o función de agregación escalar se optimiza utilizando información que proviene exclusivamente de los subproblemas vecinos.

2.3.1. Descomponiendo el MOP

Recordando la formulación de un problema multiobjetivo (MOP), podemos escribir:

$$\begin{aligned} &Minimizar && F(x) = \left(f_1(x), \ldots, f_m(x)\right)^T \\ &Sujeto\ a && x \in \Omega \end{aligned} \tag{2.8}$$

Donde Ω es el espacio de decisión, $F: \Omega \longrightarrow \mathbb{R}^{\mathrm{m}}$ son las m funciones objetivo y $\mathbb{R}^{\mathrm{m}}$ es el espacio objetivo. La región factible del espacio objetivo es el conjunto $\{F(x) | x \in \Omega\}$.

Existen varios métodos para descomponer el MOP en un determinado número de subproblemas de optimización escalar:

1. *Método de sumas ponderadas* (*Weighted Sum*): Se define un vector de pesos $\lambda = (\lambda_1, \ldots, \lambda_m)^T$ con $\lambda_i \geq 0$ para $i = 1, \ldots, m$ y de forma que la suma de los elementos de este vector sea uno. Ahora se reescribe el problema de optimización como:

$$\begin{aligned} &Minimizar && g^{ws}(x|\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x) \\ &Sujeto\ a && x \in \Omega \end{aligned} \tag{2.9}$$

2. *Método de Tchebycheff*: El problema de optimización escalar es de la forma:

$$\begin{aligned} &Minimizar && g^{te}(x|\lambda, z^*) = \max_{1 \leq i \leq m} \{\lambda_i |f_i(x) - z_i^*|\} \\ &Sujeto\ a && x \in \Omega \end{aligned} \tag{2.10}$$

Donde $z^* = (z_1^*, \dots, z_m^*)^T$ es un punto de referencia definido anteriormente como vector objetivo ideal y sus coeficientes son $z_i^* = \min\{f_i(x) | x \in \Omega\}$. Para cada óptimo de Pareto x^*existe un vector de pesos λ tal que x^* es la solución óptima al subproblema (2.9) y cada solución óptima de (2.9) es una solución óptima de Pareto del MOP (2.7).

3. *Método de intersección de límites (Boundary Intersection)*: Geométricamente, estos métodos tratan de buscar puntos de intersección de un conjunto de rectas con los valores extremos. Si estas rectas están distribuidas de forma homogénea se puede esperar una buena distribución de los puntos del frente de Pareto.

$$\begin{aligned} &Minimizar && g^{bi}(x|\lambda, z^*) = d \\ &Sujeto\ a && F(x) - z^* = d\lambda, \\ & && x \in \Omega \end{aligned} \tag{2.11}$$

Estos métodos se pueden utilizar para descomponer la aproximación del Frente de Pareto en un número de problemas de optimización escalares. Un número razonablemente elevado de vectores de ponderación homogéneamente distribuidos normalmente conduce a un conjunto de óptimos de Pareto que no tienen por qué estar homogéneamente distribuidos, aunque sí aproximarse bien al frente.

2.3.2. Marco General

Disponiendo de un conjunto de vectores de ponderación $\lambda^1, \dots, \lambda^N$ homogéneamente distribuidos y un punto de referencia z^* podemos descomponer el problema MOP en N subproblemas de optimización escalar utilizando, en este caso, el *método de Tchebycheff*, de forma que la función objetivo del subproblema $j - ésimo$ es:

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

con $\lambda_j = \left(\lambda_1^j, \dots, \lambda_m^j\right)^T$.

En MOEA/D se define una comunidad de vecinos de vectores de ponderación λ^i formada por los N elementos más cercanos $\{\lambda_1, \dots, \lambda_N\}$. Así, la comunidad del subproblema $i - ésimo$ consiste en todos los subproblemas con los vectores de peso del vecindario de λ^i. La población se compone de las mejores soluciones encontradas hasta el momento para cada subproblema. Sólo las soluciones a subproblemas vecinos son explotadas para optimizar el subproblema actual.

En cada generación t, se mantiene:

- Una población de N puntos $x^1, \dots, x^N \in \Omega$, donde x^i es la solución al $i - ésimo$ subproblema.
- $FV^1, \dots, FV^N$donde FV^i es el valor de fitness de x^i, esto es, $FV^i = F\left(x^i\right)$ para cada $i = 1, \dots, N$.
- $z = (z_1, \dots, z_m)^T$ donde z_i es el mejor valor encontrado hasta el momento para el objetivo f_i.
- Una población externa EP utilizada para almacenar las soluciones no dominadas encontradas durante la búsqueda.

2.3.3. Funcionamiento

El algoritmo funciona de la siguiente forma:

Entradas:

- MOP;
- Criterio de parada
- N: Número de subproblemas considerados en MOEA/D
- Una distribución uniforme de N vectores de ponderación $\lambda^1, \dots, \lambda^N$.
- T: Número de vectores ponderación en el vecindario de cada vector ponderación.

Salidas: Población externa EP

Paso 1. Inicialización

1.1. Fijar $EP = \emptyset$

1.2. Calcular las distancias euclídeas entre dos vectores ponderación cualesquiera y obtener los T vectores más cercanos a cada vector ponderación. Para cada $i = 1, \dots, N$, se fija $B(i) = \{i_1, \dots, i_T\}$, donde $\lambda^{i_1}, \dots, \lambda^{i_T}$ son los T vectores de ponderación más cercanos a λ^i.

1.3. Generar una población inicial $x^1, \dots, x^N$ de forma aleatoria o por un método específico del problema. Fijar $FV^i = F(x^i)$.

1.4. Inicializar $z = (z_1, \dots, z_m)^T$ por un método específico del problema.

Paso 2. Actualización:

De forma iterativa, para cada $i = 1, \dots, N$:

2.1. Reproducción: Seleccionar dos índices k, l de forma aleatoria de $B(i)$ y generar la nueva solución y a partir de x^k y x^l mediante operadores genéticos.

2.2. Mejora: Aplicar una reparación o mejora heurística en y para producir a y'

2.3. Actualización de z: Para cada $j = 1, \dots, m$, si $z_j < f_j(y')$, entonces fija $z_j = f_j(y')$.

2.4. Actualización de soluciones vecinas: Para cada índice $j \in B(i)$, si $g^{te}(y'|\lambda^j, z) \leq g^{te}(x^j|\lambda^j, z)$, entonces se establece que $x^j = y'$ y que $FV^j = F(y')$.

2.5. Actualización de EP: Eliminar de EP a todos los vectores dominados por $F(y')$, así como agregar $F(y')$ a EP si no hay vectores en EP que dominen a $F(y')$.

Paso 3. Criterio de Parada: Si se satisface el criterio de parada establecido, para e imprime la población externa EP. En otro caso, vuelve al **Paso 2**

2.3.4. Operadores Genéticos y Pérdida de Diversidad

Para comprobar la eficacia de MOEA/D, en el estudio de Zhang se utilizaron como operadores genéticos de cruce y mutación, el operador SBX (*Simulated Binary Crossover*) propuesto en (Deb et al., 2007) y como operador de cruce la mutación polinómica. Ambos operadores se implementaron en MOEA/D y NSGA-II en una gama de problemas de test, mostrando buenos resultados a favor de MOEA/D en términos de velocidad de evaluación de individuos y convergencia.

Sin embargo, en un estudio más reciente (Li & Zhang, 2009) se ha probado que, para problemas de test con fronteras de Pareto complejas, los resultados obtenidos en MOEA/D con esta estrategia de cruce y mutación no son favorables.

Una de las razones de que esto ocurra podría encontrarse en la pérdida de diversidad, necesaria para explorar el espacio de búsqueda de forma efectiva, particularmente en las primeras etapas del proceso de búsqueda.

MOEA/D, en su forma tradicional de realizar estas operaciones, obtiene buenos resultados para problemas con frentes sencillos, pero no consigue obtener una buena distribución del frente cuando la forma de éste es compleja.

2.3.5. Estrategia de cruce: Differential Evolution Crossover (MOEA/D-DE)

Para prevenir la pérdida de diversidad en problemas donde la frontera de Pareto tiene formas complejas es importante definir una estrategia de cruce y mutación que se ajuste a esta complejidad. Para ello se define la estrategia de reproducción de evolución diferencial (DE), que utiliza un operador de evolución diferencial junto con un operador de mutación polinómica para producir nuevas soluciones, además de contar con dos medidas extra para mantener la diversidad de la población.

Paso 1. Inicialización

1.1. Se calculan las distancias euclídeas entre parejas de vectores de ponderación y se seleccionan los T vectores más cercanos a cada uno. Para cada $i = 1, \dots, N$, se define $B(i) = \{i_1, \dots, i_T\}$, donde $\lambda^{i_1}, \dots, \lambda^{i_T}$ son los T vectores ponderación vecinos de λ^i.

1.2. Se genera una población inicial $x^1, \dots, x^N$ muestreando de forma aleatoria de Ω y se obtienen los valores $FV^i = F(x^i)$.

1.3. Se inicializa $z = (z_1, \dots, z_m)^T$ de forma que $z_j = \min_{1 \le i \le N} f_j(x^i)$.

Paso 2. Actualización

Para cada subproblema $i = 1, \dots, N$

2.1. Selección del rango de apareamiento y actualización: Se genera un número aleatorio $rand$ uniforme tal que $rand \in [0, 1]$. A continuación, se define el conjunto P tal que:

$$P = \begin{cases} B(i) & si\ rand < \delta \\ \{1, \dots, N\} & en\ otro\ caso \end{cases}$$

2.2. Reproducción: Se fija al primer reproductor $r_1 = i$ y se seleccionan dos índices aleatorios r_1 y r_2 de P. Después se genera la solución $\bar{y}$ a partir de los individuos x^{r_1}, x^{r_2} y x^{r_3} utilizando el operador de evolución diferencial (DE). A continuación, con una probabilidad p_m se utiliza el operador de mutación polinómica en $\bar{y}$ para producir la nueva solución y.

2.3. Reparación: Si un elemento de y está fuera del límite de Ω, se reinicia su valor a uno seleccionado de forma aleatoria dentro de los límites del espacio de búsqueda.

2.4. Actualización de z: Si a lo largo de las iteraciones se encuentra un valor de $f_j(y')$ que esté por debajo del correspondiente z_j, se actualiza el valor de este último reemplazándolo. Es decir, para cada $j = 1, \dots, m$, si $z_j > f_j(y)$, entonces $z_j = f_j(y)$.

2.5. Actualización de las soluciones: Se fija $c = 0$ y se hace lo siguiente:

1) Si $c = n_r$ o P está vacío, se avanza hasta el paso 3. En otro caso, se selecciona de forma aleatoria un índice j de P.
2) Si $g^{te}(y|\lambda^j, z) \le g^{te}(x^j|\lambda^j, z)$, entonces se fija $x^j = y$ y $FV^j = F(y)$. Actualiza $c = c + 1$.
3) Elimina el índice j de P y vuelve al paso 1).

Paso 3. Criterio de Parada: Si se satisface el criterio de parada, saca como resultado $\{x^1, \dots, x^N\}$ y $\{F(x^1), \dots, F(x^N)\}$. En otro caso se vuelve al paso 2.

En la operación de evolución diferencial, cada elemento $\bar{y}_k$ en $\bar{y} = (\bar{y}_1, \dots, \bar{y}_n)$ se genera de la siguiente forma:

$$\bar{y}_k = \begin{cases} x_k^{r_1} + F \times \left(x_k^{r_2} - x_k^{r_3}\right) & con\ probabilidad\ CR \\ x_k^{r_1} & con\ probabilidad\ 1 - CR \end{cases}$$

Donde F y CR son dos parámetros de control.

Ahora la mutación polinómica genera la solución $y = (y_1, \dots, y_n)$ a partir de $\bar{y}$ de la siguiente forma:

$$y_k = \begin{cases} \bar{y}_k + \sigma_k \times (b_k - a_k) & con\ probabilidad\ p_m \\ \bar{y}_k & con\ probabilidad\ 1 - p_m \end{cases}$$

En donde:

$$\sigma_k = \begin{cases} \bar{y}_k + \sigma_k \times (b_k - a_k) & si\ rand < 0.5 \\ 1 - (2 - 2 \times rand)^{\frac{1}{\eta + 1}} & en\ otro\ caso \end{cases}$$

Tal que $rand$ es un número aleatorio uniforme $rand \in [0, 1]$ y η y p_m con dos parámetros de control denominados índice de distribución y tasa de mutación, respectivamente. a_k y b_k son los límites inferior y superior de la $k - ésima$ variable de decisión.

2.4. Problema de trabajo – Amortiguador de Masa Sintonizado (TMD)

Para probar la eficacia de los algoritmos se ha seleccionado un problema de trabajo objeto de investigación actualmente en el Instituto de las Tecnologías Avanzadas de la Producción de la Universidad de Valladolid, en la línea de investigación del modelado de estructuras esbeltas y calibración de modelos. Se trata de obtener el diseño y localización óptimos en la instalación de amortiguadores de masa sintonizados para edificios de altura variable (Salcedo-Sanz et al., 2017).

Un amortiguador de masa sintonizado o TMD (*Tuned Mass Damper*) es un dispositivo mecánico que reduce la amplitud de las vibraciones absorbiendo la energía cinética del sistema, en este caso las oscilaciones que se producen en un edificio esbelto ante la acción sísmica o la acción del viento, entre otras.

2.4.1. Modelo de un TMD

Los TMDs que se diseñan a día de hoy para los grandes edificios son de tipo péndulo o masa móvil (Magdaleno, 2017) pudiéndose modelar como un sistema mecánico de un grado de libertad constituido por una masa móvil m_j solidaria a la estructura S mediante un medio elástico k_j y un elemento amortiguador c_j, como se muestra en la Figura 2.8.

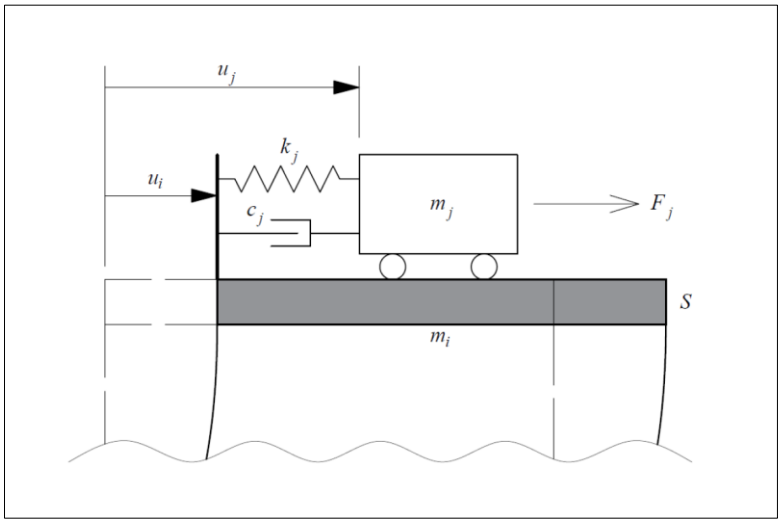

Figura 2.8. Definición del modelo de un TMD. Fuente (Magdaleno, 2017)

Aplicando la Segunda Ley de Newton a la masa móvil se obtiene la ecuación del movimiento del TMD:

$$m_j\ddot{u}_j + c_j(\dot{u}_j - \dot{u}_\iota) + k_j(u_j - u_i) = F_j \tag{2.13}$$

La reacción R del TMD se obtiene de la aplicación de la misma ley al elemento elástico k_j y al amortiguador c_j:

$$c_j(\dot{u}_\iota - \dot{u}_j) + k_j(u_i - u_j) = R_i \tag{2.14}$$

Matricialmente el sistema queda:

$$\begin{bmatrix} 0 & 0 \\ 0 & m_j \end{bmatrix}\begin{bmatrix} \ddot{u}_i \\ \dot{u}_j \end{bmatrix} + \begin{bmatrix} c_j & -c_j \\ -c_j & c_j \end{bmatrix}\begin{bmatrix} \dot{u}_i \\ \dot{u}_j \end{bmatrix} + \begin{bmatrix} k_j & -k_j \\ -k_j & k_j \end{bmatrix}\begin{bmatrix} u_i \\ u_j \end{bmatrix} = \begin{bmatrix} R_i \\ F_j \end{bmatrix} \tag{2.15}$$

Para definir la sintonización de los TMDs se habla de frecuencia propia ω_j y factor de amortiguamiento crítico ξ_j, en lugar de rigidez y constante de amortiguamiento:

$$\omega_j^2 = \frac{k_j}{m_j} \qquad\qquad 2\xi_j\omega_j = \frac{c_j}{m_j} \tag{2.16}$$

2.4.2. Modelo de un edificio de n plantas

Según la Figura 2.9 se establece que cada piso i se desplaza una cantidad u_i horizontalmente con el nivel inferior perfectamente empotrado y su masa, de valor m_i, concentrada en cada uno de ellos. La rigidez entre pisos es k_i y el amortiguamiento estructural queda representado por amortiguadores situados entre plantas, de valor c_i. Aplicando la Segunda Ley de Newton planta por planta se obtiene la Ecuación (2.16):

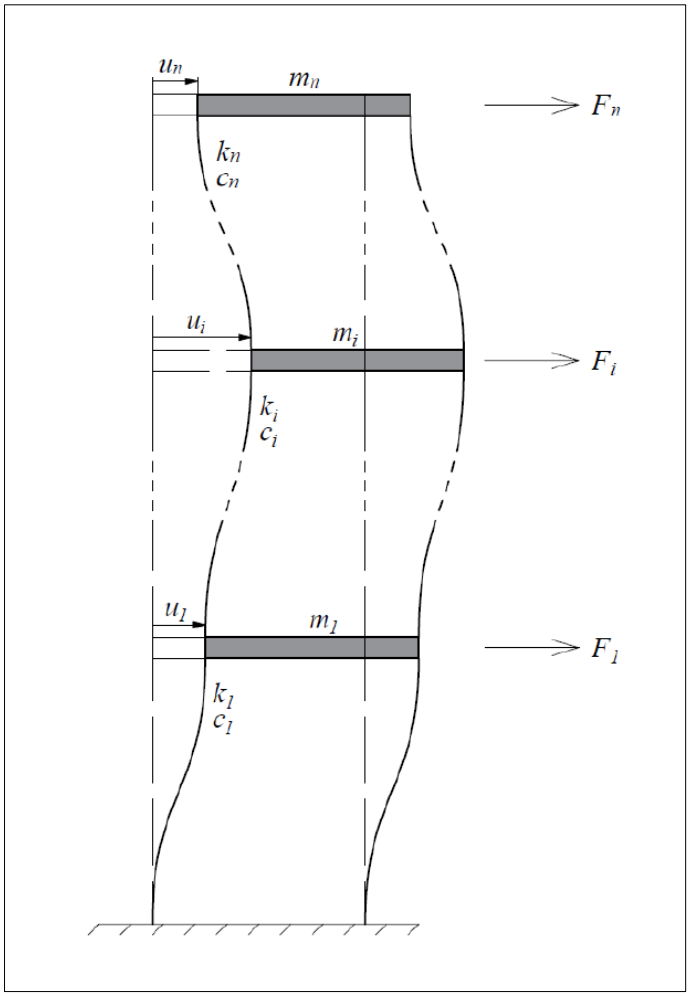

Figura 2.9. Modelo del edificio. Fuente (Magdaleno, 2017)

$$\begin{cases} F_1 = m_1\ddot{u}_1 + (c_1 + c_2)\dot{u}_1 - c_2\dot{u}_2 + (k_1 + k_2)u_1 - k_2u_2 \\ F_2 = m_2\ddot{u}_2 - c_2\dot{u}_1 + (c_2 + c_3)\dot{u}_2 - c_3\dot{u}_3 - k_2u_1 + (k_2 + k_3)u_2 - k_3u_3 \\ \vdots \\ F_i = m_i\ddot{u}_i - c_i\dot{u}_{i-1} + (c_i + c_{i+1})\dot{u}_i - c_{i+1}\dot{u}_{i+1} - k_iu_{i-1} + (k_i + k_{i+1})u_i - k_{i+1}u_{i+1} \\ \vdots \\ F_n = m_n\ddot{u}_n + c_n(\dot{u}_n - \dot{u}_{n-1}) + k_n(u_n - u_{n-1}) \end{cases} \quad (2.17)$$

Considerando la planta de un edificio que lleva instalado un TMD bajo la acción de una fuerza exterior, aplicando la Segunda Ley de Newton se obtiene:

$$F_i - R_i = m_i\ddot{u}_i - c_i\dot{u}_{i-1} + (c_i + c_{i+1})\dot{u}_i - c_{i+1}\dot{u}_{i+1} - k_iu_{i-1} + (k_i + k_{i+1})u_i - k_{i+1}u_{i+1} \quad (2.18)$$

Sustituyendo en esta expresión la reacción del TMD sobre la estructura:

$$F_i = m_i\ddot{u}_i - c_i\dot{u}_{i-1} + \left(c_i + c_{i+1} + c_j\right)\dot{u}_i - c_{i+1}\dot{u}_{i+1} - k_iu_{i-1} + \left(k_i + k_{i+1} + k_j\right)u_i - k_{i+1}u_{i+1} \quad (2.19)$$

A partir de las ecuaciones de movimiento de cada planta y en función de si tiene instalado un TMD o no, se puede reescribir este sistema de forma matricial a partir de las matrices de masa $\mathbb{M}$, rigidez $\mathbb{K}$ y amortiguamiento $\mathbb{C}$:

$$\mathbb{M}\,\ddot{q} + \mathbb{C}\,\dot{q} + \mathbb{K}\,q = \mathbb{F} \quad (2.20)$$

2.4.3. Modelo en espacio de estados

Para abordar este problema en MATLAB resulta útil la representación en espacio de estados. Para este caso, las variables de estado serán la posición, velocidad y aceleración de cada planta y TMD. Luego para N plantas y M TMDs tenemos el vector de estados x:

$$x = [u_1 \quad \cdots \quad u_N \quad u_{t1} \quad \cdots \quad u_{tM} \quad \dot{u}_1 \quad \cdots \quad \dot{u}_N \quad \dot{u}_{t1} \quad \cdots \quad \dot{u}_{tM}]^T \tag{2.21}$$

Así se puede hacer una representación en espacio de estados del problema de la forma:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{2.22}$$

Donde u es el vector de entradas del sistema, A es la matriz de estados, B es la matriz de entradas, D la matriz de transmisión directa y C e y son la matriz de salida y el vector de salidas, respectivamente.

2.4.4. Análisis Frecuencial – FRF (Frequency Response Function)

Para realizar un análisis de la respuesta en frecuencia del sistema se hace uso de la *Función de Respuesta en Frecuencia (FRF)*, que es una curva característica en donde se muestra la relación de la frecuencia de excitación de la estructura con la amplitud de respuesta en un punto de esta.

La función FRF se define como:

$$H^{ib}(\omega) = \frac{X_i(\omega)}{A_b(\omega)} \tag{2.23}$$

De esta función se obtienen las frecuencias propias del sistema, ubicadas en los valores máximos que toma la amplitud de la señal como muestra la Figura 2.10 (a).

El objeto de instalar dispositivos TMD en la estructura es suavizar los picos de la curva, como en la Figura 2.10 (b), para cumplir con los objetivos de seguridad en términos de seguridad estructural y confort de los habitantes. Así, un TMD correctamente sintonizado en torno a una frecuencia propia del sistema provoca el desdoblamiento de la cresta de la curva y, además, si el ajuste de la frecuencia y amortiguamiento del TMD es óptimo, el valor de las crestas será mínimo en el mismo (Guerra, 2017).

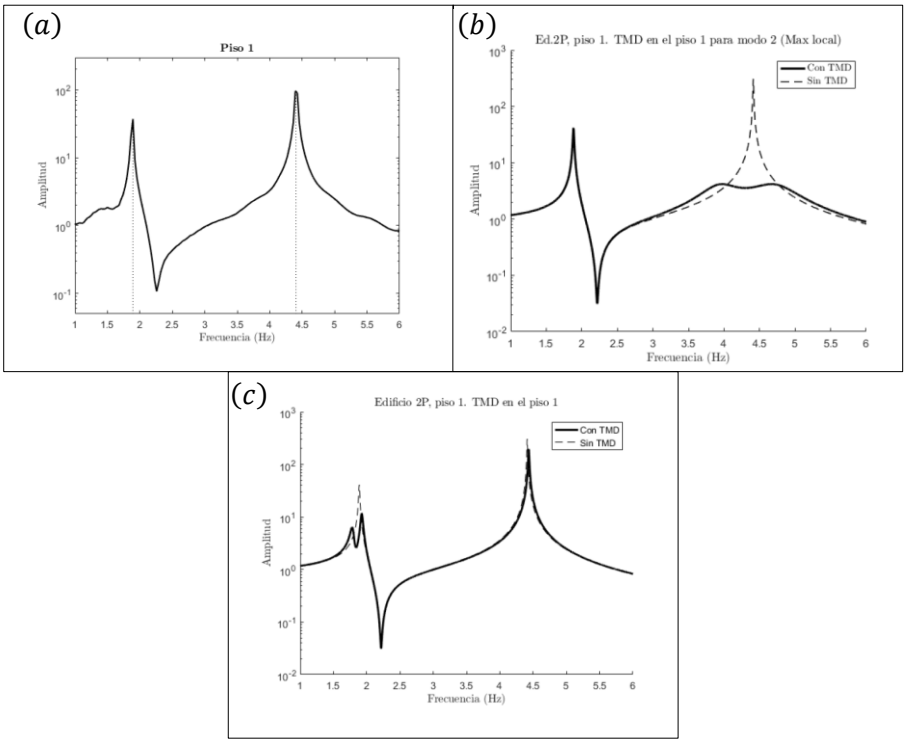

Figura 2.10. (a)FRF sin TMD. (b) FRF con TMD a frecuencia propia. (c) FRF con TMD a frecuencia no óptima. Fuente (Magdaleno, 2017)

Magdaleno, en su estudio sobre la sintonización óptima de TMDs múltiples, define indicadores en el dominio de la frecuencia para conseguir una sintonización óptima, generando una función de aptitud para la sintonización de estos dispositivos.

1. Máximo local de una FRF: Este indicador se define atendiendo al efecto de la instalación de un TMD no óptimo. Cuando esto ocurre, el desdoblamiento de la curva produce dos picos de distinta amplitud, como en la Figura 2.10 (c). El objetivo de este indicador es minimizar el máximo de estos picos, independientemente de cuál sea, provocando el ajuste deseado.

$$\min_x\big(\max\big(H(\omega)\big)\big)/\omega \in [\omega_1, \omega_2] \tag{2.24}$$

2. Máximo global de una FRF: Cuando las frecuencias propias del sistema están próximas entre sí, puede ser preferible optar por una función de coste que retorne el máximo absoluto de una FRF para que el diseño del TMD reduzca al máximo la respuesta global del sistema.

$$\min_x\big(\max\big(H(\omega)\big)\big)/\omega \in [\Omega_1, \Omega_2] \tag{2.25}$$

3. Máximo local de todas las FRF: Complementario al enfoque de optimización basado en máximo local, se basa en extrapolar la búsqueda de dicho máximo en el entorno de una frecuencia propia al conjunto de las FRFs de forma que la función de coste retorne el máximo de todas simultáneamente.

$$\min_x\big(\max\big(H_1(\omega), H_2(\omega), \dots, H_n(\omega)\big)\big)/\omega \in [\omega_1, \omega_2] \tag{2.26}$$

4. Máximo global de todas las FRF: Se incluyen todas las crestas de todas las FRFs en el indicador a minimizar, representando éste a la máxima de todas ellas.

$$\min_x\big(\max\big(H_1(\omega), H_2(\omega), \dots, H_n(\omega)\big)\big)/\omega \in [\Omega_1, \Omega_2] \tag{2.27}$$

3. EXPERIMENTACIÓN Y ANÁLISIS

La experimentación en este trabajo consiste en analizar la eficacia del algoritmo MOEA/D ante un problema multiobjetivo (MOP) basado en la sintonización óptima de amortiguadores de masa TMD (*Tuned Mass Damper*) en un edificio de altura variable. Para ello, se va a hacer una adecuada selección de los parámetros que lo configuran y, posteriormente, se van a contrastar los resultados obtenidos de la Frontera de Pareto con respecto a SPEA2 y NSGA-II, utilizando las métricas descritas en el capítulo 1.

3.1. Modelo de trabajo en el problema TMD

La estructura del problema es la siguiente: Se parte de cuatro modelos del problema para la sintonización de TMD:

- $2P$-1 TMD: Edificio de dos plantas para la sintonización de un TMD.
- $2P$-2 TMD: Edificio de dos plantas para la sintonización de dos TMDs
- $5P$-1 TMD: Edificio de cinco plantas para la sintonización de un TMD.
- $5P$-5 TMD: Edificio de cinco plantas para la sintonización de un TMDs.

La forma de codificar los datos de entrada, y el posterior tratamiento de las soluciones, está diseñada para que el algoritmo resuelva el problema desde el modelo más sencillo ($2P$-1 TMD), pasando por modelos intermedios ($2P$-2 TMD), hasta el más complejo ($5P$-5 TMD).

3.1.1. Fronteras de Pareto complejas

La sencillez del modelo no necesariamente alivia la complejidad del frente obtenido. Más concretamente, tras analizar los primeros resultados de los frentes obtenidos para los diferentes modelos, se ha observado una singularidad en uno de ellos ($2P$-2 TMD), y es que la Frontera de Pareto Óptima (OPF) está formada por dos fronteras Pareto claramente diferenciadas, donde los algoritmos no siempre son capaces de mantener su eficacia para encontrar soluciones no dominadas (Figura 3.1).

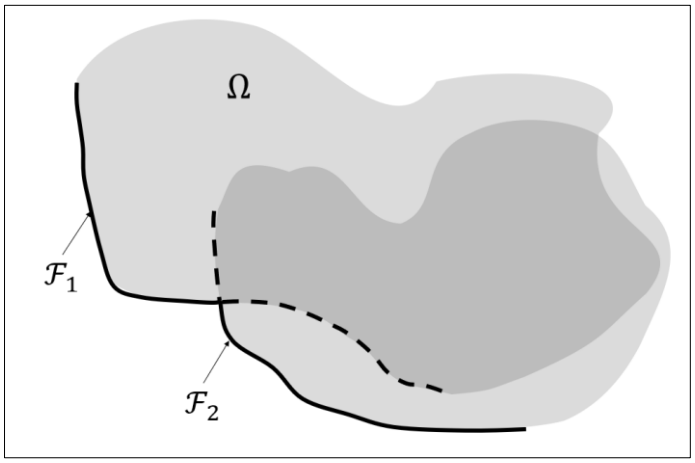

Figura 3.1. Tipología del frente para 2P-2TMD: Unión de fronteras Pareto

La formación de estos dos frentes tiene que ver con la colocación de los TMD, generando así un frente diferenciado cuando las dos masas se colocan en la segunda planta $(2,2)$ y otro cuando las dos masas se colocan en la primera $(1,1)$, unidos por el caso intermedio $(1,2) = (2,1)$. Es por esta complejidad del frente por la que se ha decidido desde un primer momento utilizar el algoritmo MOEA/D, apto para tratar con frentes de tipología compleja (Li & Zhang, 2009).

Por tanto, para la experimentación llevada a cabo en este estudio, se ha hecho uso exclusivo del modelo:

$2P$-2 TMD

3.1.2. Funciones Objetivo

El objeto de la correcta sintonización del TMD es cumplir con los requisitos estructurales de Estado Límite de Servicio (ELS) y Estado Límite Último (ELU)

Como se ha explicado en el anterior capítulo, se ha realizado una representación en espacio de estados donde se trabaja con funciones de respuesta en frecuencia (FRF) para el análisis de la magnitud de las salidas con respecto a las entradas. Se va a trabajar con las aceleraciones como parámetro de salida. Así, cada punto de la FRF representa la relación salida-entrada de cada frecuencia de excitación, y los valores máximos se corresponden con el fenómeno de la resonancia. Se trata por tanto de asumir un rango ancho de banda para la frecuencia de excitación en donde se puedan ver varios picos.

De esta forma, para calcular las aceleraciones, la formulación en espacio de estados debe devolver como salidas las aceleraciones de cada grado de libertad de la estructura.

$$C_{SS} = [-M^{*^{-1}}K^* \quad -M^{*^{-1}}C^*] \qquad D_{SS} = \emptyset \tag{3.1}$$

El indicador de ELS se calcula como el máximo de todos los existentes en la FRF, pero sólo en el rango de interés $[\omega_a, \omega_b]$, teniendo en cuenta la sensibilidad humana a las frecuencias y descartando frecuencias elevadas. Para ello se aplica un factor de ponderación de vibración lateral, W_d, de forma que las frecuencias elevadas van multiplicadas por un factor inferior a la unidad. La función objetivo 1 tendrá la forma:

$$J_{ELS} = \max\left(\max\left(W_d \cdot FRF_i(\omega)\right)\right) \qquad \omega \in [\omega_a, \omega_b] \tag{3.2}$$

La siguiente función objetivo es la que contempla los esfuerzos desarrollados en el interior del material. Así, el objetivo es minimizar este indicador no solamente para asegurar que ningún punto de la estructura alcance el límite de fluencia, sino para reducir la posibilidad de fallo por fatiga. Con la tipología de cargas presentes en el estudio, sólo se consideran los esfuerzos normales, rechazando el efecto de los cortantes. Por otra parte, el valor máximo del esfuerzo normal se encuentra en los extremos del elemento, donde el momento flector es máximo, representado en la Figura 3.2:

$$M = \frac{6EI}{L^2}\delta \tag{3.3}$$

Donde E representa el módulo de Young del material, I es el momento de inercia de segundo orden y L es la longitud del pilar.

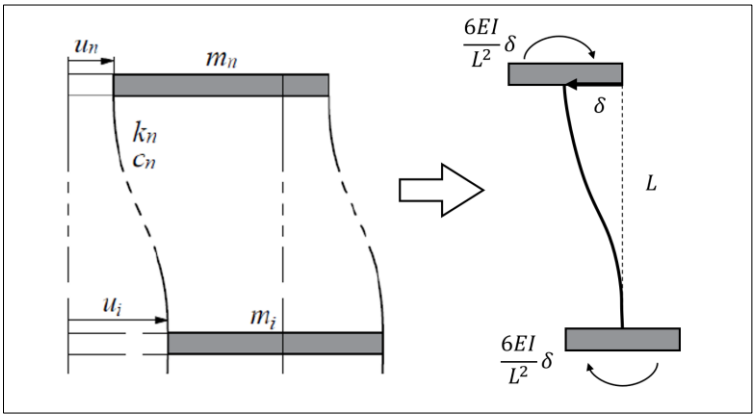

Figura 3.2. Momento flector máximo en pilar biempotrado.

Así, descartando el efecto de la fuerza axial, la cual se suele considerar constante y con un efecto menor al del esfuerzo normal, el esfuerzo normal máximo σ_i está directamente relacionado con el momento flector máximo a través de la altura h de la sección en donde se registra σ_i. Suponiendo dicha altura simétrica de puede definir el esfuerzo máximo en la $i-$ésima planta como:

$$\sigma_i = \frac{M_i}{I_i} y_i^{max} = \frac{M_i h_i}{2I_i} = 3E \frac{h_i}{L_i^2} \delta_i = 3E \Delta_i \delta_i \tag{3.4}$$

Aquí la relación $\Delta = \frac{h_i}{L_i^2}$ es un factor que relaciona el desplazamiento y el esfuerzo normal. Así, minimizar el esfuerzo normal implica minimizar la cantidad adimensional $\Delta_i \delta_i$. Para ello, se obtiene esa magnitud desde una representación en espacio de estados, de forma que $\mathbf{\Delta}$ es una matriz de vectores columna que contienen los factores de relación de cada planta y C_δ es una matriz con los valores apropiados para realizar la multiplicación matricial y obtener la matriz C_{SS}:

$$C_{SS} = \mathbf{\Delta^t}\, C_\delta \qquad donde \qquad \mathbf{\Delta} = \begin{bmatrix} \Delta_1 & \Delta_1 & \dots \\ \Delta_2 & \Delta_2 & \dots \\ \Delta_3 & \Delta_3 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \;\; ; \;\; C_\delta = \begin{bmatrix} 1 & 0 & 0 & \dots \\ -1 & 1 & 0 & \dots \\ 0 & -1 & 1 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{3.5}$$

Una vez que todos los esfuerzos máximos se han calculado, se trata de minimizar el indicador J_{ELU}, que se define como el máximo de todos los máximos de los esfuerzos calculados, en valor absoluto:

$$J_{ELU} = \max_i \left(\max_t (|\sigma_i(t)|) \right) \tag{3.6}$$

Ahora, cuando se analicen los frentes de Pareto obtenidos por cada algoritmo, se graficarán de manera que en el eje de abscisas aparezca el primer objetivo, esto es, J_{ELS}, y en el eje de ordenadas el segundo, J_{ELU}.

3.1.3. Codificación de las soluciones

Cada individuo en el espacio de búsqueda tiene los siguientes genes:

$$\boldsymbol{x} = [m_j \quad \omega_j \quad \xi_j \quad x_j] \qquad con \qquad j = 1,2,\dots,K \tag{3.7}$$

En donde K es el número total de TMDs a sintonizar, m_j son las masas de cada uno, ω_j son las frecuencias de sintonización, ξ_j son los factores de amortiguamiento crítico y x_j son las posiciones en donde se van a instalar (en este caso, las plantas, tomando números enteros).

Las restricciones a las que están sujetas estas variables son:

$$\begin{array}{ll} m_j \in \{0, M\} & valor\ continuo \\ \omega_j \in \{0, 12\} & valor\ continuo \\ \xi_j \in \{0, 1\} & valor\ continuo \\ x_j \in \{1, K\} & valor\ discreto \end{array} \tag{3.8}$$

Donde M es la masa máxima admisible en el TMD bajo el problema de estudio y K es el número de dispositivos TMD a instalar, que coincide con el máximo de alturas que tenga el edificio (número de plantas).

El hecho de que la codificación de la planta en donde se van a instalar los TMD se corresponda con un valor discreto, hace que el gen correspondiente a esta variable no pueda seguir el mismo proceso de cruce y mutación que el resto de los genes (NSGA-II y SPEA2) o, en caso de que sí lo siga (MOEA/D-DE), tenga que pasar por un proceso de reparación o mejora de su valor.

Habiendo definido el espacio de búsqueda, se procede con la definición del espacio objetivo. En este caso, es un espacio bidimensional compuesto por las dos funciones de coste correspondientes a la minimización del indicador de máximos en la FRF, J_{ELS}, y el indicador de esfuerzos, J_{ELU}.

Cada individuo tiene como misión minimizar los dos objetivos que se han propuesto, por lo que la codificación de las soluciones incluye los valores de las funciones objetivo, como se muestra en la Tabla 3.1, con $\boldsymbol{FV}$ el vector de funciones objetivo y $\boldsymbol{\lambda}$ el vector ponderación utilizado en MOEA/D:

$\boldsymbol{m_j}$			$\boldsymbol{\omega_j}$			$\boldsymbol{\xi_j}$			$\boldsymbol{x_j}$			$\boldsymbol{FV}$		$\boldsymbol{\lambda}$
m_1^1	$\cdots$	m_K^1	ω_1^1	$\cdots$	ω_K^1	ξ_1^1	$\cdots$	ξ_K^1	x_1^1	$\cdots$	x_K^1	J_{ELS}^1	J_{ELU}^1	λ^1
$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
m_1^N	$\cdots$	m_K^N	ω_1^N	$\cdots$	ω_K^N	ξ_1^N	$\cdots$	ξ_K^N	x_1^N	$\cdots$	x_K^N	J_{ELS}^N	J_{ELU}^N	λ^N

Tabla 3.1. Codificación de una población de N individuos para la sintonización de K TMDs

3.1.4. Formato de los datos de entrada

Los datos de entrada para el problema del TMD están codificados en distintos ficheros de texto para cada modelo, en donde se tienen parámetros propios del MOP y parámetros propios del algoritmo de uso. Para poder dimensionar los vectores de entrada en cada algoritmo, el primer dato de entrada es la dimensión de estos vectores.

A continuación, los tres primeros argumentos son de tipo vector y son específicos del problema TMD. Contienen, por orden:

- Máximos admisibles en masa del TMD, frecuencias de sintonización, amortiguamiento crítico y planta.
- Mínimos admisibles.
- Tipo de variable que se va a tratar. Necesario durante las operaciones de cruce para conocer el itinerario de cada tipo de gen. La codificación para saber si la variable es continua o discreta es identificada mediante un 0 o un 1, respectivamente.

Los siguientes tres argumentos del fichero son comunes a los tres algoritmos de trabajo:

- N: Número de individuos en la población
- $Eval$: Número de evaluaciones a los individuos
- $Ejec$: Número de ejecuciones del algoritmo

Los argumentos restantes son específicos del algoritmo de trabajo y, por tanto, el fichero tiene un formato diferente si el problema se trata con NSGA-II o SPEA2 (parámetros comunes) a si se hace con el algoritmo MOEA/D-DE. En la Tabla 3.2 se muestra un ejemplo de codificación de los datos de entrada para el problema.

m_1^{max}	$\cdots$	m_K^{max}	ω_1^{max}	$\cdots$	ω_K^{max}	ξ_1^{max}	$\cdots$	ξ_K^{max}	x_1^{max}	$\cdots$	x_K^{max}
m_1^{min}	$\cdots$	m_K^{min}	ω_1^{min}	$\cdots$	ω_K^{min}	ξ_1^{min}	$\cdots$	ξ_K^{min}	x_1^{min}	$\cdots$	x_K^{min}
C/D	$\cdots$	C/D	C/D	$\cdots$	C/D	C/D	$\cdots$	C/D	C/D	$\cdots$	C/D

N

$Eval$

$Ejec$

p_c
p_m
$Cruce$
σ_E

Parámetros propios de NSGA-II y SPEA 2

T
δ
n_r
CR
F
η

Parámetros propios de MOEA/D-DE

Tabla 3.2. Codificación de los datos de entrada para los distintos MOEAs.

3.2. Parámetros de control en los MOEAs

3.2.1. SPEA2 y NSGA-II

La estrategia de cruce y mutación seleccionada es común para estos dos algoritmos. Los operadores de cruce y mutación se utilizan para la etapa de recombinación de soluciones. Son los encargados de mantener equilibrado el balance de exploración y explotación de resultados. Los operadores genéticos no actúan sobre todos los individuos de la población, sino que trabajan con probabilidades. Así, para cada dos soluciones $x^{(1,t)}$ y $x^{(2,t)}$ (progenitores), el operador cruce producirá dos nuevas soluciones $x^{(1,t+1)}$ y $x^{(2,t+1)}$ (descendientes) con una probabilidad p_c. Estos dos descendientes reemplazarán a los progenitores. Después, el operador genético de mutación modificará cada gen del individuo entrante con una probabilidad p_m. Finalmente, el gen mutado reemplaza al gen original.

1) Parámetros de control para el cruce y la mutación:
   - Probabilidad de cruce $p_c = 0.9$
   - Probabilidad de mutación $p_m = 0.1$
2) Operador de cruce:

El operador cruce implementado para las variables continuas del problema es el operador binario simulado o SBX, *Simulated Binary Crossover,* (Deb et al., 2007). Como sugiere su nombre, simula el funcionamiento del operador de cruce en cadenas binarias (Figura 3.3).

A continuación, se define el procedimiento para generar las soluciones descendientes, $x^{(1,t+1)}$ y $x^{(2,t+1)}$, a partir de los progenitores, $x^{(1,t)}$ y $x^{(2,t)}$.

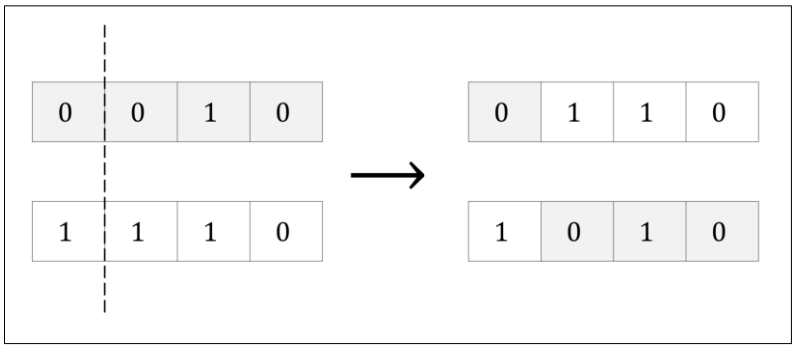

Figura 3.3. Operador de cruce binario en individuos de cuatro genes, representación gráfica.

Se comienza con la definición de un *factor de dispersión* β_i, (*spread factor*), que se define como la relación en valor absoluto de la diferencia de las soluciones de los descendientes entre las de los progenitores:

$$\beta_i = \left| \frac{x_i^{2,t+1} - x_i^{1,t+1}}{x_i^{2,t} - x_i^{1,t}} \right| \tag{3.9}$$

Primeramente, se genera un número aleatorio u_i entre el intervalo $[0,1)$. A continuación, y a partir de una función de distribución de probabilidad específica, se obtiene el valor de la ordenada β_{q_i}, tal que el área bajo la función de distribución entre 0 y β_{q_i} es igual al valor aleatorio generado u_i.

La función de distribución creada tiene una capacidad de búsqueda similar a la utilizada en el cruce binario, y se define como:

$$\mathcal{P}(\beta_i) = \begin{cases} 0.5(\eta+1)\beta_i^{\eta}, & si\ \beta_i \leq 1 \\ 0.5(\eta+1)\dfrac{1}{\beta_i^{\eta+2}} & en\ otro\ caso \end{cases} \tag{3.10}$$

Donde η es un índice de la función de distribución y toma valores reales positivos. Un valor elevado de η aumenta la probabilidad de crear soluciones cercanas en el frente, mientras que un valor bajo contribuye al distanciamiento de las soluciones generadas.

Mediante la ecuación (3.10) se calcula el valor de β_{q_i} igualando el área de la función de distribución al valor de u_i:

$$\beta_{q_i} = \begin{cases} (2u_i)^{\frac{1}{\eta+1}}, & si\ u_i \leq 0.5 \\ \left(\frac{1}{2(1-u_i)}\right)^{\frac{1}{\eta+1}}, & en\ otro\ caso \end{cases} \tag{3.11}$$

Tras obtener el valor de β_{q_i}, las soluciones descendientes se obtienen de la siguiente forma:

$$x_i^{(1,t+1)} = 0.5\left[(1+\beta_{q_i})x_i^{(1,t)} + (1-\beta_{q_i})x_i^{(2,t)}\right] \tag{3.12}$$

$$x_i^{(2,t+1)} = 0.5\left[(1-\beta_{q_i})x_i^{(1,t)} + (1+\beta_{q_i})x_i^{(2,t)}\right] \tag{3.13}$$

Así, el procedimiento se resume en los siguientes tres pasos:

1. Seleccionar un número aleatorio $u_i \in [0,1)$.
2. Calcular β_{q_i} utilizando la ecuación (3.11)
3. Calcular las soluciones descendientes utilizando las ecuaciones (3.12) y (3.13).

Las soluciones descendientes son simétricas respecto de las progenitoras. Esta propiedad se usa de forma intencionada para evitar la parcialidad hacia una solución progenitora particular en cada operación de cruce. Otro aspecto interesante de esta estrategia es que, para un valor establecido de η, la dispersión de las soluciones descendientes es proporcional a la de los progenitores:

$$\left(x_i^{(2,t+1)} - x_i^{(1,t+1)}\right) = \beta_{q_i}\left(x_i^{(2,t)} - x_i^{(1,t)}\right) \tag{3.14}$$

En esencia, el operador SBX tiene dos propiedades:

- La extensión de las soluciones descendientes es proporcional a la de los progenitores
- Soluciones progenitoras cercanas son más probables de ser escogidas como soluciones descendientes que soluciones distantes de sus progenitores.

3) Operador de Mutación:

Para la mutación se ha seleccionado el operador de mutación del Algoritmo Genético Reproductor, *BGA Breeder Genetic Algorithm,* (Mühlenbein & Schlierkamp-Voosen, 1993). El procedimiento es el siguiente:

Se selecciona una variable x_i^j (gen i del individuo j) con una probabilidad p_m para el proceso de mutación. Normalmente se utiliza como probabilidad $p_m = 1/n$ para que, al menos, una variable (gen) pase por mutación.

A continuación, se añade un valor a la variable seleccionada. Este valor agregado se obtiene del intervalo $[-rango_i, +rango_i]$, donde $rango_i$ define el rango de mutación, normalmente establecido en $rango_i = 0.10 \cdot \left(\max\left(x_i^j\right) - \min\left(x_i^j\right)\right)$.

El nuevo valor mutado, z_i, se calcula según:

$$z_i = x_i \pm rango_i \cdot \delta \tag{3.15}$$

Para las variables discretas directamente se toma un valor aleatorio en el dominio de la variable:

$$z_i^{discreta} = aleatorio\left(\max\left(x_i^j\right) - \min\left(x_i^j\right)\right) \quad (3.16)$$

El signo positivo o negativo se selecciona con una probabilidad de 0.50. El valor de δ se obtiene de una función de distribución con preferencia por valores pequeños, de la siguiente forma:

$$\delta = \sum_{i=0}^{15} \alpha_i 2^{-i}, \qquad \alpha_i \in \{0,1\} \quad (3.17)$$

Antes de iniciar el proceso de mutación se inicializa el valor de $\alpha_i = 0$. Después, cada α_i muta de 0 a 1 con una probabilidad $p_\delta = 1/16$. Sólo $\alpha_i = 1$ contribuye a la suma. Con la probabilidad seleccionada, en la media habrá solo un α_i con valor 1, denominado α_j. Luego, el valor de δ vendrá dado por:

$$\delta = 2^{-j} \quad (3.18)$$

El operador BGA es capaz de generar cualquier punto en el hipercubo centrado en x, definido por $x_i \pm rango_i$, aunque suele tomar valores en torno a la comunidad de vecinos de x.

4) Parámetro σ_E: Utilizado en NSGA-II para obtener el radio de nicho y descartar elementos de R_t que vivan en zonas muy pobladas (Sección 2.1.1). También se utiliza en SPEA2 para determinar la distancia del elemento k- ésimo y obtener la densidad (Sección 2.2.1).

3.2.2. MOEA/D-DE

El ajuste de parámetros en este estudio experimental ha sido el siguiente:

1) Parámetros de control para el cruce DE y la mutación polinómica:
   - Tasa de cruce $CR = 1.0$ y $F = 0.5$ para el operador DE.
   - $\eta = 20$ y probabilidad de mutación $p_m = 1/n$ en la mutación polinómica, siendo n el número de variables.
2) Número de variables de decisión:
   - $2P\text{-}2\ TMD \longrightarrow n = 8$.
   - $5P\text{-}1\ TMD \longrightarrow n = 4$
3) Número de ejecuciones y criterio de parada: Cada algoritmo se ejecuta $R = 20$ veces para cada instancia. El algoritmo se detiene cuando llega a un número determinado de evaluaciones, en este caso está fijado a 300.000 evaluaciones.
4) Tamaño de la población y vectores ponderación: Con $N = 300$ individuos en la población, éstos se controlan a través de un número entero H, siendo $\lambda^1, \ldots, \lambda^N$ los vectores ponderación en los que el peso de cada individuo se asigna a través de:

$$\left\{\frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H}\right\} \quad (3.19)$$

   Se ha fijado el valor de $H = 199$ para todas las instancias consideradas y, en consecuencia, $N = 200$
5) Otros parámetros de control:
   - Número de vecinos en la comunidad: $T = 20$
   - Probabilidad en la configuración del conjunto P: $\delta = 0.9$
   - Número máximo de reproducciones: $n_r = 2$

3.3. Presentación de los resultados

Determinar el rendimiento de los distintos algoritmos es una tarea difícil debido a la multitud de Frentes de Pareto a comparar. Debido a su naturaleza estocástica, cada algoritmo se ejecuta veinte veces. Esto implica analizar veinte fronteras Pareto generadas por cada algoritmo y en cada escenario que, en total, suman: $20\frac{\text{Fronteras}}{\text{Escenario}} \cdot 5\frac{\text{Escenarios}}{\text{Problema}} = 100\frac{\text{Fronteras}}{\text{Problema}}$.

Para resolver esta situación, las fronteras Pareto obtenidas en cada ejecución se unen en un conjunto de soluciones donde se eliminan las dominadas, dando lugar al *Frente de Pareto Envolvente* (EPF).

La frontera EPF resulta práctica para analizar a primera vista los resultados obtenidos por cada algoritmo y analizar gráficamente su desempeño en cuanto a la dispersión obtenida, distancia entre soluciones e incluso uniformidad en los resultados según el escenario de trabajo.

Para cuantificar el rendimiento de los algoritmos, en este estudio se van a utilizar valores normalizados de los indicadores de calidad definidos en la Sección 1.3.6. Así, para un número total de R ejecuciones, el valor normalizado de las métricas será:

$$\overline{\boldsymbol{SP}} = \frac{\sum_{r=1}^{R} SP_r}{R} \qquad \overline{\boldsymbol{HPV}} = \frac{\sum_{r=1}^{R} HPV_r}{R} \qquad \overline{\boldsymbol{MS}} = \frac{\sum_{r=1}^{R} MS_r}{R}$$
$$\overline{\boldsymbol{GD}} = \frac{\sum_{r=1}^{R} GD_r}{R} \qquad \overline{\boldsymbol{C(AB)}} = \frac{\sum_{r=1}^{R} C(AB)_r}{R} \tag{3.20}$$

Estas métricas son una herramienta útil para cuantificar la determinación de un algoritmo en cuanto al objetivo fijado, pudiendo centrarse en obtener soluciones lo más cercanas posibles al *Frente de Pareto Óptimo* (OPF), obtener soluciones lo más dispersas y uniformes que sea posible u obtener soluciones no dominadas que pertenezcan al frente Pareto de forma robusta y no ocasional.

Para contabilizar las soluciones obtenidas por cada algoritmo y comparar su calidad, a mayores se hace un análisis mediante *métricas cardinales*. Estas métricas hacen referencia al número de soluciones en la frontera Pareto, y su uso se extiende al análisis de las fronteras:

- $|PF(A)|$: Cardinalidad de la frontera de Pareto (PF) obtenida en una ejecución, mediante el algoritmo A.
- $|EPF(A)|$: Cardinalidad de la envolvente de todas las fronteras, *Envelope Pareto Front (EPF)*, obtenidas con el algoritmo A.
- $|OPF(A)|$: Cardinalidad de la Frontera de Pareto óptima (OPF).

Al igual que en el caso de las métricas vistas hasta el momento, también se va a normalizar el resultado para su análisis, salvo el caso de la cardinalidad envolvente por carecer de sentido práctico.

$$\left|\overline{PF(A)}\right| = \frac{\sum_{r=1}^{R}(|PF(A)|_r)}{R} \qquad \left|\overline{OPF(A)}\right| = \frac{\sum_{r=1}^{R}(|OPF(A)|_r)}{R} \tag{3.21}$$

En el análisis de estas métricas, también se tienen en cuenta el porcentaje de soluciones en la envolvente resultante de las R ejecuciones, que pertenecen a la frontera de Pareto óptima.

3.4. Problema 2P-2TMD

En este problema se tiene un edificio de dos plantas para la instalación de dos dispositivos TMD. Las restricciones de los datos de entrada se recogen en la Tabla 3.3:

$m_1^{max,i}$	$m_2^{max,i}$	ω_1^{max}	ω_2^{max}	ξ_1^{max}	ξ_2^{max}	x_1^{max}	x_2^{max}
0.04	0.04	12.00	12.00	1.00	1.00	2	2
m_1^{min}	m_2^{min}	ω_1^{min}	ω_2^{min}	ξ_1^{min}	ξ_2^{min}	x_1^{min}	x_2^{min}
0.0001	0.0001	0.00	0.00	0.00	0.00	1	1

Tabla 3.3. Restricciones de los datos de entrada para 2P-2TMD

Así, un individuo en el espacio de búsqueda tiene ocho dimensiones comprendidas entre estos límites, de modo que con estas restricciones se genera la población aleatoria inicial para comenzar a trabajar con los algoritmos.

Para obtener diversos frentes en función de la masa del TMD a instalar, se ha trabajado con cinco escenarios donde los valores máximos de masa del TMD varían según la Tabla 3.4.

Escenario 1	*Escenario* 2	*Escenario* 3	*Escenario* 4	*Escenario* 5
$m^{max,1}$	$m^{max,2}$	$m^{max,3}$	$m^{max,4}$	$m^{max,5}$
0.04	0.06	0.08	0.10	0.12

Tabla 3.4. Escenarios posibles en TMD

Los parámetros generales de los algoritmos como el tamaño de la población N, número de ejecuciones y evaluaciones se ha escogido teniendo en cuenta los parámetros propuestos por los autores de las diversas metaheurísticas. Así, los valores seleccionados son:

N	200
Evaluaciones	300.000
Ejecuciones	20

Los parámetros de control de cada algoritmo son los establecidos en la Sección 3.2., donde NSGA-II y SPEA2 comparten estrategia de cruce y mutación mientras que MOEA/D utiliza el operador de evolución diferencial.

3.4.1. Resultados

A continuación, se muestran los resultados gráficos de las fronteras EPF obtenidas para cada algoritmo.

3.4.1.1. NSGA-II

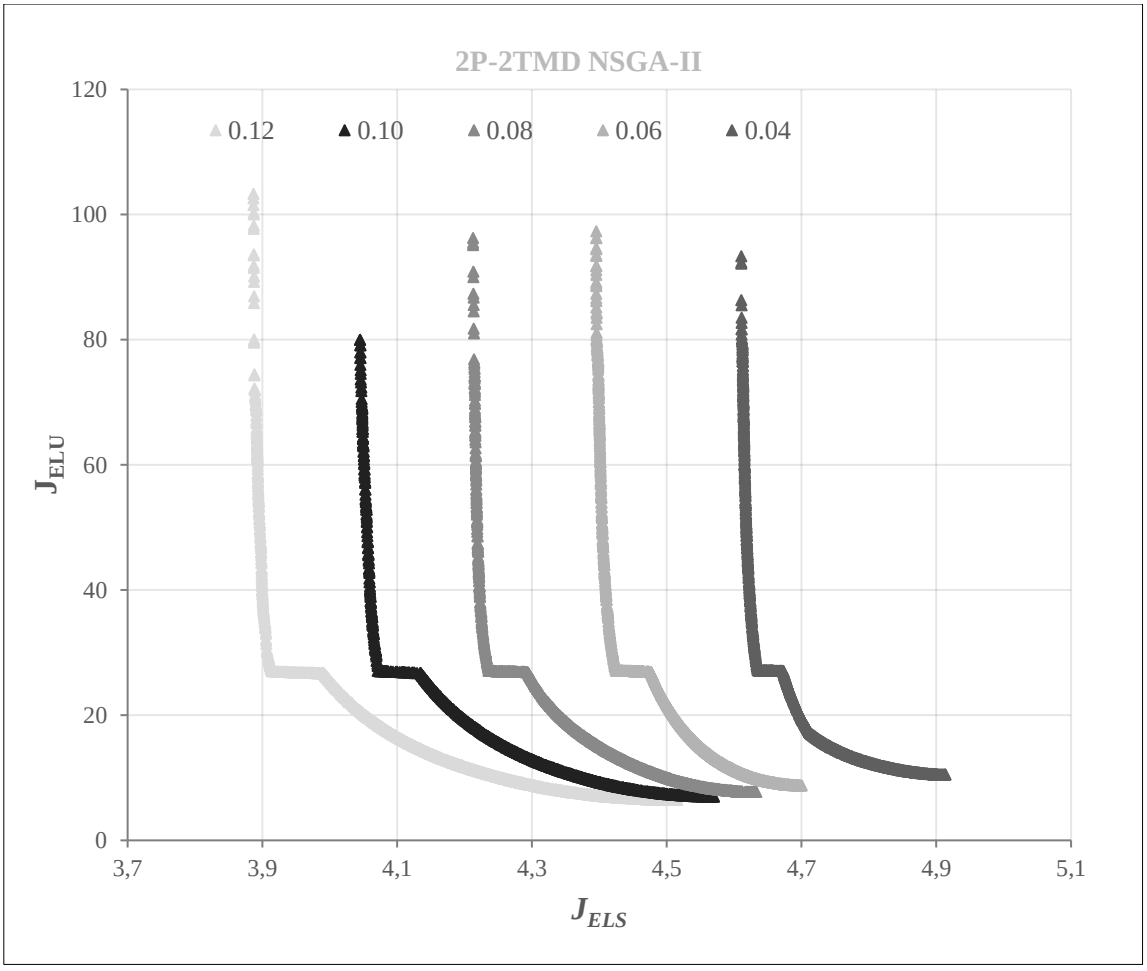

Figura 3.4. Fronteras Pareto para NSGA-II en 2P-2TMD de 0.12, 0.10, 0.08, 0.06, 0.04

En la Figura 3.4. se muestran los resultados obtenidos por NSGA-II en las veinte ejecuciones para cada una de las masas máximas, desde 0.04 hasta 0.12, definiendo las Fronteras de Pareto Envolventes (EPF).

Podemos observar que estamos ante una tipología de frente no convexo, definido en la Sección 1.3.5., donde acontece la unión de dos fronteras Pareto diferenciadas, originadas por la restricción de colocación de las masas (variable discreta), dando lugar a esta tipología de cornisa entre frentes, que se corresponde con la unión entre dos casos de colocación del dispositivo:

- $(1,1)$: Caso donde colocamos dos dispositivos TMD en la primera planta (cresta inferior derecha)
- $(2,2)$: Caso donde colocamos los dos dispositivos en la segunda planta (cresta superior izquierda).
- $(1,2) = (2,1)$: Caso donde colocamos distintos dispositivos en diferentes plantas (unión de crestas).

Analizando gráficamente los resultados de NSGA-II se aprecia la aptitud del algoritmo para enfrentarse a este tipo de problema de decisión, obteniendo una buena densidad de soluciones no dominadas, lo que resulta en una mejor definición de la Frontera de Pareto.

Por otro lado, también se aprecia la dificultad del algoritmo para encontrar soluciones que minimizan el primer objetivo, J_{ELS}, conforme nos acercamos al caso $(2,2)$, mostrando una menor densidad de soluciones no dominadas más allá de $J_{ELU} = 80$, especialmente para cuando la masa máxima es de 0.10.

En cambio, también se observa la capacidad para obtener un mayor número de soluciones para el caso $(1,1)$, cresta inferior derecha, especialmente conforme se aumenta la masa máxima.

3.4.1.2. SPEA2

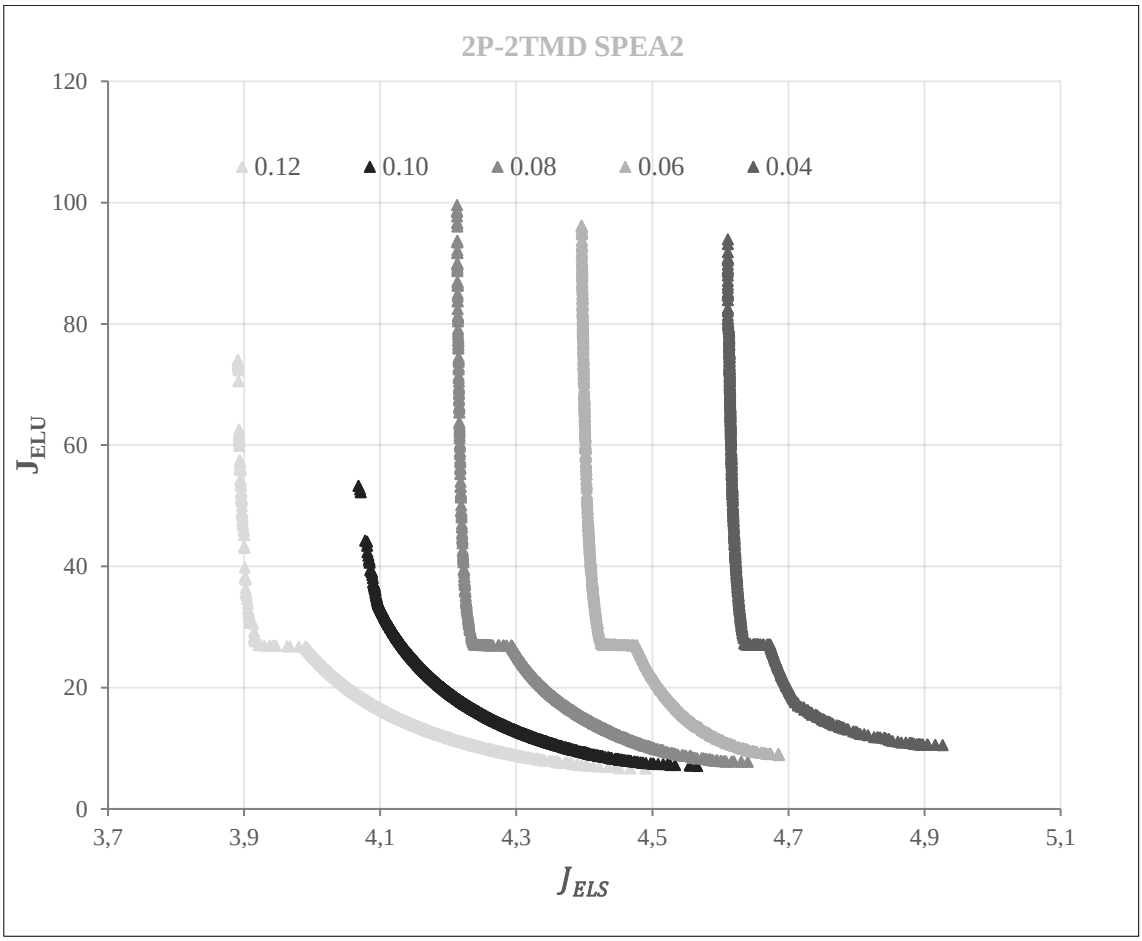

Figura 3.5. Fronteras Pareto para SPEA2 en 2P-2TMD de 0.12, 0.10, 0.08, 0.06, 0.04

Atendiendo a los resultados obtenidos por SPEA2 (Figura 3.5) y comparando gráficamente con los obtenidos por NSGA-II (Figura 3.4), podemos observar que para el caso donde la masa máxima es 0.10, no es capaz de encontrar soluciones no dominadas pertenecientes al frente óptimo, correspondiente al caso $(2,2)$.

Algo similar ocurre para el caso de masa máxima 0.12 que, aunque tiene una actuación mejor que la anterior, no es capaz de encontrar suficientes valores del frente que minimicen al primer objetivo, quedando incluso por debajo de $J_{ELU} = 80$.

En cambio, para el resto de los casos, el algoritmo parece obtener una buena densidad de soluciones, con una dispersión y distancia entre soluciones que, a simple vista, parece suficiente.

3.4.1.3. MOEA/D-DE: Versión 1

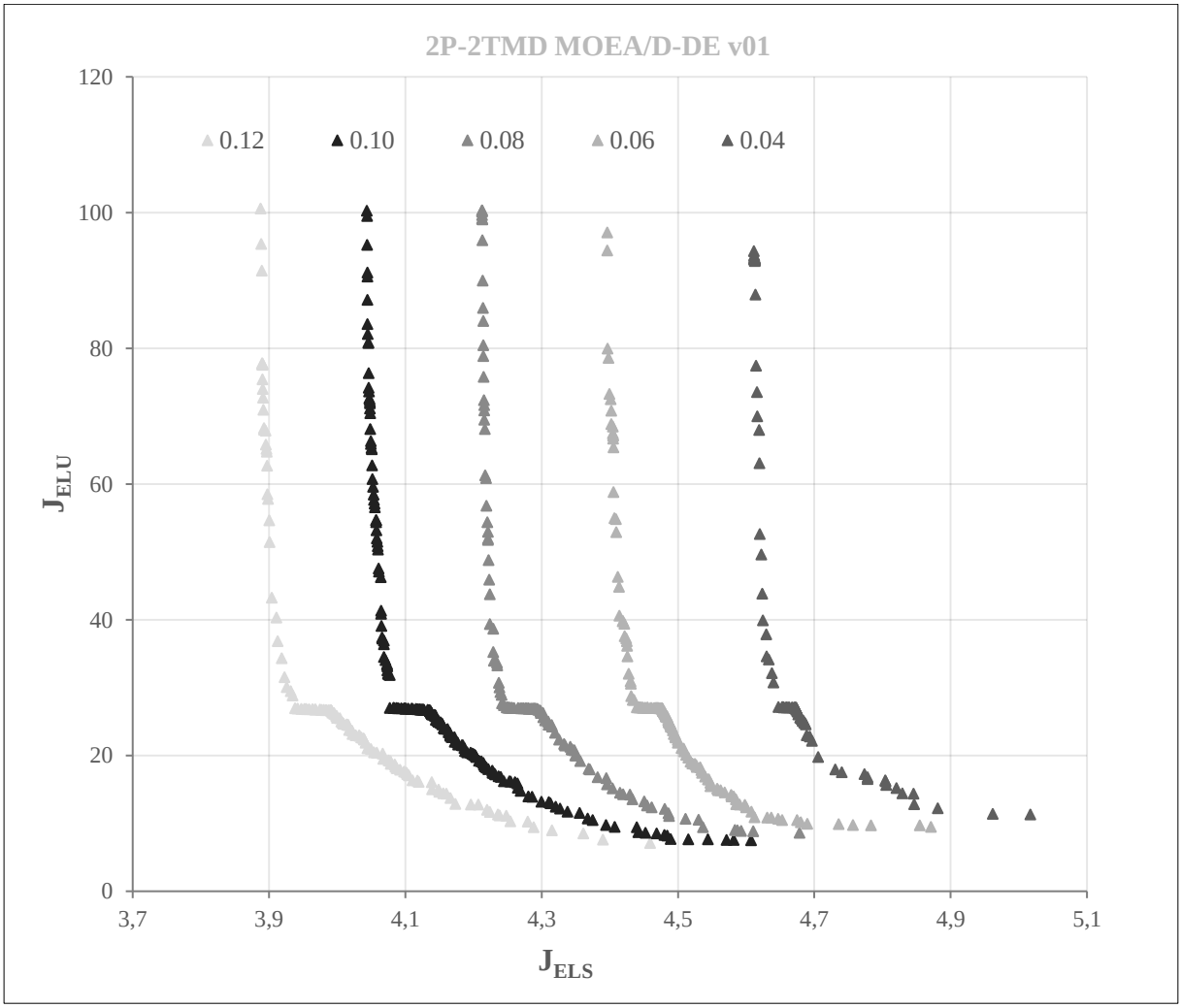

Figura 3.6. Fronteras Pareto para MOEA/D-DE v01 en 2P-2TMD de 0.12, 0.10, 0.08, 0.06, 0.04

A primera vista, resulta evidente que la densidad de soluciones no dominadas con respecto a los otros dos algoritmos es inferior para cualquier masa máxima utilizada.

En cambio, los primeros resultados de MOEA/D-DE consiguen saltar del primer frente al segundo, incluso obteniendo soluciones en los extremos más alejadas con respecto a sus competidores y pasando el umbral $J_{ELU} = 80$ en el escenario crítico de masa 0.10, dando la impresión de que apenas discrepa entre escenarios en cuanto a su capacidad de obtener soluciones.

El hecho de encontrar soluciones más distantes entre sí puede ser síntoma de haber seleccionado una población de vecinos muy grande para el problema, aumentando la probabilidad de saltar a soluciones más lejanas.

Esto explicaría también la diversidad de soluciones encontradas que, pese a ser distantes entre sí, se extienden a lo largo de los objetivos de una forma más eficaz.

Por este motivo se ha hecho una modificación del algoritmo, incluyendo al propio individuo como vecino y descartando por tanto al vecino más lejano de la comunidad. Los resultados se muestran en la Figura 3.7.

3.4.1.4. MOEA/D-DE: Versión 2

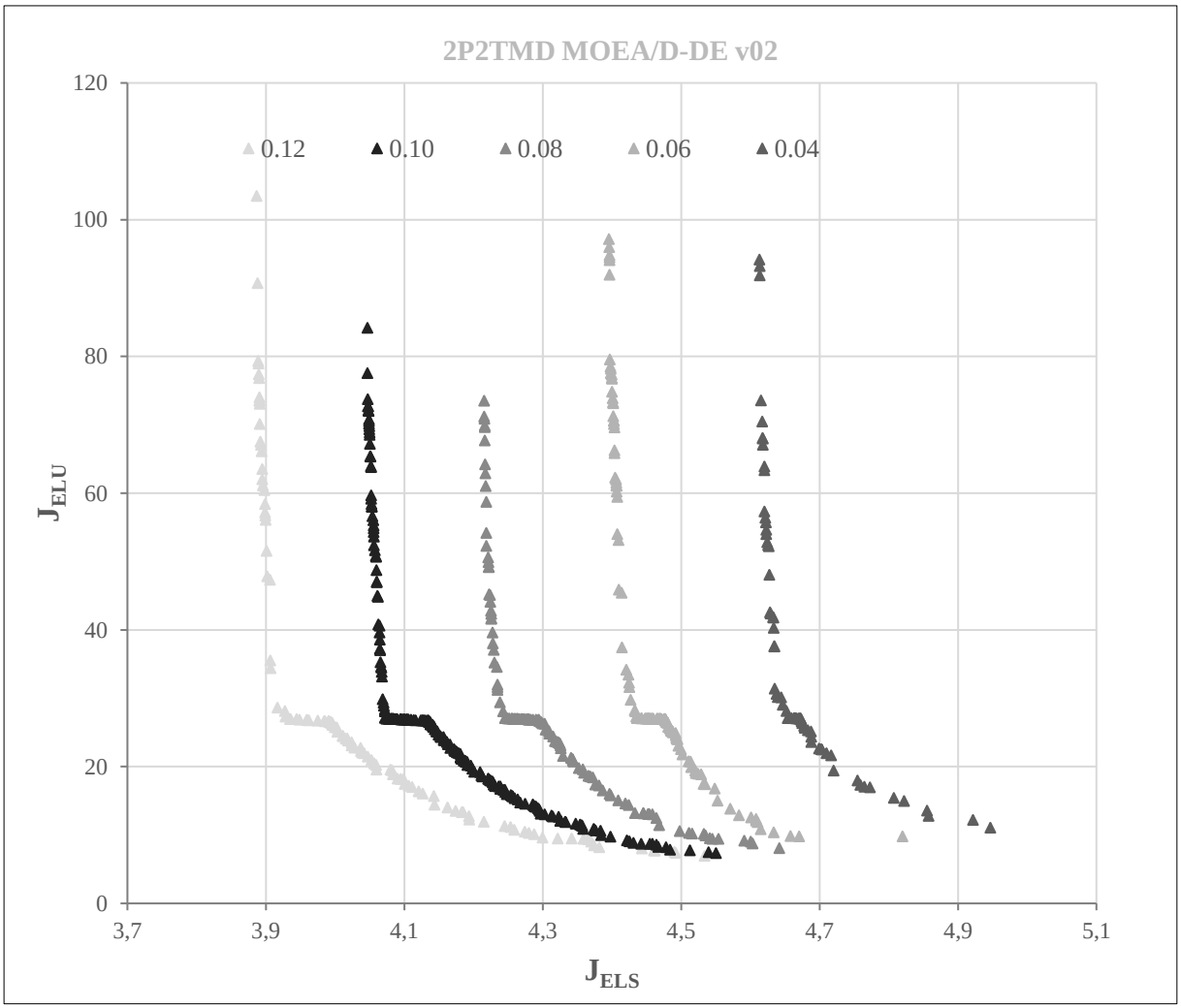

Figura 3.7. Fronteras Pareto para MOEA/D-DE v02 en 2P-2TMD de 0.12, 0.10, 0.08, 0.06, 0.04

Los resultados obtenidos con esta versión parecen tratar de resolver el problema de la distancia entre soluciones, aunque no de forma tan efectiva, y a costa de perder extensión en el dominio de soluciones para los cinco escenarios.

En la siguiente sección se va a realizar un análisis de las métricas obtenidas de estas dos versiones para ver cómo de efectiva es la estrategia de incluir o no al propio individuo dentro de su comunidad.

3.4.1.5. Comparación Gráfica: MOEA/D vs. SPEA2 vs. NSGA-II

A continuación, se muestran en la Figura 3.8 las envolventes superpuestas para cada escenario de masa:

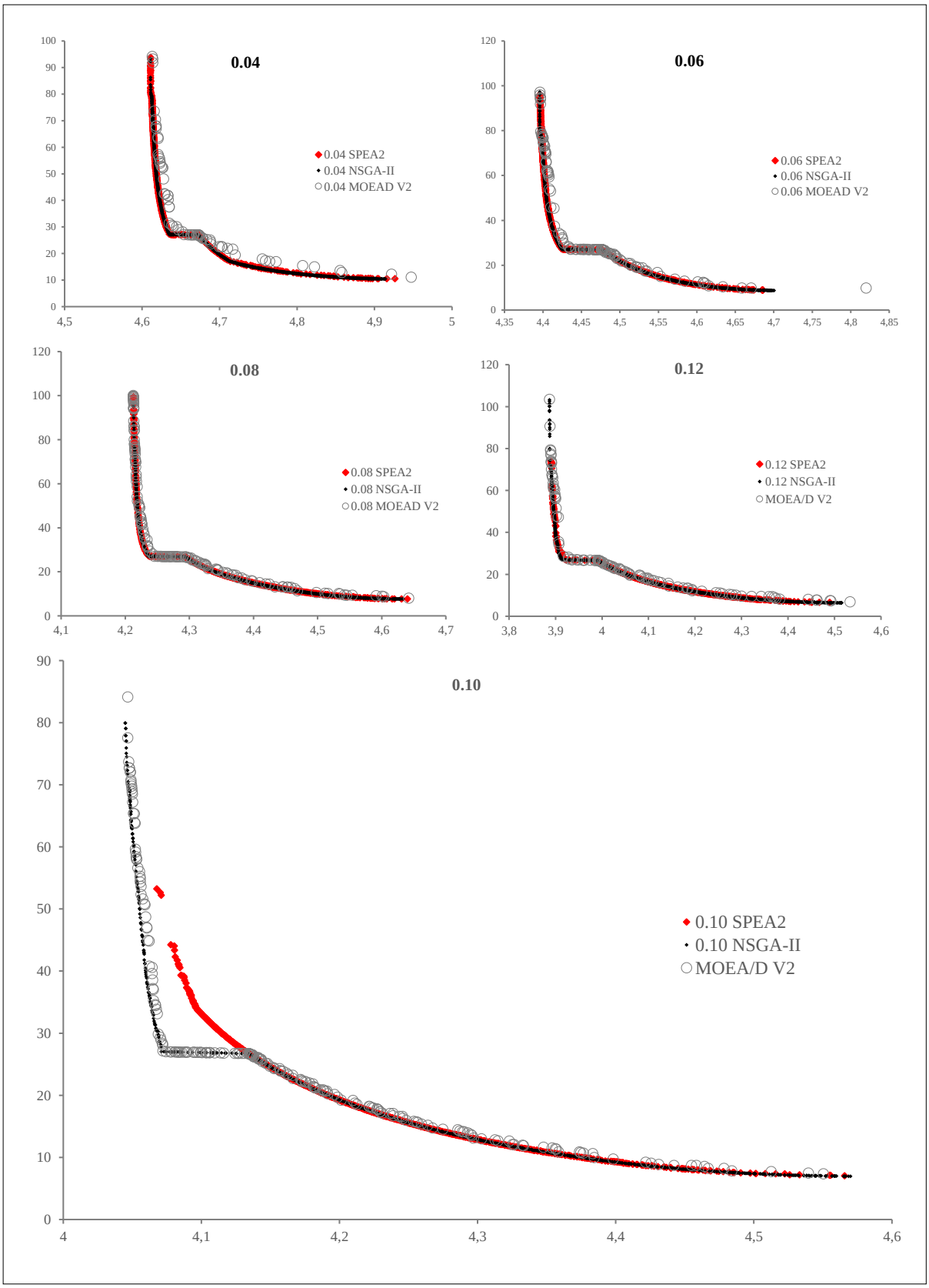

Figura 3.8. Comparación de envolventes en cada escenario de masas

Aquí se observa bien claro cómo el algoritmo SPEA2 tiene dificultades para encontrar soluciones no dominadas pertenecientes a la frontera óptima cuando la masa máxima es 0.10. Concretamente en la

zona correspondiente al caso $(2,2)$ de colocación de ambos dispositivos en el segundo piso (cresta superior izquierda), mientras que los otros dos algoritmos sí consiguen obtener soluciones pertenecientes al frente óptimo.

3.4.1.6. Frontera Óptima de Pareto

Al comienzo del capítulo se hizo una introducción sobre la singularidad que presenta el frente de Pareto en este problema (Sección 3.1.1), y es que se trata de un frente compuesto por dos frentes diferenciados: el que resulta de la colocación de los dos dispositivos en la primera planta $(1, 1)$ y en la segunda $(2, 2)$, respectivamente. Es por ello por lo que los resultados gráficos de algunos algoritmos parecen tener dificultades en obtener soluciones en alguno de los frentes, concretamente, el correspondiente al caso $(2, 2)$, correspondiente a la cresta superior izquierda.

Con objeto de intentar encontrar una respuesta a este hecho, se ha elaborado la Tabla 3.5, que recoge el porcentaje de ejecuciones del algoritmo en donde es capaz de encontrar soluciones de la frontera óptima (OPF), es decir, la capacidad de encontrar, simultáneamente, soluciones intermedias $(1,2) = (2,1)$ y soluciones de la cresta superior $(2,2)$:

$2P\text{-}2TMD$	**0.04**	**0.06**	**0.08**	**0.10**	**0.12**
$MOEA/D_{v2}$	50 %	40 %	50 %	35 %	40 %
$SPEA2$	75 %	80 %	35 %	0 %	10 %
$NSGA-II$	90 %	85 %	90 %	65 %	65 %

Tabla 3.5. Porcentaje de ejecuciones donde se obtiene el frente OPF

Esta tabla muestra la dificultad de los tres algoritmos para encontrar soluciones pertenecientes a la frontera óptima en el escenario de masa 0.10, donde SPEA2 directamente no es capaz de encontrar soluciones, NSGA-II disminuye notablemente la proporción que venía acumulando y MOEA/D disminuye en una proporción menor, pero considerable.

Esta dificultad de encontrar resultados por parte de SPEA2 motivó la ejecución del algoritmo en un análisis preliminar, donde se restringieron las posiciones de los dispositivos, de forma que se ejecutó específicamente para dos masas a determinar por el algoritmo: una en la primera planta y, otra en la segunda $(1,2)$ y. separado, la colocación de los dos dispositivos en la segunda planta $(2,2)$.

De esta manera, se pudo identificar la complejidad del frente que justifica el uso de algoritmos diseñados para tratar esta tipología de frente.

En la Figura 3.9 se muestran las soluciones del análisis preliminar a este estudio, que pertenecen a la frontera de Pareto del algoritmo SPEA2 con las condiciones de diseño especificadas anteriormente. Se puede ver en detalle la formación de estas dos fronteras para cada escenario, prestando especial atención al caso 0.10, donde el algoritmo directamente no puede obtener una frontera lo suficientemente definida.

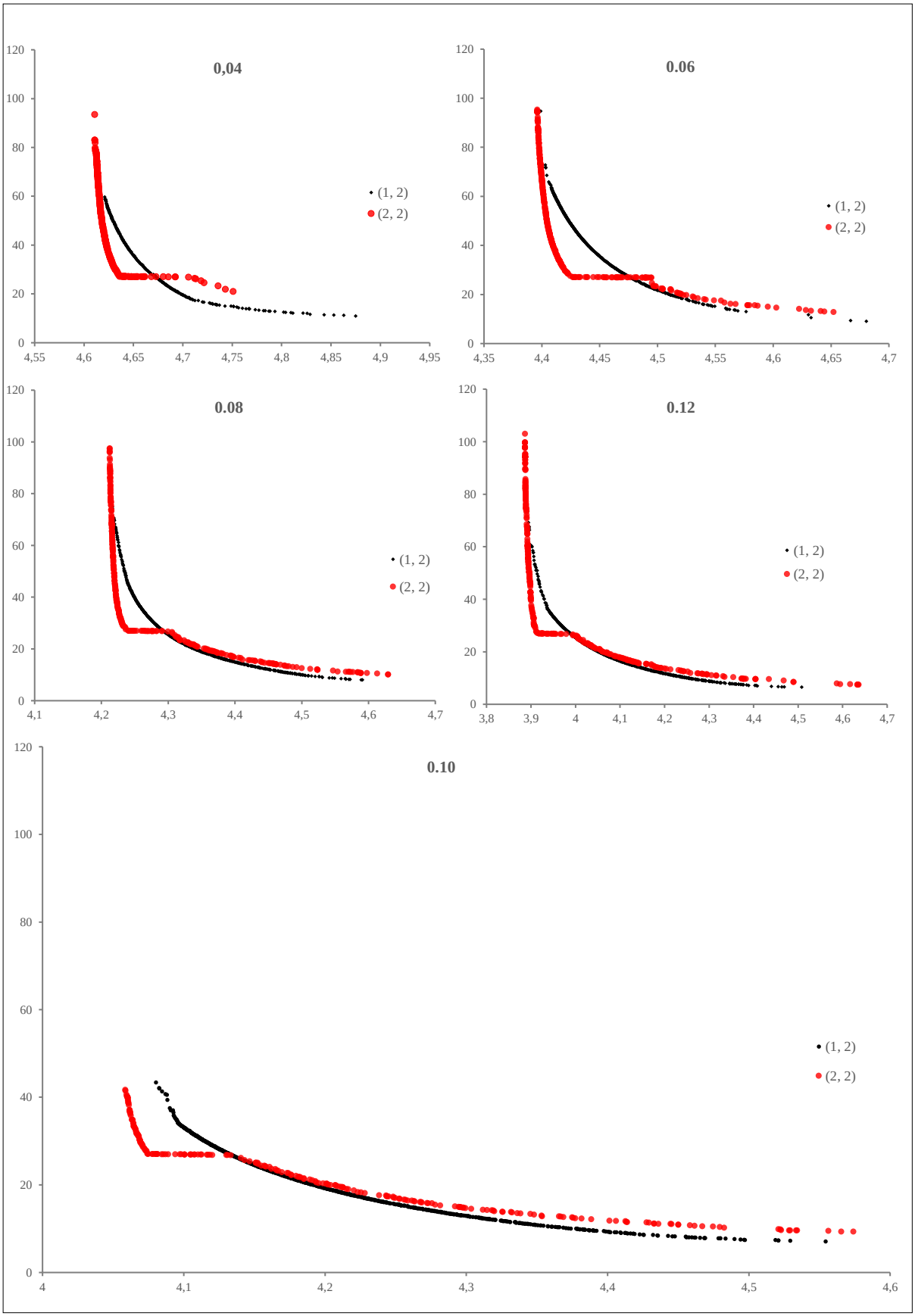

Figura 3.9. Fronteras de Pareto para SPEA2 fijando posiciones

3.4.2. Análisis de métricas

Para realizar un análisis cuantitativo de los frentes obtenidos en cada escenario por los algoritmos de trabajo se van a utilizar los indicadores de calidad definidos en la Sección 1.3.6., comprobando el desempeño de cada uno de los algoritmos en las métricas de máxima dispersión, hipervolumen generado con el frente, espaciado entre soluciones y distancia de éstas a la frontera óptima. Tras esto, se va a realizar un análisis de las métricas cardinales, en donde compararemos el número de soluciones obtenidas en la envolvente de cada algoritmo, así como su pertenencia a la frontera óptima.

3.4.2.1. Indicadores de calidad

A continuación, se recogen los resultados obtenidos en las métricas definidas anteriormente para el algoritmo MOEA/D en sus dos versiones, NSGA-II y SPEA2:

$2P\text{-}2TMD$	**0.04**	**0.06**	**0.08**	**0.10**	**0.12**
$\overline{MS}\ MOEA/D_{v1}$	0.49	1.30	1.13	0.92	0.95
$\overline{MS}\ MOEA/D_{v2}$	0.52	1.43	0.95	0.99	0.94
$\overline{MS}\ SPEA2$	0.83	0.83	0.81	0.63	0.73
$\overline{MS}\ NSGA-II$	0.92	0.94	1.01	1.00	0.94
$\overline{HPV}\ MOEA/D_{v1}$	1.04	1.02	1.06	1.23	1.06
$\overline{HPV}\ MOEA/D_{v2}$	1.01	0.99	0.87	0.85	0.91
$\overline{HPV}\ SPEA\ 2$	0.88	0.95	0.69	0.55	0.67
$\overline{HPV}\ NSGA-II$	0.92	0.97	1.03	0.96	0.85
$\overline{SP}\ MOEA/D_{v1}$	0.029	0.026	0.033	0.034	0.027
$\overline{SP}\ MOEA/D_{v2}$	0.042	0.024	0.025	0.036	0.029
$\overline{SP}\ SPEA2$	0.022	0.015	0.014	0.012	0.016
$\overline{SP}\ NSGA-II$	0.005	0.005	0.005	0.006	0.006
$\overline{GD}\ MOEA/D_{v1}$	$0.82\ 10^{-3}$	$5.13\ 10^{-3}$	$4.88\ 10^{-3}$	$3.87\ 10^{-3}$	$2.42\ 10^{-3}$
$\overline{GD}\ MOEA/D_{v2}$	$1.09\ 10^{-3}$	$6.25\ 10^{-3}$	$2.90\ 10^{-3}$	$3.95\ 10^{-3}$	$2.76\ 10^{-3}$
$\overline{GD}\ SPEA2$	$1.16\ 10^{-3}$	$0.93\ 10^{-3}$	$1.75\ 10^{-3}$	$1.78\ 10^{-3}$	$1.51\ 10^{-3}$
$\overline{GD}\ NSGA-II$	$0.58\ 10^{-3}$	$0.73\ 10^{-3}$	$0.65\ 10^{-3}$	$1.41\ 10^{-3}$	$0.64\ 10^{-3}$

Tabla 3.6. Métricas obtenidas para 2P-2TMD: Máximo Spread $\overline{MS}$, Hipervolumen $\overline{HPV}$, Espaciado Normalizado, $\overline{SP}$ y Distancia Generacional $\overline{GD}$

Máximo Spread Normalizado, $\overline{MS}$: Este indicador mide la distancia euclídea entre dos soluciones extremas en el espacio objetivo y las compara con los extremos del frente óptimo, Figura 3.9.

Alcanzar un valor próximo a la unidad indica que tenemos una amplia dispersión en las soluciones encontradas.

En el caso de MOEA/D, los resultados son buenos a partir del escenario de masa máxima 0.08 y mejoran conforme aumentamos el valor de la masa máxima, llegando a igualar e incluso superando los resultados de NSGA-II.

SPEA2 mantiene la regularidad en sus resultados hasta llegar al escenario crítico de masa 0.10, donde disminuye notablemente debido a las dificultades para encontrar soluciones no dominadas en el escenario $(2,2)$, obteniendo un frente menos completo que sus adversarios.

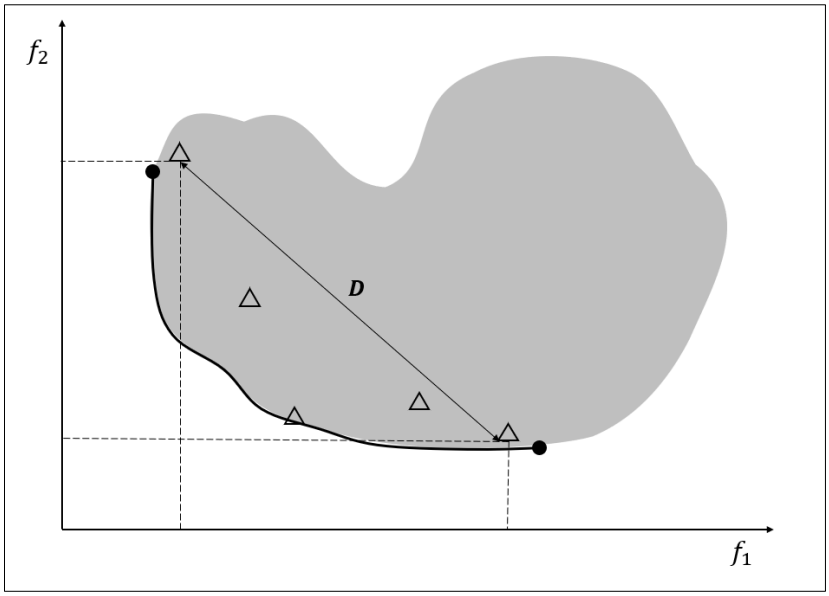

Figura 3.10. Representación del máximo spread. Fuente:(Deb, 2001a)

Medida del Hipervolumen Normalizado, $\overline{HPV}$: Esta métrica calcula el volumen en el espacio objetivo cubierto por los miembros del frente obtenido. En la Figura 3.10 se muestra como una región con relleno de trama. En el resultado obtenido en la Tabla 3.5, el indicador exhibe el ratio de este hipervolumen con el que se obtiene de la Frontera Óptima de Pareto (OPF), por lo que son deseables valores próximos a la unidad.

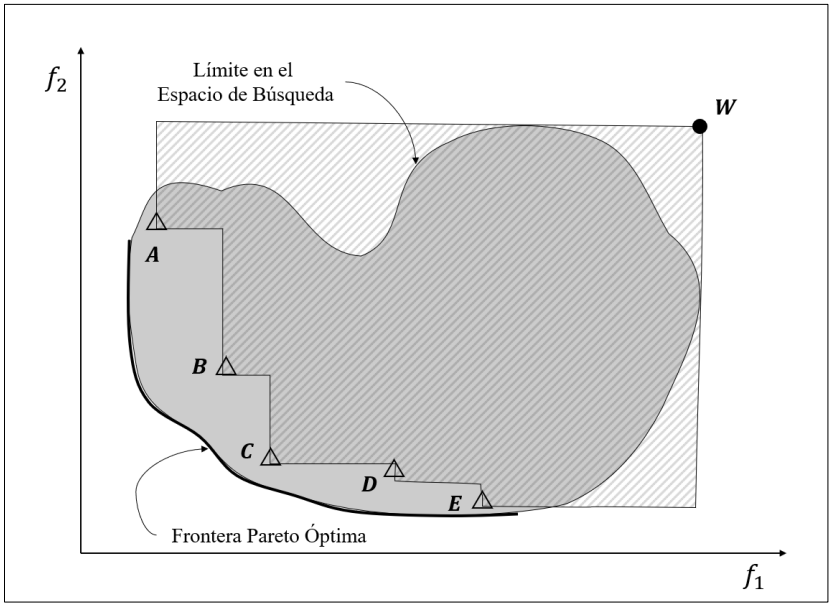

Figura 3.11. Hipervolumen encerrado por las soluciones no dominadas. Fuente:(Deb, 2001a)

En vista de los resultados MOEA/D se desempeña favorablemente en sus dos versiones, aunque en la segunda versión es capaz de mejorar sus resultados, obteniendo valores más uniformes que mejoran conforme avanzamos en escenarios, salvo en el caso crítico 0.10, donde obtiene sus peores resultados.

SPEA2 no consigue obtener buenos resultados más allá del escenario 0.06, donde se observa que las soluciones, conforme avanzan en el frente hacia las crestas, son cada vez más distantes entre sí, haciendo que el hipervolumen resultante sea menor que el de sus adversarios.

Por el contrario, NSGA-II consigue mantener su hipervolumen en los cinco escenarios, tomando valores muy próximos a la unidad en todos los casos salvo en el escenario 0.12.

Espaciado Normalizado $\overline{SP}$: Mide la distancia entre soluciones consecutivas, de manera que cuando las soluciones están cerca y espaciadas de forma uniforme, la distancia correspondiente será pequeña, por lo que un algoritmo con soluciones de bajo espaciado es mejor que otro con espaciados mayores; idealmente, el mejor espaciado es el de valor nulo.

MOEA/D es el peor parado en el resultado de esta métrica, obteniendo valores en todos los escenarios que están por encima de los de sus adversarios. Esto ya se veía venir atendiendo a las envolventes de sus dos versiones, donde se podía observar la mayor distancia entre soluciones con respecto a los algoritmos SEPA2 y NSGA-II.

Comparando ahora los resultados entre los dos algoritmos restantes, NSGA-II consigue obtener valores en esta métrica 3 veces más pequeños que los que obtiene SPEA2, manteniendo prácticamente constante la distancia entre soluciones, consiguiendo así obtener un frente poblado con una buena densidad en los cinco escenarios.

Distancia Generacional Normalizada $\overline{GD}$: Esta métrica obtiene un valor medio de la distancia de las soluciones no dominadas encontradas por el algoritmo hasta las soluciones pertenecientes al conjunto de Pareto óptimo. Así, por ejemplo, en la Figura 3.10, la solución $\boldsymbol{E}$ es la más cercana al frente de Pareto óptimo.

Por lo tanto, un algoritmo que dé valores bajos en esta métrica indica que está más próximo a la frontera óptima, por lo que es mejor.

MOEA/D obtiene valores dispares en esta métrica, con resultados generalmente peores a los de sus adversarios, salvo en su primera versión, donde es capaz de obtener un buen resultado (próximo al obtenido por NSGA-II) en el primer escenario 0.04. Esto refleja la dificultad de MOEA/D para encontrar soluciones que pertenezcan al frente óptimo, quedándose lejos con respecto a NSGA-II y SPEA2.

De nuevo, NSGA-II consigue obtener resultados que, en general, son 2 veces mejor que los que obtiene SPEA2. Aunque es precisamente a través de este indicador donde podemos observar la singularidad del escenario 0.10, reflejando la dificultad de NSGA-II para encontrar soluciones menos distantes en este caso particular.

3.4.2.2. Indicadores de Calidad: Cobertura de conjunto

$2P$-$2TMD$	0.04	0.06	0.08	0.10	0.12
$\overline{C(AB)}$	12.64 %	9.91 %	23.60 %	12.27 %	14.88 %
$\overline{C(BA)}$	84.85 %	87.21 %	56.00 %	42.60 %	45.35 %
$\overline{C(AC)}$	4.54 %	5.94 %	6.38 %	2.76 %	4.20 %
$\overline{C(CA)}$	88.07 %	87.02 %	89.87 %	77.21 %	86.33 %

Tabla 3.7. y Cobertura de Conjunto $\overline{C(AB)}$ para los algoritmos MOEA/D-DE v01 (A), SPEA2 (B), NSGA-II (C)

$2P$-$2TMD$	0.04	0.06	0.08	0.10	0.12
$\overline{C(AB)}$	9.81 %	8.79 %	20.82 %	15.16 %	12.79 %
$\overline{C(BA)}$	84.23 %	88.29 %	57.11 %	42.70 %	47.35 %
$\overline{C(AC)}$	4.58 %	4.37 %	7.02 %	4.60 %	1.56 %
$\overline{C(CA)}$	87.67 %	87.58 %	86.04 %	79.89 %	86.86 %

Tabla 3.8. Cobertura de Conjunto $\overline{C(AB)}$ para los algoritmos MOEA/D-DE v02 (A), SPEA2 (B), NSGA-II (C)

$2P$-$2TMD$	0.04	0.06	0.08	0.10	0.12
$\overline{C(BC)}$	13.51 %	11.85 %	7.89 %	11.12 %	11.90 %
$\overline{C(CB)}$	41.92 %	43.54 %	54.05 %	31.33 %	38.43 %

Tabla 3.9. Cobertura de Conjunto $\overline{C(AB)}$ para los algoritmos SPEA2 (B), NSGA-II (C)

Cobertura de conjunto normalizada $\overline{C(AB)}$: Esta métrica se utiliza para tener una idea aproximada de la dispersión relativa de las soluciones obtenidas por dos algoritmos A y B. Recordando lo explicado anteriormente, un valor de $C(AB) = 1$ indica que todas las soluciones del algoritmo B son dominadas débilmente por soluciones el algoritmo A.

De las Tablas 3.6 y 3.7 se deduce que gran parte de las soluciones en MOEA/D en sus dos versiones son dominadas por sus adversarios, con diferencias sutiles donde la segunda versión parece que se defiende ligeramente mejor.

Queda reflejada la dificultad de SPEA2 para encontrar soluciones que dominen a las de sus adversarios conforme aumentamos la masa máxima de trabajo, disminuyendo hasta dominar al 42.60 % de las soluciones, apreciándose aquí la bondad de MOEA/D con respecto a SPEA2 para encontrar soluciones en escenarios con masas elevadas. a pesar del mal resultado en este indicador.

3.4.2.3. Métricas Cardinales

$2P\text{-}2TMD$	0.04	0.06	0.08	0.10	0.12
$\lvert OPF \rvert$	1397	1138	1112	1293	1298
$\lvert EPF(A_{v1}) \rvert$	98	139	113	188	118
$\lvert EPF(A_{v2}) \rvert$	68	122	143	217	112
$\lvert EPF(B) \rvert$	707	827	775	1238	863
$\lvert EPF(C) \rvert$	984	980	893	902	980
$\%EPF(A_{v1}) \in OPF$	0.25 %	0.90 %	0.54 %	1.39 %	0.31 %
$\%EPF(A_{v2}) \in OPF$	0.15 %	0.00 %	0.45 %	1.24 %	0.31 %
$\%EPF(B) \in OPF$	26.68 %	18.61 %	34.35 %	43.63 %	41.14 %
$\%EPF(C) \in OPF$	73.06 %	81.30 %	65.11 %	54.98 %	58.55 %
$\lvert \overline{PF(A_{v1})} \rvert$	445.95	478.4	199.6	211.25	230.95
$\lvert \overline{PF(A_{v2})} \rvert$	386.50	689.95	235.15	207.80	220.20
$\lvert \overline{PF(B)} \rvert$	232.10	231.05	238.05	245.35	244.35
$\lvert \overline{PF(C)} \rvert$	222.00	229.70	222.90	223.80	224.80

Tabla 3.10. Cardinalidad de fronteras. Algoritmos: MOEA/D-DE v01 (A), SPEA2 (B), NSGA-II (C)

Cardinalidad de la envolvente $\lvert EPF(A) \rvert$: Los resultados en esta métrica no son favorables para el algoritmo MOEA/D, con tan solo un 1.24% de soluciones pertenecientes a la frontera óptima en el mejor de los casos y con un máximo de 217 soluciones en la envolvente, una medida aproximadamente siete veces inferior a la de sus adversarios.

Un punto interesante de esta tabla es la cardinalidad de la envolvente SPEA2 para el escenario crítico 0.10 e incluso el 0.12, obteniendo un valor notablemente elevado con respecto a la cardinalidad de los otros algoritmos. Parece que la carencia de encontrar superiores en la cresta superior la suple encontrando una proporción mayor de soluciones en la cresta inferior, consiguiendo incrementar el porcentaje de soluciones que pertenecen a la frontera óptima.

NSGA-II es capaz de encontrar muchas soluciones no dominadas, hasta 984, de forma robusta, manteniendo constante el valor salvo para los escenarios 0.08 y 0.10 en donde disminuye ligeramente su rendimiento. Con todo, el porcentaje de soluciones que pertenecen a la frontera óptima es siempre notablemente superior a la de sus adversarios.

Cardinalidad normalizada de los Frentes obtenidos $\lvert \overline{PF(A)} \rvert$: Aunque según el indicador anterior hayamos comprobado que un gran porcentaje de las soluciones de MOEA/D no pertenecen a la frontera óptima, este indicador demuestra la capacidad del algoritmo para encontrar soluciones no dominadas, quedando expuesta su cualidad para encontrar soluciones donde, en algunos casos, triplica a sus

adversarios. Esta cualidad va en sintonía con las premisas realizadas con los gráficos a simple vista, donde se afirmaba la capacidad de encontrar soluciones extremas más distantes del origen y en mayor número que sus adversarios. En cambio, al formar la envolvente de Pareto, queda reflejada la escasa diversidad en las soluciones obtenidas, quedando la mayoría dominadas por unas pocas, en el peor caso un 82% de las soluciones obtenidas en el total de las ejecuciones se descartan por estar dominadas por el restante 18%.

SPEA2 y NSGA-II son más regulares en la cardinalidad de sus conjuntos, obteniendo un valor medio de 230 soluciones dominadas en cada escenario.

4. CONCLUSIONES

El objeto del presente trabajo ha sido realizar un estudio de las metaheurísticas multiobjetivo, comprender su funcionamiento y probar su eficacia ante un problema de decisión multiobjetivo (MOP) real llevado a cabo en las instalaciones de la escuela de ingenierías industriales de esta Universidad. Este problema de decisión, de tipo estructural, se basa en la sintonización de amortiguadores de masa (TMD) que minimice dos objetivos relacionados con la seguridad estructural y el confort de los habitantes ante la acción sísmica.

De las posibles casuísticas recogidas en el modelo de trabajo, capaz de trabajar con la instalación de hasta cinco dispositivos TMD en cinco plantas diferentes, se ha detectado una singularidad particular en los primeros resultados obtenidos en uno de ellos. Esta singularidad, ubicada en el modelo de la instalación de dos dispositivos TMD en dos plantas diferentes ($2P\text{-}2TMD$), hace que el Frente de Pareto Óptimo (OPF) esté compuesto por dos frentes diferenciados, correspondientes a la colocación de los dos dispositivos en la primera altura y en la segunda, respectivamente, unidos por las soluciones que corresponden a la colocación de los dispositivos de diferentes masas en diferentes alturas. Así, la envolvente de las soluciones obtenidas para el problema $2P\text{-}2TMD$ tiene una tipología compleja, donde resulta difícil obtener un gran número de soluciones no dominadas, esto es, con el grado suficiente de dispersión, densidad y extensión a lo largo de la frontera óptima, que proporcionen una gama de selección suficiente para la toma de decisiones del diseñador.

Tras realizar un estudio de las metaheurísticas disponibles en el mercado actual, se ha concluido que la metaheurística MOEA/D, basada en la descomposición del MOP, es una buena candidata para competir en este problema contra dos de los algoritmos de mayor éxito en su implementación hasta el momento: NSGA-II y SPEA2. El motivo que impulsa esta decisión es su demostrada aptitud para tratar MOPs con frentes de Pareto complejos, habiendo mostrado resultados prometedores para otros MOP.

En vista a los resultados obtenidos, cabía esperar un mejor desempeño del algoritmo seleccionado. Por un lado, MOEA/D ha sido capaz de hacer frente a la tipología compleja de la frontera del problema, consiguiendo obtener una gama de soluciones no dominadas pequeña en comparación con NSGA-II y SPEA2, pero capaz de seguir el camino del frente. Por otro lado, analizando los resultados obtenidos en las métricas, la calidad de las soluciones es inferior con respecto a la de sus adversarios, especialmente NSGA-II. Así, el número y proporción de soluciones que pertenecen a la frontera óptima con respecto a la de sus adversarios es mínima: teniendo en cuenta la cardinalidad de la envolvente, apenas el 1% de las soluciones pertenecen al conjunto óptimo en el mejor de los casos, mientras que sus competidores consiguen obtener hasta un 43.64% SPEA2 y un 81.30% NSGA-II. También queda reflejada la dificultad de SPEA2 para obtener soluciones pertenecientes a la frontera óptima en la región superior, donde el algoritmo no es capaz encontrar la configuración de masas $(2{,}2)$, dando una gama de soluciones que son dominadas tanto por MOEA/D como por NSGA-II en esa zona del frente. En cambio, sí muestra un buen desempeño para poblar la región inferior. En cambio, parece suplir esta carencia encontrando un mayor número de soluciones no dominadas en la cresta inferior, de configuración de masas $(1{,}1)$, incrementando notablemente la cardinalidad de la envolvente y el porcentaje de soluciones pertenecientes al óptimo en estos escenarios.

En cuanto a la programación del algoritmo, se ha comprobado la dificultad de crear metaheurísticas o marcos normativos de nivel superior que escapen de la dependencia del problema, así como la complejidad de realizar las modificaciones adecuadas para conseguir una mejora en los resultados.

En general, se ha comprobado la aptitud de estas técnicas para tratar MOP reales, así como la capacidad de MOEA/D para tratar con Fronteras de Pareto complejas, destacando su habilidad para obtener soluciones extremas de mayor amplitud que la de sus competidores, y con una dispersión suficiente como para que el diseñador conozca una gama de soluciones lo bastante amplia como para inclinarse por una solución concreta. La principal diferencia con sus competidores es la calidad de las soluciones obtenidas, especialmente frente a NSGA-II, donde este último es capaz de obtener una mayor proporción de soluciones pertenecientes a la frontera óptima, con una mayor dispersión, más cercanas a la Frontera Óptima y con una distancia mínima entre ellas.

5. LÍNEAS FUTURAS DE DESARROLLO

Se ha visto en este estudio la dificultad de encontrar una gama de óptimos de Pareto que haga frente a la tipología de la frontera abordada en este problema MOP. A continuación, se recogen una serie de propuestas para continuar con el estudio de algoritmos genéticos multiobjetivo y selección de parámetros, de forma que consigan hacer frente a esta tipología de problema:

- Profundizar en la parametrización de los operadores de cruce y mutación, de forma que se consiga abarcar todo el frente de dominancia con un buen equilibrio entre la exploración de nuevas soluciones y la explotación de la información obtenida en cada iteración, particularmente para resolver la dificultad para conseguir soluciones no dominadas en el escenario particular mencionado en este trabajo.
- Estudiar la selección de parámetros de control propios de MOEA/D y modificarlos, como se ha tratado en el presente trabajo sin conseguir mejorar notablemente los resultados, aunque viendo los resultados prometedores obtenidos en problemas de prueba realizados por el autor de la metaheurística, es probable que exista una combinación apropiada para generar una Frontera con una buena definición y densidad de soluciones.
- Incluir a las variables discretas en el proceso de mutación. El tratamiento de estas variables sigue actualmente otro itinerario en el algoritmo para prevenir la obtención de valores no factibles. Es posible incluirlos en los operadores de forma que, tras salir de la operación, se lleve el gen a un proceso de reparación, donde se restaura su valor a uno dentro de los límites aplicando una heurística propia del problema.
- Utilizar un nuevo algoritmo de la selección propuesta en la Sección 1.4.3., que haya mostrado su robustez en la solución de MOP y que pueda ser un buen candidato para competir con los algoritmos actualmente escogidos para abordar el problema, teniendo en cuenta para ello la necesidad de hacer frente a tipologías de frente complejas.

REFERENCIAS

Abdel-Basset, M., Abdel-Fatah, L., & Sangaiah, A. K. (2018). Metaheuristic Algorithms: A Comprehensive Review. En *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications* (pp. 185-231). Elsevier. https://doi.org/10.1016/B978-0-12-813314-9.00010-4

Biswas, P. P., & Suganthan, P. N. (2018). Multiobjective Evolutionary Optimization. En *Wiley Encyclopedia of Electrical and Electronics Engineering* (pp. 1-15). American Cancer Society. https://doi.org/10.1002/047134608X.W8380

Coello, C. A. C., Lamont, G. B., & Veldhuizen, D. A. V. (Eds.). (2007). MOP Evolutionary Algorithm Approaches. En *Evolutionary Algorithms for Solving Multi-Objective Problems: Second Edition* (pp. 61-130). Springer US. https://doi.org/10.1007/978-0-387-36797-2_2

Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000). A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. *Parallel Problem Solving from Nature PPSN VI*, 849-858. https://doi.org/10.1007/3-540-45356-3_83

Deb, K., & Jain, H. (2014). An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation*, *18*(4), 577-601. https://doi.org/10.1109/TEVC.2013.2281535

Deb, K., Sindhya, K., & Okabe, T. (2007). Self-adaptive simulated binary crossover for real-parameter optimization. *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, 1187–1194. https://doi.org/10.1145/1276958.1277190

Deb, K. (2001a). Elitist Multi-Objective Evolutionary Algorithms. En *Multi-Objective Optimization using Evolutionary Algorithms* (1st ed., pp. 227-274). John Wiley & Sons, Ltd.

Deb, K.. (2001b). Multi-Objective Optimization. En *Multi-Objective Optimization using Evolutionary Algorithms* (1st ed., pp. 13-48). John Wiley & Sons, Ltd.

Deb, K.. (2001c). Salient Issues of Multi-objective Evolutionary Algorithms. En *Multi-Objective Optimization using Evolutionary Algorithms* (1st ed., pp. 301-427). John Wiley & Sons, Ltd.

Fonseca, C. M., & Fleming, P. J. (1993). *Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization*. 416-423.

Fonseca, C. M. (1995). *Multiobjective genetic algorithms with application to control engineering problems*. [Phd, University of Sheffield]. http://etheses.whiterose.ac.uk/1887/

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, *13*(5), 533-549. https://doi.org/10.1016/0305-0548(86)90048-1

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning* (1st Edition). Addison-Wesley Longman Publishing Co., Inc.

Guerra, A. (2017). *Estudio de distintos algoritmos genéticos multimodales para la sintonización óptima de TMDs múltiples en estructuras esbeltas*. http://uvadoc.uva.es/handle/10324/24294

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). *A Niched Pareto Genetic Algorithm for Multi-Objective Optimization*. *1*, 82-87. https://doi.org/10.1109/ICEC.1994.350037

Jacob, C. (2001). 2—Evolutionary Algorithms for Optimization. En C. Jacob (Ed.), *Illustrating Evolutionary Computation with Mathematica* (pp. 57-77). Academic Press. https://doi.org/10.1016/B978-155860637-1/50015-1

Jahn, J. (2007). *Introduction to the Theory of Nonlinear Optimization* (3th Edition). Springer-Verlag. https://doi.org/10.1007/978-3-540-49379-2

Jones, D. F., Mirrazavi, S. K., & Tamiz, M. (2002). Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research*, *137*(1), 1-9. https://doi.org/10.1016/S0377-2217(01)00123-0

Khosrowpour, M. (2008). *Encyclopedia of Information Science and Technology* (2nd Edition, Vol. 1-8). https://doi.org/10.4018/978-1-60566-026-4

Kim, I. Y., & de Weck, O. L. (2005). Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and Multidisciplinary Optimization*, *29*(2), 149-158. https://doi.org/10.1007/s00158-004-0465-1

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671-680. https://doi.org/10.1126/science.220.4598.671

Knowles, J., & Corne, D. (1999). The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, *1*, 98-105 Vol. 1. https://doi.org/10.1109/CEC.1999.781913

Li, H., Zhang, Q. (2009). Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, *13*(2), 284-302. https://doi.org/10.1109/TEVC.2008.925798

Li, X., & Zhang, H. (2020). A multi-agent complex network algorithm for multi-objective optimization. *Applied Intelligence*, *50*(9), 2690-2717. https://doi.org/10.1007/s10489-020-01666-8

Magdaleno, Á. (2017). *Estudio de nuevos indicadores en el dominio de la frecuencia y del tiempo para la sintonización óptima de TMDs múltiples en estructuras esbeltas*. http://uvadoc.uva.es/handle/10324/23309

Monticelli, A. J., Romero, R., & Asada, E. N. (2007). Fundamentals of Tabu Search. En *Modern Heuristic Optimization Techniques* (pp. 101-122). John Wiley & Sons, Ltd. https://doi.org/10.1002/9780470225868.ch6

Mühlenbein, H., & Schlierkamp-Voosen, D. (1993). Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization. *Evolutionary Computation*, *1*(1), 25-49.

Pétrowski, A., & Ben-Hamida, S. (2017). Multi-objective Optimization. En *Evolutionary Algorithms* (Vol. 9, pp. 165-182). John Wiley & Sons, Ltd. https://doi.org/10.1002/9781119136378.ch5

Pham, D., & Karaboga, D. (2012). *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. Springer Science & Business Media.

Salcedo-Sanz, S., Camacho-Gómez, C., Magdaleno, A., Pereira, E., & Lorenzana, A. (2017). Structures vibration control via Tuned Mass Dampers using a co-evolution Coral Reefs Optimization algorithm. *Journal of Sound and Vibration*, *393*, 62-75. https://doi.org/10.1016/j.jsv.2017.01.019

Schaffer, J. (1985). *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. 93-100.

Schott, J. R. (Jason R. (1995). *Fault tolerant design using single and multicriteria genetic algorithm optimization* [Thesis, Massachusetts Institute of Technology]. https://dspace.mit.edu/handle/1721.1/11582

Sörensen, K., & Glover, F. (2013). *Metaheuristics* (pp. 960-970). https://doi.org/10.1007/978-1-4419-1153-7_1167

Srinivas, N., & Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, *2*, 221–248. https://doi.org/10.1162/evco.1994.2.3.221

Tian, Y., Cheng, R., Zhang, X., & Jin, Y. (2017). PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization [Educational Forum]. *IEEE Computational Intelligence Magazine*, *12*(4), 73-87. https://doi.org/10.1109/MCI.2017.2742868

van Veldhuizen, D. A., & Lamont, G. B. (1999). Multiobjective evolutionary algorithm test suites. *Proceedings of the 1999 ACM symposium on Applied computing*, 351–357. https://doi.org/10.1145/298151.298382

Zhang, Q., & Li, H. (2008). MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *Evolutionary Computation, IEEE Transactions on*, *11*, 712-731. https://doi.org/10.1109/TEVC.2007.892759

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, *3*(4), 257-271. https://doi.org/10.1109/4235.797969

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, *8*(2), 173-195. https://doi.org/10.1162/106365600568202

Zitzler, E., Laumanns, M., & Thiele, L. (2001). *SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization*. *3242*.

Zitzler, E., & Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: The strength Pareto approach. En *TIK-Report* (Vol. 43) [Working Paper]. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology Zürich (ETH). https://doi.org/10.3929/ethz-a-004288833