Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

Máster en Ingeniería Industrial

MASTER EN INGENIERÍA INDUSTRIAL

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

TRABAJO FIN DE MÁSTER

Optimización de Parámetros del Algoritmo Genético Multiobjetivo SPEA2 en la Sintonización de TMDs

Autor: D. Rodrigo Escudero Vega

Tutor: D. Elena Pérez Vázquez

Valladolid, septiembre, 2018

Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

Máster en Ingeniería Industrial

MASTER EN INGENIERÍA INDUSTRIAL

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

TRABAJO FIN DE MÁSTER

Optimización de Parámetros del Algoritmo Genético Multiobjetivo SPEA2 en la Sintonización de TMDs

Autor: D. Rodrigo Escudero Vega

Tutor: D. Elena Pérez Vázquez

Valladolid, septiembre, 2018

RESUMEN

El Trabajo de Fin de Máster tiene por objetivo la optimización de los parámetros de un Algoritmo Genético Multiobjetivo (AGMO) en su aplicación a la sintonización de TMD's. De estudios previos, se sabe que estos algoritmos son de gran utilidad para la resolución de este tipo de problemas. La optimización se realizará mediante el uso de métricas de comparación que permiten discernir qué parámetros obtienen mejores resultados a través del estudio de las fronteras de Pareto que obtiene el algoritmo.

ABSTRACT

This Master Thesis is focused on the optimization of a Multiobjective Genetic Algorithm (MOGA)'s parameters in its application to the tuning of TMD (Tuned Mass Damper). From previos studies, it is known that these algorithms are very useful in solving this kind of problems. The optimization will be performed by comparison metrics which allow the user to discern which parameters get better results through de study of Pareto fronts obtained by the algorithm.

AGRADECIMIENTOS

En primer lugar, a mi familia: a mi hermano, Alejandro, a mi madre, Ruth y a mi padre, Pedro. Son ellos los que me apoyan cada día en todo lo que pueden, los que me escuchan, me aconsejan y me ayudan siempre que lo necesito. Gracias a ellos he llegado a donde estoy.

A todos los profesores que me han acompañado a lo largo del Máster y me han ayudado a lograr mis objetivos. A mi tutora, Elena Pérez. También a Álvaro Magdaleno y Antolín Lorenzana, que han colaborado para la elaboración del TFM.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Los Algoritmos Genéticos (AG) son una herramienta de optimización muy potente para la búsqueda de soluciones a problemas complejos que de otra manera sería muy difícil de resolver. Se basan en comportamientos de la naturaleza para seleccionar la mejor o, al menos, la solución más adecuada para el caso al que se enfrenta en función de las restricciones establecidas.

La principal clasificación de los Algoritmos Genéticos es:

- Algoritmos Genéticos Simples (AGS)
- Algoritmos Genéticos Multimodales (AGMM)

Además, cada uno de los grupos anteriores puede subdividirse en:

- Algoritmos Genéticos de un Único Objetivo (AGUO)
- Algoritmos Genéticos Multi-Objetivo (AGMO)

Todos ellos son válidos para la resolución de problemas, pero cada uno tiene sus limitaciones.

Los AGS son útiles para la resolución de problemas unimodales, pero tienen ciertas desventajas, puesto que en muchos casos las soluciones que encuentran son óptimos locales del problema planteado, es decir, puede darse el caso de que no encontremos la solución óptima.

Para evitar esto, se desarrollaron los AGMM que, mediante diversas técnicas, son capaces de encontrar la solución óptima (o conjunto de soluciones óptimas).

En cuanto a los AGUO, se trata de algoritmos genéticos que permiten encontrar soluciones a un problema en el que se establece un solo criterio de búsqueda, bien porque es el criterio "dominante" del problema o bien porque es el único criterio de interés en el caso de estudio. Con estos AG ocurre algo parecido a lo que ocurre con los AGS; se trata de algoritmos que no son realmente útiles cuando nos enfrentamos a un problema real de ingeniería, puesto que habitualmente se deben considerar múltiples objetivos.

Por esta razón, se fueron desarrollando los AGMO, que permiten enfrentarse a problemas en los que se optimiza una función (que modela el sistema a resolver) teniendo en cuenta varios objetivos simultáneamente. No obstante, estos algoritmos no encuentran la mejor solución (puesto que no existe), sino que trata de buscar una solución o conjunto de soluciones que se ajusten de la mejor manera a las necesidades establecidas. Es decir, una solución que satisfaga el problema teniendo en cuenta los diferentes objetivos marcados de la mejor manera posible.

Este TFM se va a centrar la optimización de los parámetros de un AGMO para enfrentarse a un problema de sintonización de TMDs. Un TMD (Tuned Mass Dumper) es un dispositivo empleado en estructuras que sirve para reducir las vibraciones en la misma. Las necesidades normativas de construcción (Estado Límite Último – ELU y Estado Límite de Servicio – ELS) que deben cumplir las estructuras obligan a que sea necesario instalar sistemas de este tipo para garantizar su cumplimiento.

La instalación de los TMDs no es sencilla, puesto que tratamos con estructuras, en muchos casos, de varios pisos, por lo que es necesario tener en cuenta muchas variables para su colocación: coste del/los TMD/s, lugar de instalación, cantidad…

Por esta razón, se va a utilizar un AGMO para la resolución del problema a partir de una función denominada “de coste” que permitirá obtener la solución más adecuada para la instalación de los TMDs, a partir de unas restricciones que simplificarán en cierta medida el problema. Posteriormente, se modificarán los parámetros de entrada al AGMO y se compararán los resultados para determinar cuáles son más adecuados.

1.1. Motivación del trabajo

La sintonización de TMDs en una estructura teniendo en cuenta múltiples parámetros se trata de un problema real de ingeniería y de gran aplicación. Mediante este TFM vamos a estudiar los resultados obtenidos por los Algoritmos Genéticos Multiobjetivo modificando sus parámetros para determinar con qué valores de los mismos los resultados son mejores.

1.2. Software utilizado

La implementación del AGMO utilizado en la resolución del problema ha sido la versión 2016a de Matlab y los métodos de comparación de las fronteras de Pareto han sido programadas también en esta versión de Matlab.

2. AMORTIGUADOR DE MASA O TMD (TUNED MASS DUMPER)

Un amortiguador de masa o TMD (Tuned Mass Damper) es un dispositivo capaz de reducir la vibración de un sistema de manera que se ajuste correctamente a su cometido.

El primer equipo de este tipo fue ingeniado por Hermann Frahm (Frahm, 1911) con el objeto de reducir la oscilación lateral de los barcos. Sin embargo, pronto hubo otros investigadores que profundizaron en el estudio de estos dispositivos y fueron introduciéndolos progresivamente a otras aplicaciones de ingeniería.

En lo que se refiere a los TMDs para su uso en estructuras, fue en 1928 cuando (Ormondroyd y Den Hartog, 1928) estudiaron los parámetros y su influencia en la instalación de estos sistemas en estructuras, recogidas en la publicación *Mechanical Vibrations* (Den Hartog, 1934).

El objetivo de este capítulo consiste en proporcionar una idea general sobre lo que es un TMD y su forma de modelarlo matemáticamente. Sin embargo, no se profundizará mucho en ello puesto que no es el objetivo principal del TFM, aunque influye en él, ya que la utilización de los Algoritmos Genéticos tiene el fin de sintonizar TMDs en una estructura.

2.1. Modelo de un TMD

Un TMD se puede aproximar por un sistema mecánico de un grado de libertad constituido por una masa móvil (m_j) solidaria a una estructura (S) por medio de un disipador de energía (c_j) y un elemento elástico (k_j). En la siguiente figura se puede observar el modelo indicado:

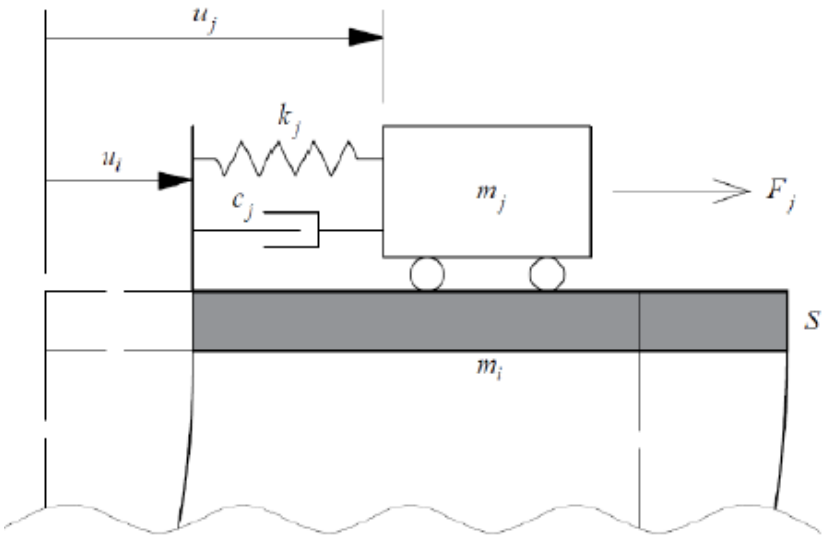

Figura 2.1 Modelo de un TMD (Magdaleno, 2017)

Idealmente, el TMD tiene su masa concentrada en un punto y se desplaza horizontalmente sobre la estructura. Cuando una fuerza actúa sobre la estructura tanto la masa como la estructura sufren un desplazamiento (u_j y u_i, respectivamente).

Mediante la Segunda Ley de Newton se puede obtener la ecuación que gobierna el movimiento del TMD:

$$m_j\ddot{u}_j + c_j(\dot{u}_j - \dot{u}_i) + k_j(u_j - u_i) = F_j \quad (2.1)$$

Siendo F_j la fuerza aplicada.

Siguiendo el mismo procedimiento se puede obtener la reacción del TMD:

$$c_j(\dot{u}_j - \dot{u}_i) + k_j(u_j - u_i) = R_i \quad (2.2)$$

Para el tratamiento de TMD es más habitual hablar de frecuencia propia (ω_j) en lugar de rigidez (k_j) y de factor de amortiguamiento crítico (ξ_j) en lugar de constante de amortiguamiento (c_j).

La frecuencia propia del sistema es aquella para la cual la respuesta del sistema presenta amplitud máxima y el factor de amortiguamiento crítico es aquel que, en el mínimo tiempo posible, disipa la respuesta del sistema.

La relación entre los cuatro parámetros citados se detalla en las siguientes ecuaciones:

$$\omega_j{}^2 = \frac{k_j}{m_j} \quad (2.3)$$

$$2\xi_j\omega_j = \frac{c_j}{m_j} \quad (2.4)$$

Este modelo es útil para sistemas en los que solo se considera una planta, pero para el TFM va a ser necesario utilizar un modelo de varias plantas, en general, de n plantas.

2.2. Estructura de n plantas

El modelo real sobre el que se va a trabajar en el trabajo es un edificio de 2 plantas, aunque se va a experimentar también con una estructura de 5 pisos para ver los resultados. Para conocer la respuesta de los TMDs es necesario modelar el sistema edificio para n plantas y, posteriormente, se particularizará para el caso de estudio.

En el problema al que nos enfrentamos, la excitación y el movimiento del edificio están en el mismo plano, lo cual simplifica en cierta medida el problema puesto que los movimientos serán traslacionales. En la siguiente imagen puede verse el modelo para n plantas.

En ella se observa un edificio de n plantas empotrado en la base en el que cada piso sufre una excitación F_i y se desplaza una magnitud u_i. Cada planta tiene su masa concentrada en un punto y sus características (masa, rigidez y amortiguamiento estructural) son propias.

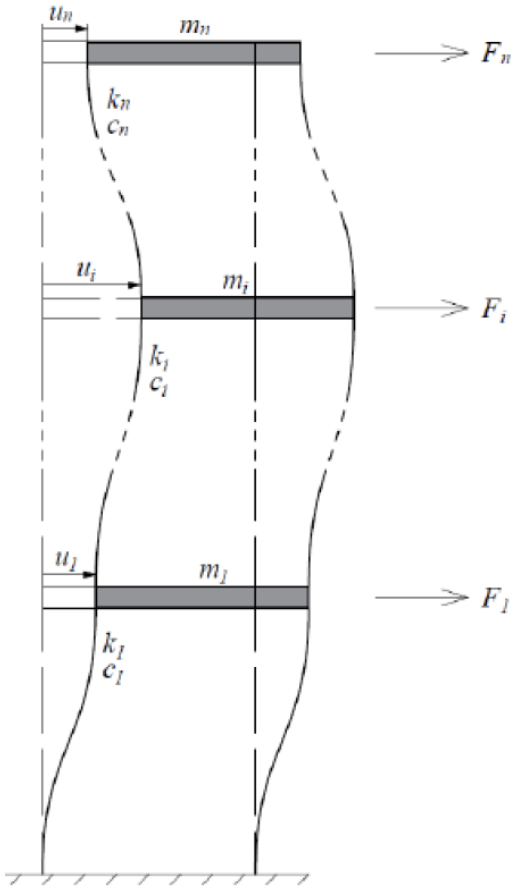

Figura 2.2 Modelo de un edificio de n plantas (Magdaleno, 2017)

Como se hizo en el apartado anterior, si se aplica la Segunda Ley de Newton se obtiene el comportamiento de la estructura:

$$F_1 = m_1\ddot{u}_1 + (c_1 + c_2)\dot{u}_1 - c_2\dot{u}_2 + (k_1 + k_2)u_1 - k_2u_2 \quad (2.5)$$

$$F_2 = m_2\ddot{u}_2 - c_2\dot{u}_1 + (c_2 + c_3)\dot{u}_2 - c_3\dot{u}_3 - k_2u_1 + (k_2 + k_3)u_2 - k_3u_3 \quad (2.6)$$

$$F_n = m_n\ddot{u}_n + (\dot{u}_n + \dot{u}_{n-1})c_n - (u_n - u_{n-1})k_n \quad (2.7)$$

Se plantean tantas ecuaciones como pisos tenga la estructura.

2.3. Modelo estructura + TMD

En los dos apartados anteriores se ha visto el modelo de un TMD y el modelo de una estructura de n plantas. El siguiente paso es unir ambos modelos de manera que se obtenga una representación matemática del comportamiento de la estructura junto con el TMD acoplado.

Para una estructura de una sola planta y aplicando una fuerza F_i, la ecuación asociada es la siguiente:

$$F_i - R_i = m_i\ddot{u}_i - c_i\dot{u}_{i-1} + (c_i + c_{i+1})\dot{u}_i - c_{i+1}\dot{u}_{i+1} - k_i u_{i-1} + (k_i + k_{i+1})u_i - k_{i+1}u_{i+1} \quad (2.8)$$

Esta ecuación es el resultado de restar a la ecuación general de una estructura de una planta (ecuación 2.5), la ecuación de la reacción del TMD en sentido contrario a la fuerza aplicada (ecuación 2.1).

Como en la ecuación 2.2 está expresada la reacción R_i en función de los parámetros del TMD, se puede sustituir en la ecuación 2.6 obteniéndose lo siguiente:

$$F_i = m_i\ddot{u}_i - c_i\dot{u}_{i-1} + (c_i + c_{i+1} + c_j)\dot{u}_i - c_{i+1}\dot{u}_{i+1} - k_i u_{i-1} + (k_i + k_{i+1} + k_j)u_i - k_{i+1}u_{i+1} \quad (2.9)$$

Esta ecuación representa el movimiento de la planta de un edificio teniendo en cuenta la rigidez y el amortiguamiento del TMD instalado.

Si se repite el procedimiento para cada piso, teniendo en cuenta la presencia o no de TMDs en cada uno de ellos, se podría expresar el sistema en conjunto de forma matricial:

$$\mathbb{M}\ddot{q} + \mathbb{c}\dot{q} + \mathbb{K}q = \mathbb{F} \quad (2.10)$$

Dado que se va a utilizar el software Matlab, el uso de matrices es muy ventajoso, puesto que el programa trabaja muy bien empleando matrices para resolver las operaciones. Además, Matlab permite operar en el espacio de estados, por lo que, a partir de la ecuación 2.9 se puede pasar al modelo de espacio de estados siguiente y proceder:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \quad (2.11)$$

- x es el vector de estado
- u es el vector de entradas
- A es la matriz de estado
- B es la matriz de entradas
- y es el vector de salidas
- C es la matriz de salida
- D es la matriz de transmisión directa

2.4. Respuesta en frecuencia

El estudio de los TMDs desde el punto de vista de su respuesta en el dominio de la frecuencia es de gran interés puesto que permite observar la excitación que experimenta la estructura a diferentes frecuencias y, de esta manera, encontrar aquellas frecuencias para las que la excitación podría comprometer la integridad del edificio de estudio. La respuesta en frecuencia relaciona las amplitudes de la respuesta y la excitación mediante la siguiente ecuación:

$$H^{ib}(\omega) = \frac{X_i(\omega)}{A_b(\omega)} \tag{2.12}$$

Las zonas en las que la respuesta se hace muy grande son aquellas para las que la frecuencia de excitación es muy parecida (o igual) a la frecuencia de resonancia de la estructura, apareciendo picos en la amplitud de la respuesta. En la siguiente imagen, puede verse la respuesta en frecuencia de un edificio de dos pisos:

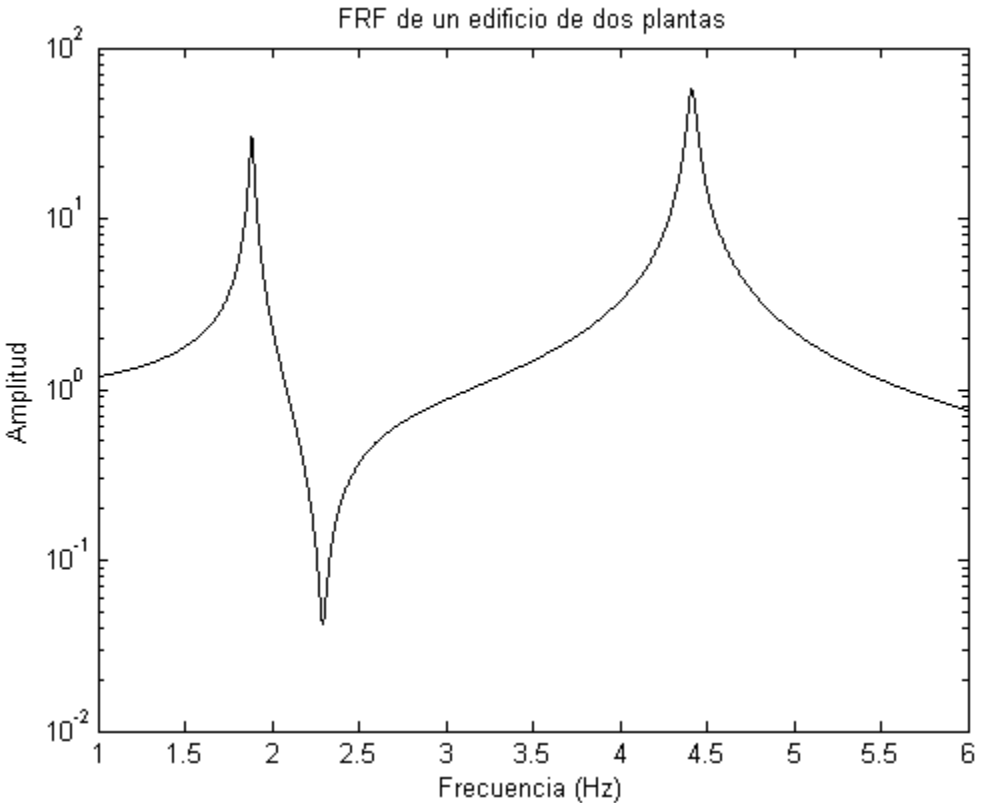

Figura 2.3 Respuesta en frecuencia de un edificio de dos pisos (Guerra, 2017)

El cometido de un TMD es eliminar o, si no es posible, reducir en la mayor medida posible los picos de amplitud para que la integridad de la estructura no se vea comprometida. Cuando la sintonización de un TMD es exitosa, se consigue reducir dicha amplitud de manera que se obtiene una respuesta en frecuencia como la indicada en la figura 2.4.

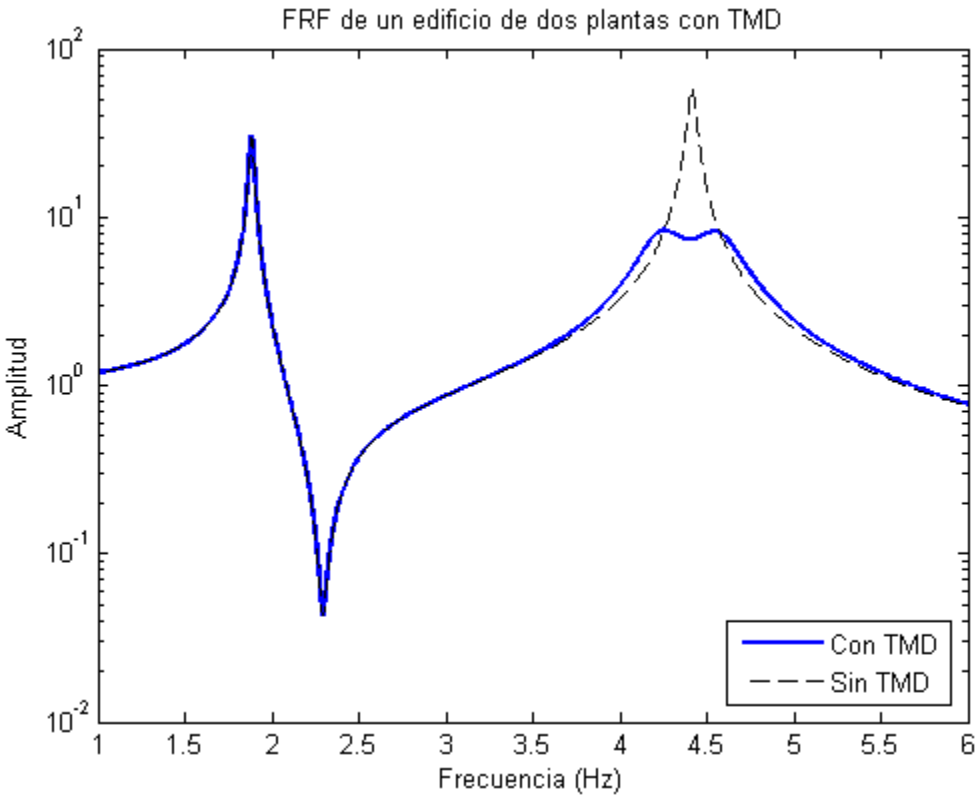

Figura 2.4 Corrección de la respuesta en frecuencia mediante la instalación de un TMD (Guerra, 2017)

(Magdaleno, 2017) define varios indicadores de respuesta en frecuencia que permiten la sintonización posterior de los TMD en función de los requerimientos en cada caso. A partir de dichos indicadores, se genera una función de coste a través de la que se sintonizan los TMDs. Dichos indicadores son:

1. Máximo local de la respuesta en frecuencia: el objetivo es minimizar el máximo de las crestas que aparecen con la instalación de un TMD (figura 2.4) en un rango de frecuencias. El rango de frecuencias normalmente está en torno a una frecuencia natural del sistema. La función de coste proporciona el máximo de las crestas en dicho rango de frecuencias. En la siguiente ecuación se muestra la definición de la función de coste:

$$\min_x \left(max\left(H(\omega)\right)\right) / \, \omega \in [\omega_1, \omega_2] \tag{2.13}$$

2. Máximo global de la respuesta en frecuencia: Cuando las frecuencias propias de la estructura se encuentran próximas unas de otras, es interesante minimizar el máximo global. Para ello, el objetivo de la función de coste es proporcionar la frecuencia para la que la respuesta del sistema se hace máxima. En este caso, la ecuación es:

$$\min_x \frac{\left(max\left(H(\omega)\right)\right)}{\omega} \in [\Omega_1, \Omega_2] \tag{2.14}$$

3. Máximo local de todas las respuestas en frecuencia: Este indicador consiste en considerar las respuestas en frecuencia de todos los pisos para optimizar el máximo local. De esta manera se consigue mejorar la respuesta de la estructura en todos los puntos de la misma cuya respuesta en frecuencia esté en torno a la frecuencia propia del sistema.

$$\min_x \left(max\left(H_1(\omega), H_2(\omega), \dots, H_n(\omega)\right)\right) / \ \omega \ \in \ [\omega_1, \omega_2] \qquad (2.15)$$

4. Máximo global de las respuestas en frecuencia: En este caso, se incluyen en la función de coste todas las crestas de todas las respuestas en frecuencia de la estructura, optimizándose el valor ara la mayor de ellas en todo el rango de frecuencias:

$$\min_x \left(max\left(H_1(\omega), H_2(\omega), \dots, H_n(\omega)\right)\right) / \ \omega \ \in \ [\Omega_1, \Omega_2] \qquad (2.16)$$

2.5. Modelo experimental

La estructura sobre la que se va a trabajar es una maqueta de dos pisos sobre la que se ha trabajado en estudios anteriores. La estructura cuenta con dos pilares y dos forjados.

Figura 2.5 Maqueta de la estructura

Los pilares están compuestos por placas de aluminio. Su cometido es aportar flexibilidad lateral a la estructura, además de restringir la posibilidad de torsión. Por lo que se trata de un movimiento unidireccional en un solo plano.

Los forjados, por su parte, se componen de placas de metacrilato, aportando rigidez al sistema en el plano vertical. De esta manera, los desplazamientos verticales de la estructura pueden despreciarse frente a los horizontales debido a que éstos últimos son varios órdenes de magnitud superiores.

3. ALGORITMOS DE OPTIMIZACIÓN (META)HEURÍSTICOS

3.1. Cálculo analítico de TMDs

En el diseño de TMDs y su instalación en estructuras, existen expresiones que permiten calcular analíticamente el lugar en el que deben situarse. Sin embargo, muchas de estas expresiones sirven exclusivamente para sistemas de un grado de libertad, por lo que se trata de problemas muy sencillos, lejos de la complejidad de los problemas de ingeniería reales.

En algunos casos, si se reducen los sistemas de varios grados de libertad a uno solamente, las expresiones pueden seguir utilizándose de forma válida.

Cuando se trata de resolver un problema de sintonización de TMDs en sistemas de más de un grado de libertad, lo que se hace es obtener los parámetros característicos (masa, rigidez y amortiguamiento) de un sistema de un solo grado de libertad forzando a que la frecuencia propia y el factor de amortiguamiento crítico del modelo de un solo grado de libertad coincida con la frecuencia de resonancia del sistema de varios grados de libertad que se estudia.

Una vez realizado lo anterior, se pueden aplicar las ecuaciones de Den Hartog para optimizar el TMD. Las ecuaciones son válidas para un sistema excitado en su base y minimizando la respuesta en aceleración:

$$f_{opt} = \frac{1}{1+u}\left(\sqrt{\frac{2-\mu}{2}}\right) \tag{3.1}$$

$$\xi_{opt} = \sqrt{\frac{3\mu}{8(1+\mu)}}\left(\sqrt{\frac{2-\mu}{2}}\right) \tag{3.2}$$

Siendo μ la relación entre las masas del TMD y el sistema según la ecuación 3.3:

$$\mu = \frac{m_{TMD}}{m_{sistema}} \tag{3.3}$$

El método analítico es útil cuando se aplica sobre sistemas muy sencillos o simplificados, pero cuando se realiza una optimización más compleja es necesario usar otros métodos más potentes, como los algoritmos genéticos.

3.2. Algoritmos para la búsqueda de soluciones

Debido a las limitaciones que tiene el cálculo analítico de TMDs, se desarrollaron otros métodos que permitieran elevar la complejidad de los problemas de sintonización de TMDs. Así, se pasó de resolver problemas que involucraban un solo TMD a estructuras en las que se optimizaba la localización de varios TMDs dentro de la misma estructura.

Algunos de los métodos de optimización están basados en la utilización de algoritmos, muchos de ellos asentados sobre procesos biológicos observables en la naturaleza. Dentro de estos algoritmos encontramos los Algoritmos Genéticos (AG), sobre los que va a tratar este trabajo.

En los siguientes apartados va a hablarse dos clases de algoritmos útiles en la búsqueda de soluciones.

3.2.1. Algoritmos heurísticos

Son aquellos basados en técnicas heurísticas, es decir, métodos de resolución de problemas fundamentados en la experiencia. En el ámbito de la ingeniería, la utilización de técnicas heurísticas permite obtener la solución de un problema a partir de la solución de otro previamente conocido.

A continuación, se indican los Algoritmos heurísticos más comunes.

3.2.1.1. Hill Climbing (Escalada Simple)

Es un método de optimización de búsqueda local. El algoritmo es iterativo. Parte de una solución inicial arbitraria y, a partir de ahí, busca en su entorno nuevas soluciones (mejores) variando un solo parámetro de la solución (Norvig, 2003).

El problema que tiene este algoritmo es que no garantiza la obtención de la mejor solución, puesto que la búsqueda puede quedarse estancada en un óptimo local de la función a optimizar. Para mejorar este método, es conveniente comenzar varias veces el proceso con soluciones iniciales que recorran el dominio de la función.

Por otra parte, tiene la ventaja de ser muy sencillo, por lo que se utiliza mucho. Además, es un algoritmo que encuentra soluciones rápidamente, por lo que cuando se dispone de tiempo limitado resulta ideal.

En la siguiente imagen se observa el problema citado anteriormente:

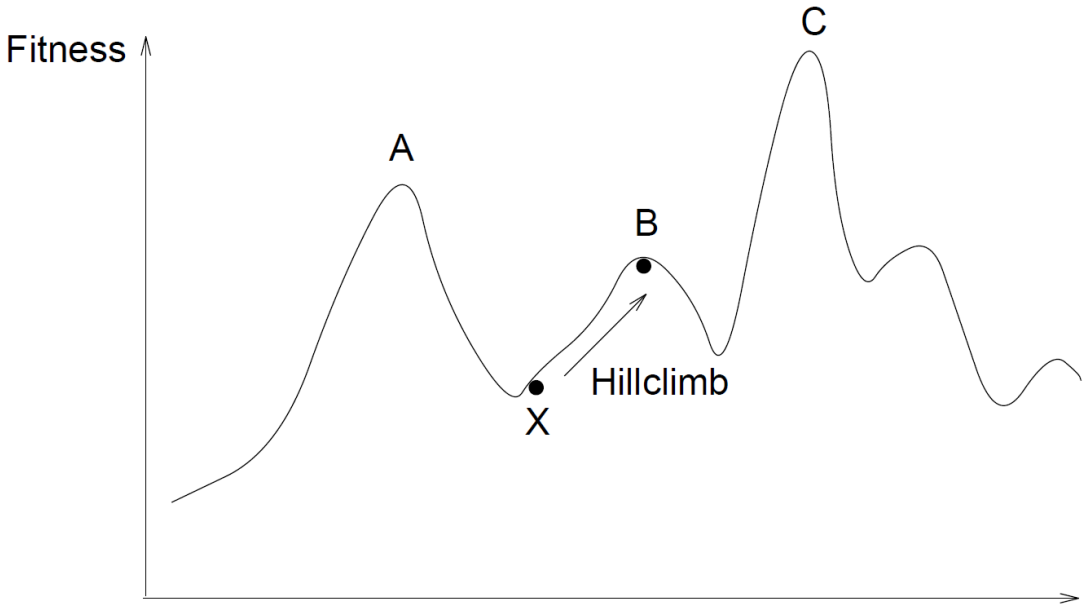

Figura 3.1 Representación gráfica del algoritmo Hill Climbing

3.2.1.2. Descenso de Gradiente

El descenso de gradiente (Norvig, 2003) es el algoritmo de minimización de cualquier función. Se trata de un método que, en ocasiones, puede resultar lento, pero es muy versátil ya que es aplicable a multitud de problemas. La velocidad de convergencia estará determinada por la forma de la función y su complejidad.

Es muy similar al método anterior (Hill Climbing), ya que también es un algoritmo iterativo. Parte de una solución inicial y busca mejores soluciones progresivamente.

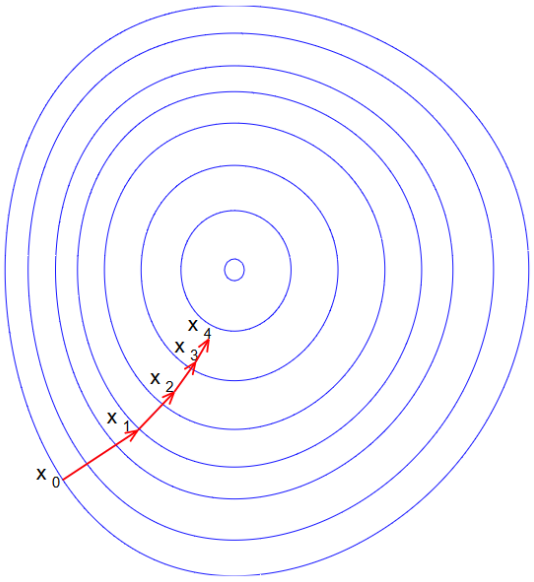

Figura 3.2 Algoritmo Descenso de Gradiente

En la figura 3.2 puede verse la manera en la que el algoritmo funciona. Es muy similar al Hill Climbing pero, en este caso sirve para minimizar la función.

3.2.2. Algoritmos metaheurísticos

Un algoritmo metaheurístico se puede definir como un método heurístico general desarrollado para resolver un problema heurístico específico. La ventaja de estos algoritmos es que la búsqueda está encaminada hacia regiones del espacio en las que hay soluciones de alta calidad.

A continuación, se van a describir algunos de los algoritmos metaheurísticos más utilizados. En el siguiente esquema se indica la clasificación de los algoritmos (Guerra, 2017) en dos grandes grupos:

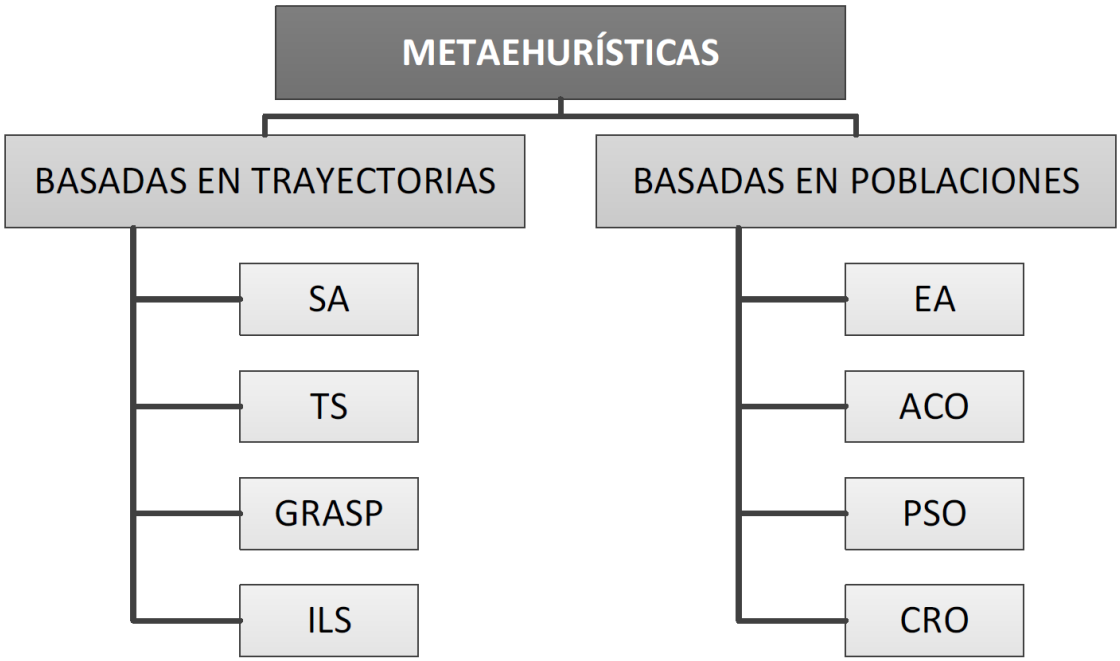

Figura 3.3 Clasificación de metaheurísticas (Guerra, 2017)

Donde las siglas se corresponden con:

SIGLA	METAHEHURÍSTICA
SA	Simulated Anneling
TS	*Tabu Search*
GRASP	Greedy Randomized Adaptative Search Procedure
ILS	Iterated Local search
EA	Evolutionary Algorithm
ACO	Ant Colony Optimization
PSO	Particle Swarn Optimization
CRO	Coral Reefs Optimization

Tabla 3.1 Siglas de las metaheurísticas

Brevemente, se va a proporcionar una definición de cada una de ellas.

3.2.2.1. *Recocido Simulado (SA)*

El Recocido Simulado (Simulated Annealing) está basado en el proceso de recocido de los metales (Kirkpatrick et al., 1993); el metal se calienta hasta una cierta temperatura y se deja enfriar hasta alcanzar un estado de mínima energía que mejora sus propiedades físicas.

El algoritmo opera de la siguiente manera: partiendo de un estado inicial s y mediante una serie de operaciones se obtiene un estado s'. Si la energía de s' es menor que la de s se cambian los estados. Si, por el contrario, la energía del nuevo estado es superior al de la original, la elección del estado dependerá de una probabilidad (función de la diferencia de energías de ambos estados) y de la temperatura del sistema.

El diagrama de flujo del sistema sería el siguiente:

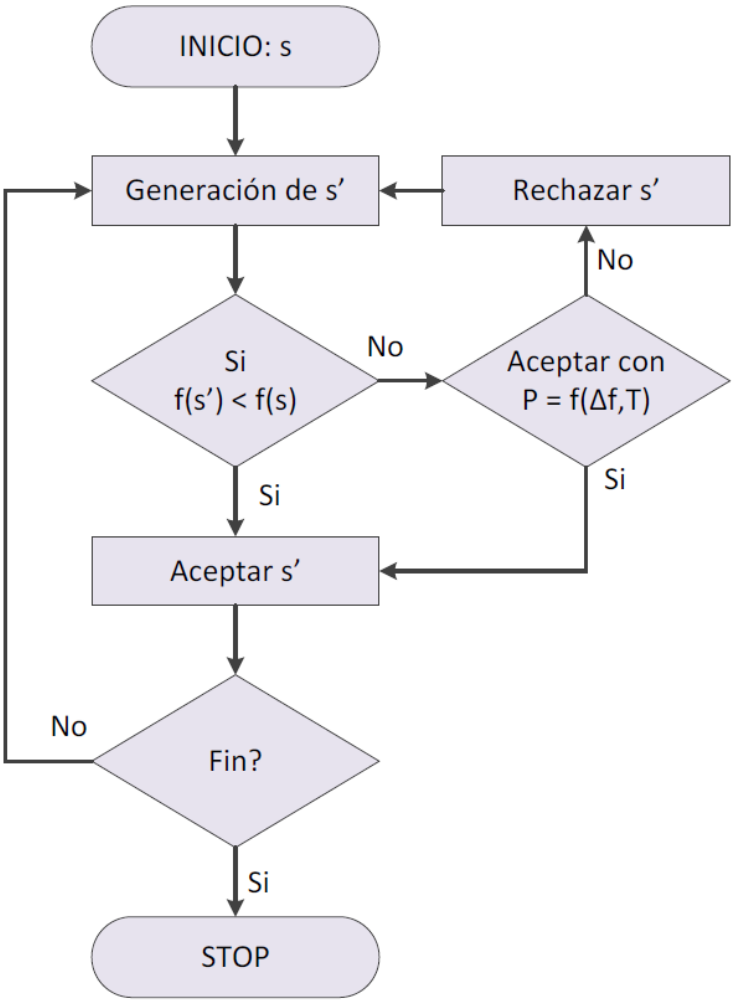

Figura 3.4 Recocido Simulado (Guerra, 2017)

3.2.2.2. Búsqueda Tabú (TS)

Este algoritmo procede partiendo de una solución inicial arbitraria (Glover, 1997) y, a partir de ella, busca en su vecindad soluciones mejores. Cuando encuentra una solución mejor, el algoritmo se desplaza a dicha solución y repite el proceso de forma iterativa.

Un aspecto a tener en cuenta cuando se utiliza este algoritmo es que no se quede estancado en los óptimos locales, por lo que la implementación debe realizarse haciendo esta consideración.

El esquema de funcionamiento del algoritmo es el siguiente:

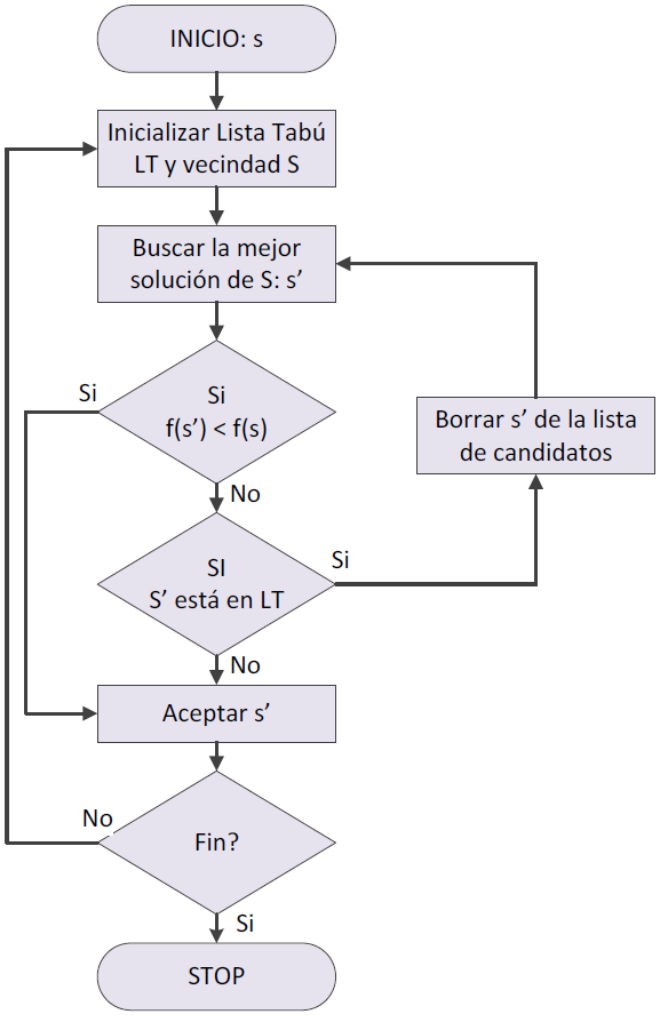

Figura 3.5 Búsqueda Tabú (Guerra, 2017)

3.2.2.3. Algoritmos Voraces (GRASP)

Es un método basado en funciones Greedy Aleatorizadas Adaptativas (Feo y Resende, 1995). Se compone de dos etapas: una de construcción y otra de mejora.

La fase de construcción tiene por objetivo generar una solución añadiendo elementos progresivamente. Los elementos que se van añadiendo depende de la función greedy, que se encarga de comprobar la mejora que supone cada elemento para la solución.

Para estudiar todo el espacio de soluciones, se genera un conjunto con las mejores y se selecciona una aleatoriamente.

La fase de mejora aplica un algoritmo a la solución construida para mejorarla.

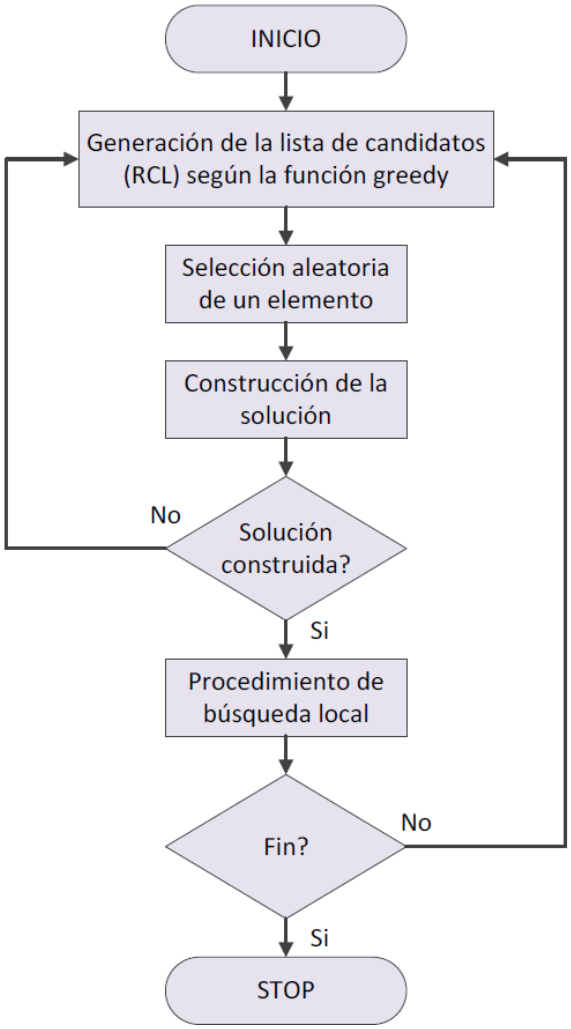

Figura 3.6 Algoritmo GRASP (Guerra, 2017)

3.2.2.4. Búsqueda Local Iterada (ILS)

Este algoritmo es una evolución de los algoritmos de búsqueda locales para evitar el estancamiento en soluciones óptimas locales (Bermúdez, Minetti y Salto, 2016). El funcionamiento es el siguiente: partiendo de una solución inicial arbitraria, se busca en el entorno de la misma mejores soluciones hasta llegar a un óptimo. A partir del óptimo encontrado, el algoritmo busca nuevas soluciones en otras regiones obteniéndose cada vez mejores soluciones.

El diagrama de flujo de este algoritmo es:

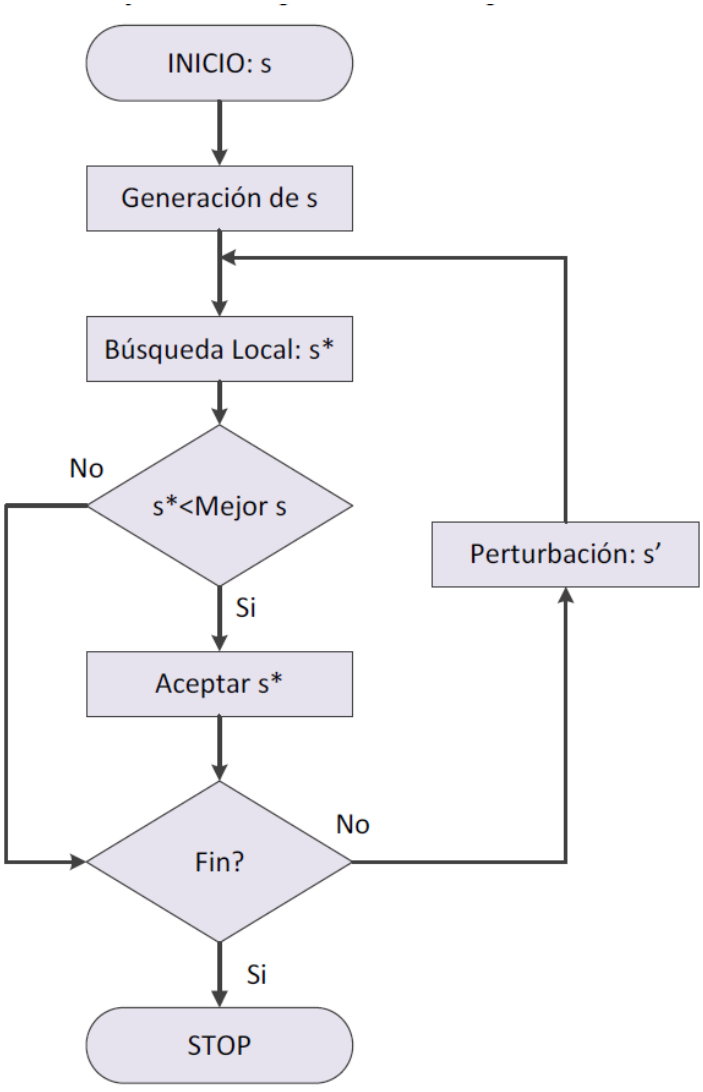

Figura 3.7 Búsqueda Local Iterada (Guerra, 2017)

3.2.2.5. Algoritmos basados en Colonias de Hormigas (ACO)

El comportamiento de las hormigas ha sido estudiado (Dorigo, 1992) y las conclusiones obtenidas es que son capaces de optimizar el proceso de recogida de comida y su transporte al hormiguero.

Las hormigas se desplazan desde el hormiguero hasta la comida y regresan. En el trayecto, van desprendiendo un rastro de feromonas. Las siguientes hormigas son capaces de detectar el rastro de feromonas y, por lo tanto, seguir la misma trayectoria. La intensidad de las feromonas se reduce con el paso del tiempo, lo que significa que el rastro más intenso es aquel para el cual el tiempo del recorrido es menor, es decir, es óptimo.

La idea del algoritmo es generar hormigas "virtuales" que encuentren la solución óptima.

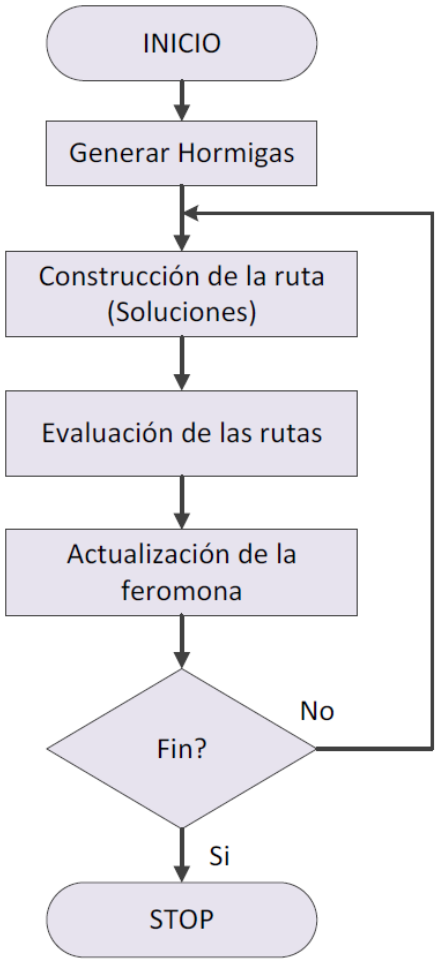

Figura 3.8 Algoritmo basado en Colonias de Hormigas (Guerra, 2017)

3.2.2.6. Algoritmos basados en Enjambres (PSO)

Estos algoritmos pretenden reproducir el comportamiento de ciertas especies que se agrupan, como las abejas (enjambres) o los peces (bancos) (Kennedy y Eberhart, 1995).

El algoritmo parte de un conjunto de posibles soluciones (partículas) que se mueven por el espacio de búsqueda siguiendo normas matemáticas. El desplazamiento de la partícula se ve afectado por las mejores soluciones halladas por el resto de partículas, por lo que el proceso obtiene mejores soluciones cada vez.

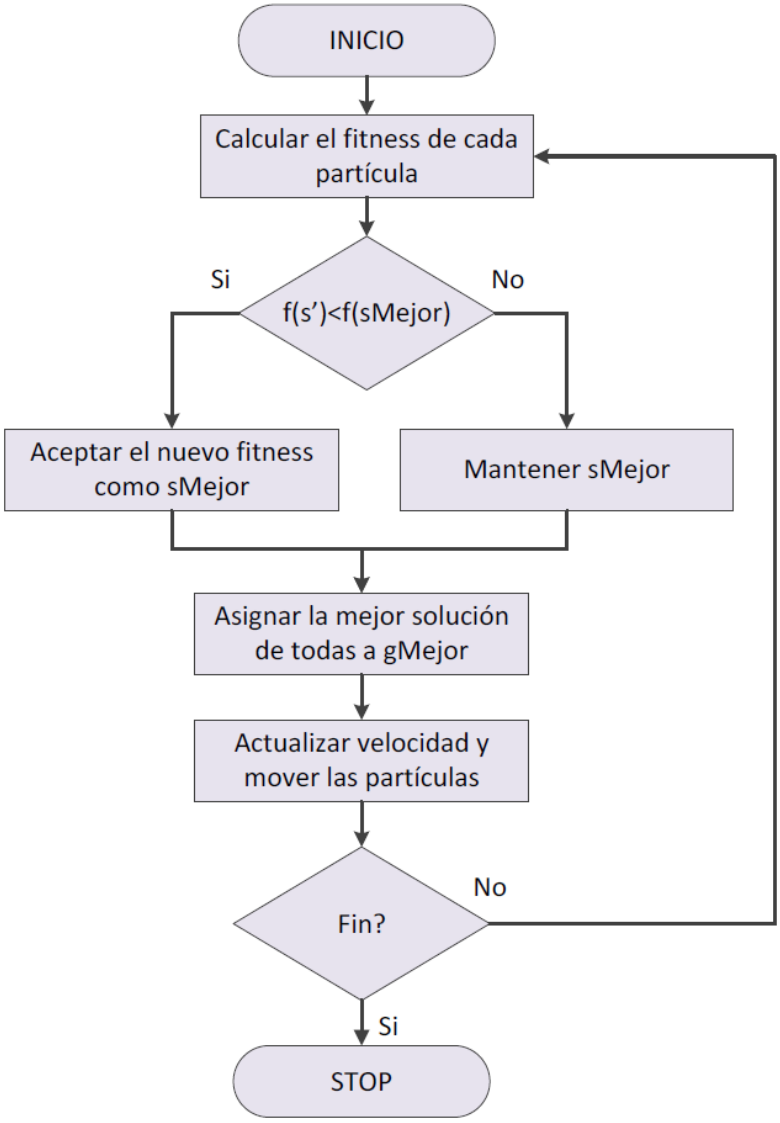

Figura 3.9 Algoritmo basado en Enjambres de Partículas (Guerra, 2017)

3.2.2.7. Algoritmos basados en los Arrecifes de Coral (CRO)

La forma en la que los arrecifes de coral se reproducen y crecen es un proceso implementable mediante un algoritmo para la resolución de problemas (Salcedo-Sanz, 2014).

El algoritmo parte de una matriz nxm en la que cada elemento puede alojar una solución (coral). Los primeros elementos se generan aleatoriamente, dejando algunos vacíos (zonas en las que los corales pueden asentarse en el futuro).

Cada componente o solución tiene asignado un valor o "fitness" asociado al problema. Cuanto mejor sea ese valor en relación al problema, mejor será dicha solución y, por lo tanto, el arrecife se desarrollará a partir de estos elementos, mientras que aquellos que tienen peor "fitness", desaparecen.

El algoritmo se apoya en cuatro funciones: Reproducción Sexual en la que dos padres generan una larva; Reproducción Asexual que, partiendo de un coral, se obtiene una larva mediante mutación; Asentamiento de Larvas en la que, cuando el elemento en el que deben asentarse está vacío, lo ocupan, mientras que si está ocupado se compara el valor del "fitness" y el lugar lo ocupa aquel con mayor magnitud; Depredación en la que algunos corales mueren dejando libre su lugar.

El esquema de funcionamiento del algoritmo sería el siguiente:

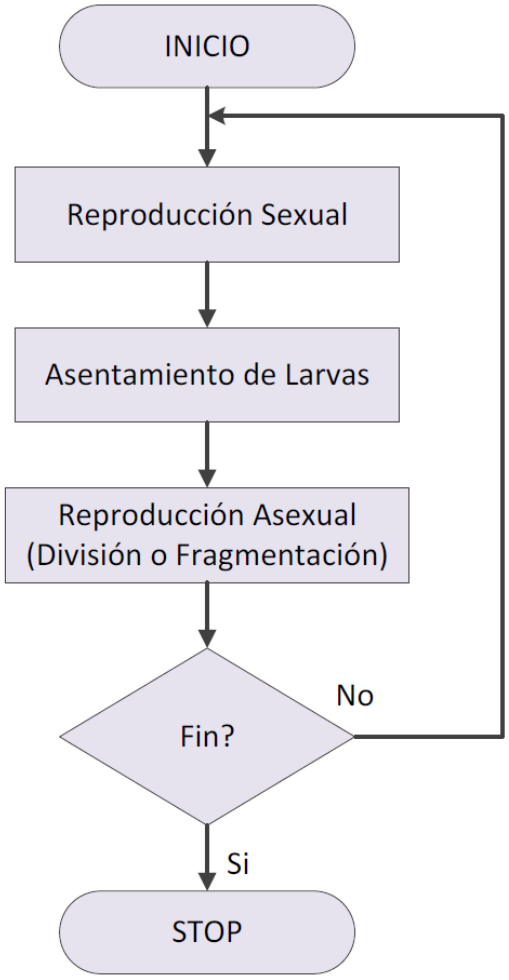

Figura 3.10 Algoritmo basado en Arrecifes de Coral (Guerra, 2017)

3.2.2.8. Algoritmos Evolutivos (EA)

Como su propio nombre indica, los Algoritmos Evolutivos se fundamentan en la evolución natural.

Para su desarrollo, se comienza con un conjunto de individuos (posibles soluciones del problema) y se van realizando operaciones sobre los mismos modificándolos de manera que se obtengan nuevos individuos que se ajustan mejor al problema (soluciones de más rendimiento). El proceso es iterativo hasta que se llega a una solución cercana a la óptima.

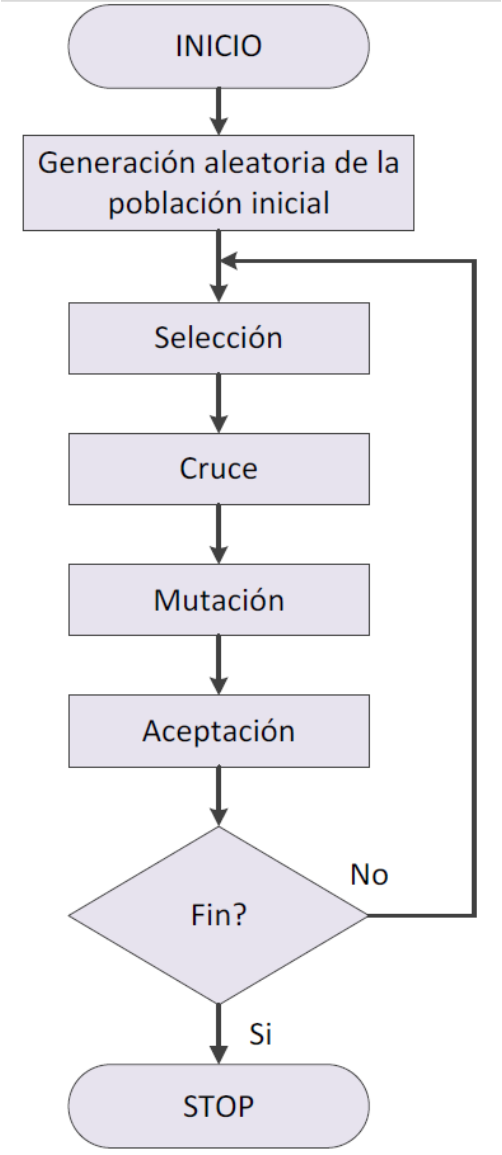

Figura 3.11 Algoritmo Evolutivo (Guerra, 2017)

Los Algoritmos Genéticos (o Evolutivos) son métodos basados en el comportamiento genético de organismos biológicos (Beasley, 1993). Las poblaciones de las distintas especies naturales evolucionan de acuerdo a los principios de la selección natural y de la supervivencia del individuo más "apto". Mediante la adaptación de estos comportamientos biológicos de forma adecuada, se pueden posteriormente utilizar para la resolución satisfactoria de problemas de muy diferentes ámbitos, como el de la ingeniería que ocupa este TFM.

De forma general, los AGs trabajan con un grupo de individuos que, en función del problema que se trate, tendrán unas características u otras (por ejemplo, en el caso de este trabajo, las características serán los parámetros característicos de los TMDs, entre otros). A partir de los individuos originales, el AG opera como la propia naturaleza. Los mejores individuos progresarán y se reproducirán entre ellos, generando una descendencia con mejores genes cada vez. Al cabo de varias generaciones (que se corresponden con iteraciones del AG) se habrán obtenido individuos de la suficiente calidad para el problema, siendo la solución al problema planteado. Habitualmente, el AG termina convergiendo, lo que significa que ha encontrado la solución óptima.

La ventaja de los AGs es que normalmente se trata de técnicas robustas que pueden aplicarse a muchas áreas distintas. Sin embargo, hay que tener en cuenta que no siempre encuentran la solución óptima, puesto que no son infalibles, pero proporcionan resultados aceptablemente buenos y con la cierta celeridad.

Aunque los Algoritmos Genéticos son muy útiles para multitud de problemas, no son los únicos algoritmos basados en la naturaleza. Las Redes Neuronales también están basados en procesos naturales (en este caso, en el comportamiento de las neuronas del cerebro) y son aplicables a problemas como reconocimiento de patrones, tratamiento de imágenes, aprendizaje de máquinas…

Principios Básicos

Puesto que el TFM va a utilizar algoritmos genéticos, se van a introducir aquí los conceptos básicos de los mismos (Beasley, 1993):

1. *Codificación:* En función de las necesidades del problema planteado, es necesario realizar una codificación de los parámetros a optimizar. Dicha codificación contendrá los genes de los individuos y, por lo tanto, a medida que evolucionan se irán almacenando los genes de los nuevos individuos generados.
2. *Función de aptitud (fitness):* Para cada individuo, la función de aptitud evaluará la capacidad de dicho individuo dentro de los requerimientos del problema, y devolverá un valor (fitness) que posteriormente permitirá comparar los individuos entre sí y discernir cuales son los mejores.
3. *Reproducción:* Una vez que los individuos han sido analizados por la función de aptitud, los padres se reproducirán teniendo en cuenta cuáles son los mejores individuos. La reproducción tiene dos variantes:
    - Cruce: Se cruzan dos individuos de manera que los genes de la siguiente generación son una mezcla de los genes de los padres. Dicha mezcla puede realizarse de muchas maneras.
    - Mutación: Se aplica a cada hijo individualmente. Se alteran, normalmente de forma aleatoria, los genes con una probabilidad reducida. Aunque la reproducción ha resultado ser el mejor método para explorar el espacio total de soluciones, la mutación permite realizar búsquedas aleatorias asegurando que ninguna parte del espacio de soluciones tiene probabilidad nula de ser examinada.

4. *Convergencia:* Si la implementación de AG es correcta, la población de individuos evolucionará de tal manera que se llegue a la solución óptima. Cabe decir que no siempre se alcanza dicha solución óptima.

3.3. Algoritmos Evolutivos

Anteriormente, se han introducido muy brevemente los Algoritmos Evolutivos. Ahora se va a profundizar más en ellos. Como su propio nombre indica, se trata de métodos basados en los principios de la genética y la selección natural. Algunas de las ideas fundamentales de la genética se adoptan y utilizan para construir algoritmos de búsqueda que son robustos y requieren una cantidad mínima de información del problema.

Los algoritmos evolutivos engloban una serie de técnicas basadas en la teoría Darwiniana de la evolución natural. Para programar un proceso evolutivo en un software, es necesario:

- Codificar las estructuras que se modificarán. De esta manera se consigue almacenar los individuos de la población.
- Operadores que modifiquen la estructura de los individuos.
- Una función que permita obtener la aptitud ("fitness") de los individuos para poder comparar las soluciones entre sí y discernir cuales son mejores.
- Un mecanismo de selección.

Cuando se habla de algoritmos evolutivos, se distinguen tres paradigmas fundamentales, estableciéndose la siguiente clasificación (Santana y Coello, 2006):

1) Programación Evolutiva
2) Estrategias Evolutivas
3) Algoritmos Genéticos

A continuación, se explicarán brevemente cada uno de ellos.

3.3.1. Programación Evolutiva

La programación evolutiva se caracteriza por centrarse en los nexos de comportamiento de padres e hijos en lugar de simular los operadores genéticos específicos (tal y como hacen los algoritmos genéticos).

El desarrollo general de la programación evolutiva es el siguiente:

- Generar aleatoriamente una población inicial.
- Aplicar la mutación
- Calcular la aptitud de cada individuo generado y, mediante selección por torneo, determinar aquellos individuos que formarán parte de la generación sucesiva.

Como se puede observar, en este caso no se aplica el operador cruce, puesto que la programación evolutiva está basada en la evolución a nivel de especie. De la naturaleza sabemos que diferentes especies no pueden cruzarse entre sí. Además, utiliza una selección probabilística.

Algunas de las aplicaciones de la programación evolutiva son:

- Predicción.
- Generalización.
- Juegos.
- Control automático.
- Problema del viajero.
- Planificación de rutas.
- Diseño y entrenamiento de redes neuronales.
- Reconocimiento de patrones.

3.3.2. Estrategias Evolutivas

Las estrategias evolutivas fueron ideadas inicialmente para resolver problemas hidrodinámicos de gran complejidad en Alemania.

Originalmente, se generaba un padre y, a partir de él, un hijo, que sustituye al padre solo si es mejor que él. En caso contrario se elimina (selección extintiva).

Más adelante se introdujo el concepto de población. En este caso, se utiliza un conjunto de padres que dan lugar a un hijo. El hijo se compara con los padres y reemplaza al peor de ellos si procede (su aptitud debe ser mayor que la del peor padre).

Otra versión de esta técnica consiste en generar múltiples hijos a partir de la población de padres y, posteriormente, realizar la selección de los mejores individuos. La selección se puede realizar de dos maneras:

1) Los mejores individuos del conjunto padres e hijos sobreviven.
2) Sólo los mejores hijos de la generación creada sobreviven.

Los operadores de cruce de las estrategias evolutivas pueden ser de dos clases:

- Sexuales: Se seleccionan dos individuos del conjunto de padres aleatoriamente y se cruzan.
- Panmíticos: Se elige un padre que permanecerá fijo y se elige aleatoriamente el segundo padre.

Las estrategias evolutivas tienen un campo de aplicación muy amplio:

- Problemas de rutas y redes.
- Bioquímica.
- Óptica.
- Diseño en ingeniería.
- Magnetismo.

3.3.3. Algoritmos Genéticos

Los algoritmos genéticos se centran en la importancia del cruce frente a la mutación (aunque utiliza ambos) y usa una selección probabilística. La idea general de los algoritmos genéticas es la siguiente:

- Generar una población inicial (normalmente de forma aleatoria).
- Calcular la aptitud de cada individuo.
- Seleccionar a los mejores individuos en base a dicha aptitud.
- Aplicar los operadores genéticos.
- Iterar hasta que se satisfaga la condición.

La representación tradicional para este tipo de problemas es una cadena binaria que contiene una serie de bits.

Para poder aplicar el algoritmo genético se necesitan cinco elementos:

1) Una representación de las soluciones.
2) Una herramienta para obtener la población original (normalmente se realiza de forma aleatoria).
3) Una función que evalúe la adecuación de las soluciones al problema (aptitud o “fitness”).
4) Operadores genéticos que modifiquen la estructura de los padres para obtener nuevos individuos (hijos) de las generaciones futuras.
5) Parámetros necesarios para el algoritmo: tamaño de la población, probabilidades de cruce y mutación, etc.

Al igual que los paradigmas anteriores, el campo de aplicación de los algoritmos genéticos es muy amplio:

- Optimización.
- Bases de datos.
- Reconocimiento de patrones.
- Planificación de movimientos de robots.
- Predicción.

3.3.4. Ventajas de las técnicas evolutivas

Cabe destacar las ventajas que tienen los procedimientos evolutivos en la resolución de problemas de búsqueda y optimización tales como al que se enfrenta esta TFM. Algunas de las ventajas principales son:

- Simplicidad conceptual.
- Gran campo de aplicación.
- Capacidad superior a las técnicas tradicionales.
- Potencial para combinarse con otras técnicas de búsqueda y optimización (adaptabilidad).
- Útiles para estructuras en paralelo.
- Son robustas ante cambios dinámicos.

Para finalizar, es importante resaltar que se trata de métodos capaces de encontrar soluciones a problemas reales de gran complejidad, pero, hay que tener en cuenta que no garantizan la convergencia al óptimo del problema dado. Por lo tanto, la aplicación de las técnicas evolutivas debe hacerse siempre comprobando la adecuación de las soluciones al problema.

3.4. Algoritmos Genéticos Simples (AGS)

Los Algoritmos Genéticos pertenecen a los denominados Algoritmos Evolutivos (Pérez, 2010). Se basan en comportamientos presentes en el proceso evolutivo de las especies. Mediante mecanismos de imitación y adaptación de los mismos a un problema se pueden obtener soluciones.

La diferencia fundamental entre un AG y una metaheurística de optimización es el uso de conjuntos de soluciones simultáneamente durante la búsqueda de soluciones. Además, el AG mejora sus soluciones progresivamente mediante la reproducción a lo largo de las generaciones sucesivas de la población. De esta manera las soluciones iniciales van evolucionando a lo largo del proceso.

3.4.1. Estructura de los AGS

El pseudo código de un AGS es el siguiente:

1. Elegir la población inicial de individuos.
2. Obtener la aptitud de cada uno de ellos (fitness) mediante la función objetivo correspondiente.
3. Repetir cíclicamente la siguiente secuencia:
    a. Seleccionar los mejores individuos para la reproducción.
    b. Generar una nueva población a partir del cruce y la mutación.
    c. Evaluar los nuevos individuos creados con la función objetivo.

d. Sustituir los padres con los hijos obtenidos.

El código indicado anteriormente está reflejado en la siguiente figura mediante un diagrama de flujo:

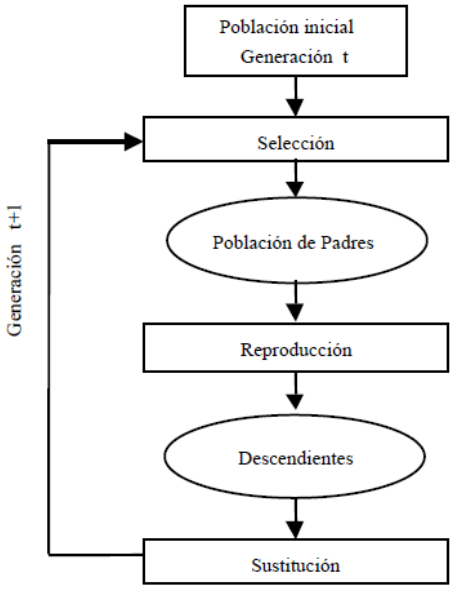

Figura 3.12 Diagrama de flujo AGS (Pérez, 2010)

3.4.2. Problema a resolver

Es fundamental realizar la formulación completa del problema a resolver: función objetivo, restricciones, etc.

En la mayoría de los problemas de optimización, existe una función de calidad (*fitness function*) que permite obtener el valor de una solución. La función de calidad solo puede utilizarse si todas las soluciones obtenidas son positivas y la función está definida de manera que el problema sea de maximización.

Dada la función *g(x)*, una función de beneficio *f(x)* puede definirse como:

$$f(x) = \begin{cases} Cmax - g(x)\ Cuando\ g(x) < Cmax \\ \\ 0\ Resto\ de\ casos \end{cases} \quad (4.11)$$

Siendo Cmax el máximo valor observado hasta el momento o en una serie de generaciones k.

Cuando puede obtenerse un valor negativo, la transformación de la función puede ser:

$$f(x) = \begin{cases} u(x) + Cmin\ Cuando\ g(x) < Cmax \\ \\ 0\ Resto\ de\ casos \end{cases} \quad (4.12)$$

En este caso, Cmin es igual al menor valor negativo de u(x) (en valor absoluto) encontrado en las últimas k generaciones.

3.4.3. Codificación de las soluciones

Existen distintos tipos de codificación de las soluciones. Según (Fang, 1994), las principales consideraciones a tener en cuenta para diseñar la codificación son:

1. Cada solución del problema tendrá su correspondiente cadena codificada.
2. Para cada cadena codificada producida mediante los operadores genéticos existirá su correspondiente solución decodificada.
3. Codificación y solución se corresponderán una a una (no habrá redundancia).
4. La codificación deberá permitir heredar de padres a hijos las características inherentes a los primeros.

3.4.3.1. Codificación binaria

Esta codificación ha sido la más utilizada por la fácil adaptabilidad a los sistemas informáticos y porque permite aplicar operadores genéticos simples.

En este tipo de codificación, destaca el código *Gray*, utilizado en lugar del binario porque la separación que existe en el espacio codificado entre soluciones adyacentes en el espacio decodificado siempre es 1. Al aplicar operadores genéticos, lo anterior es de gran importancia al aplicar operadores genéticos.

La codificación binaria presenta poca adaptabilidad a muchos problemas, por lo que su utilización reduce la eficiencia del algoritmo y, por lo tanto, no tiene mucho sentido usarla.

3.4.3.2. Codificación simbólica

Esta codificación permite mayor libertad para la codificación de problemas, pero el diseño debe ser minucioso puesto que una codificación inapropiada puede hacer que el algoritmo fracase.

3.4.3.3. Codificación permutacional

La codificación permutacional se utiliza en problemas en los que es necesario ordenar algo. Cada solución del problema se podría obtener permutando los *n* números de la cadena y siendo el espacio de búsqueda de tamaño *n!*.

Esta codificación sirve para problemas de programación de operaciones del tipo flow shop.

Presenta una modificación (*codificación permutacional con repetición*) en la que la cadena que representa a la solución se repite tantas veces como sea necesario.

3.4.3.4. Codificación real

Esta última codificación utiliza números reales en el que el valor de la solución es, directamente, el valor real asignado.

3.4.4. Inicialización

La forma más eficaz de obtener individuos de la población original es mediante generación aleatoria (Michalewicz, 1995), ya que los individuos estarán uniformemente distribuidos por el espacio de búsqueda. Para mejorar las soluciones, será necesario realizar iteraciones que irán aumentando la calidad de las mismas.

Otra manera de inicializar el problema es utilizar metaheurísticas más sencillas que encuentran distintas soluciones que formarán parte de la población inicial. El problema que presenta esta opción es que no se garantía que las soluciones estén distribuidas por todo el espacio de búsqueda.

A la hora de inicializar es importante tener en cuenta que la generación aleatoria obtiene un mayor número de soluciones, lo cual es bueno desde el punto de vista de la exploración, pero hace que la convergencia a la solución óptima sea más lenta. Por otra parte, la utilización de metaheurísticas permite una mayor rapidez de convergencia, pero puede ocurrir que estemos centrándonos en una zona concreta del espacio de búsqueda (explotación) y se pierda diversidad del conjunto de soluciones.

3.4.5. Selección

En la naturaleza, la selección determina los individuos que sobreviven y se reproducen y cuales desaparecen. Los algoritmos genéticos imitan este comportamiento natural y lo aplica al conjunto de soluciones. Así, las soluciones que pasan de generación en generación tienen las mejores características en función del problema a resolver.

Un concepto importante es la presión selectiva. Representa el grado con el que las mejores soluciones son favorecidas para formar parte de la población de padres. Cuanto mayor sea esa presión, más se promocionan las mejores soluciones.

El valor de la presión selectiva influye en gran medida sobre la ratio de convergencia del algoritmo. Cuanto más grande sea la presión, más rápida es la convergencia, pero cabe la posibilidad de que se obtenga una solución que no sea la óptima debido a la pérdida de diversidad (explotación de una zona reducida del espacio de búsqueda). Por lo tanto, hay que encontrar un compromiso entre ambos aspectos.

3.4.5.1. Métodos proporcionales a la calidad

La selección de los padres se realiza considerando la magnitud de la calidad de las soluciones respecto a l del resto de la población. Aquellos individuos con valores de calidad muy altos llenarán la población de padres, lo que genera falta de diversidad.

Las variantes fundamentales de esta selección son:

- Selección proporcional: la forma de proceder es asignar probabilidades a las distintas soluciones. Cuanto mejor sea la calidad de una solución, más probabilidades tendrá de ser seleccionada. Se puede representar por una ruleta con un solo marcador. Se realizan *n* tiradas para obtener los individuos seleccionados.
- Selección estocástica por restos: Asigna a cada individuo un número de copias en la población de padres igual a $[p_i \ x \ n]$, redondeando al menor número entero más cercano. Con los restos se aplica el método proporcional para obtener las copias vacías.
- Selección estocástica universal: Similar a la selección proporcional usando *n* marcadores igualmente distanciados. En este caso, la representación es una ruleta con *n* marcadores, por lo que de una tirada se seleccionan todos los individuos.

3.4.5.2. Métodos basados en el orden o métodos de ordenación

Estos métodos seleccionan a los individuos en función de la posición que ocupan en una lista ordenada según la calidad. De esta manera se independiza la presión selectiva de los rangos en los que se mueve la calidad. Hay distintas variantes de estos métodos:

- Selección por ordenación lineal: La probabilidad para el individuo i-ésimo de la clasificación decreciente de calidades será:

$$\begin{gathered} p(i) = q - (i-1) \cdot r \\ q = r \cdot \frac{n-1}{2} + \frac{1}{n} \\ r \in \left[0, \frac{2}{n \cdot (n-1)}\right] \end{gathered} \tag{4.13}$$

  En la ecuación 4.13, n es el tamaño de la población y r el parámetro que controla la presión selectiva.

- Selección por ordenación no lineal: Desarrollado por (Michalewicz, 1995). La diferencia entra las probabilidades de selección no es igual a lo largo de toda la ordenación. Se rige por la ecuación:

$$p(i) = q \cdot (1-q)^{i-1} \tag{4.14}$$

  Siendo q el parámetro que permite controlar la presión selectiva. Así se favorece a los individuos mejor clasificados.

3.4.5.3. Métodos basados en competiciones o torneos

Para este tipo de selección, el padre se obtiene mediante un torneo entre z individuos seleccionados de forma aleatoria, siendo el ganador el de mejor calidad. El número de individuos que compiten viene determinado por el valor de z.

Hay algunas modificaciones de los métodos de torneo recogidas por (Horn *et al*., 1994) diseñadas para problemas multimodales y multiobjetivo para controlar el número de individuos que pasan a la siguiente población en función del lugar del espacio de búsqueda en el que se encuentran (así se mantiene la diversidad).

3.4.6. Operadores genéticos: Reproducción

La reproducción se realiza intercambiando el material genético de los dos padres mediante un operador cruce y, por otra parte, sobre las cadenas obtenidas se realiza la mutación.

Los principales operadores de la reproducción son el cruce y la mutación.

3.4.6.1. Operadores cruce

Es el operador fundamental de los algoritmos genéticos, puesto que es el que permite intercambiar material genético entre los individuos y, por lo tanto, es el operador responsable de la evolución de las soluciones. Su aplicación está determinada por una probabilidad asignada arbitrariamente.

Para realizar el cruce, los padres deben ser seleccionados en función de sus características genéticas. Así se tienen más garantías de obtener hijos con propiedades aún mejores a lo largo de las generaciones. Es importante que la programación del cruce sea tal que se cumplan estos objetivos. Si no se aprovechan las características de los padres, se obtienen nuevos individuos en los que le material genético ha cambiado en su mayoría (tal como ocurre en una mutación) y, de esta manera, se estarían desperdiciando recursos de computación.

Los principales operadores cruce existentes son los siguientes:

- Operador con un punto de cruce: Es el operador clásico para la codificación binaria. La combinación de las cadenas de material genético de los padres se realiza mediante un número aleatorio.

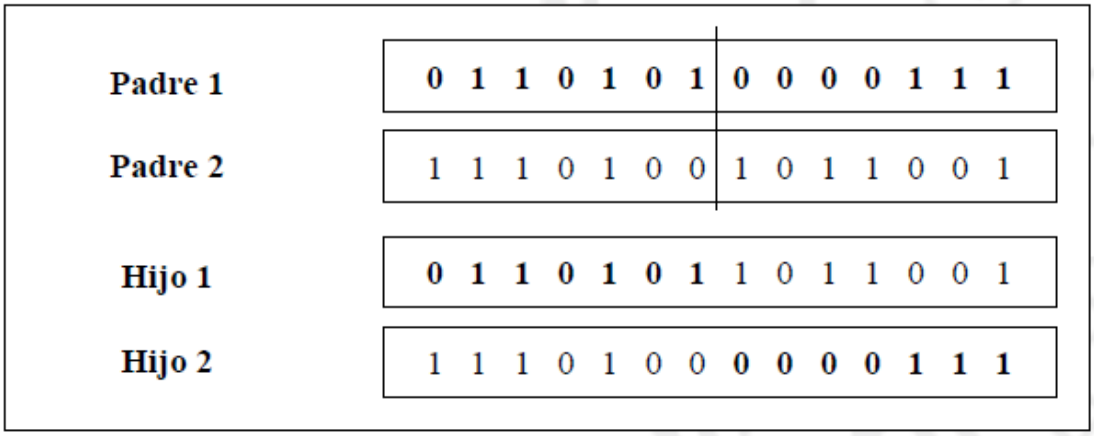

Figura 3.13 Operador con un punto de cruce (Pérez, 2010)

- Operador con N puntos de cruce: En lugar de utilizar un único número aleatorio se usan tantos como se quiera y se mezcla el material genético de igual manera que en el método anterior.
- Operador cruce uniforme: Es la generalización de los operadores anteriores. El intercambio de las cadenas se realiza gen a gen con una probabilidad del 50% para cada uno de ellos.
- Operador cruce ordenado (OX): Este operador es el utilizado para la codificación permutacional. Se selecciona un segmento de las cadenas de los padres y se mantiene

sin cambios en los hijos. El resto de genes se van intercambiando. Para el hijo 1 se utilizan los genes del padre 2 y para el hijo 2 los del padre 1 (figura 4.5).

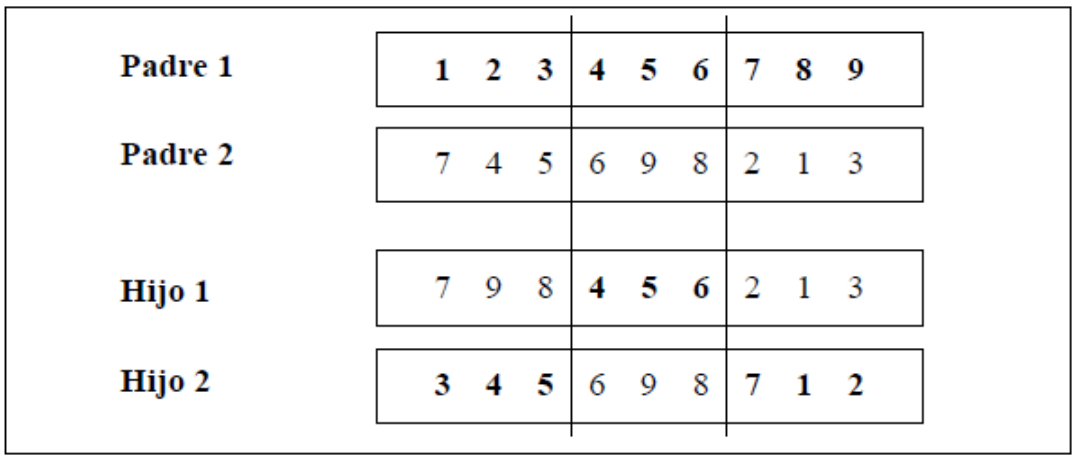

Figura 3.14 Operador cruce ordenado (Pérez, 2010)

- Cruce generalizado basado en el orden (GOX): Utilizada para la codificación permutacional. Es similar al método anterior, pero en este caso para seleccionar la subcadena invariable que pasa a los hijos se obtiene mediante dos números aleatorios (figura 4.6).

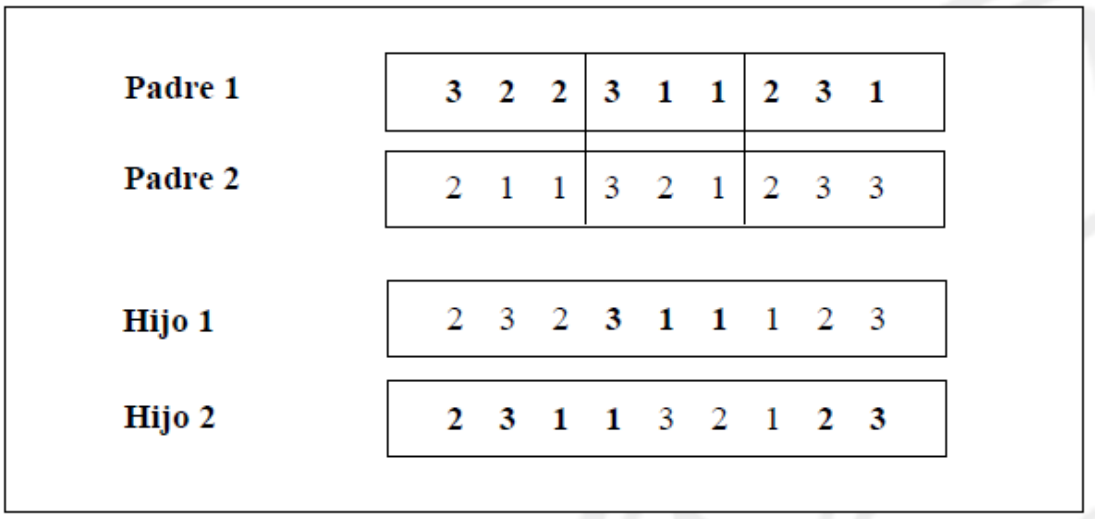

Figura 3.15 Operador cruce generalizado basado en el orden (Pérez, 2010)

Como se ha indicado en la descripción de cada uno de los métodos, es importante destacar que los operadores deben estar adaptados a la codificación utilizada.

3.4.6.2. Operadores mutación

La importancia de los operadores mutación dentro de los AG radica en que, al contrario que los operadores cruce (que se encargan de potenciar las buenas características de los individuos a lo largo de las generaciones), mantienen la diversidad de la población. Permite recuperar características de individuos que, siendo buenas, podrían perderse con la selección y el cruce.

Los operadores mutación fundamentales son:

- Mutación estándar: Es el más utilizado en la codificación binaria. La probabilidad asociada al operador debe ser reducida para evitar introducir demasiada diversidad que sea contraproducente en el desarrollo del AG. El proceso se realiza gen a gen y se cambia según la probabilidad elegida (figura 4.7).

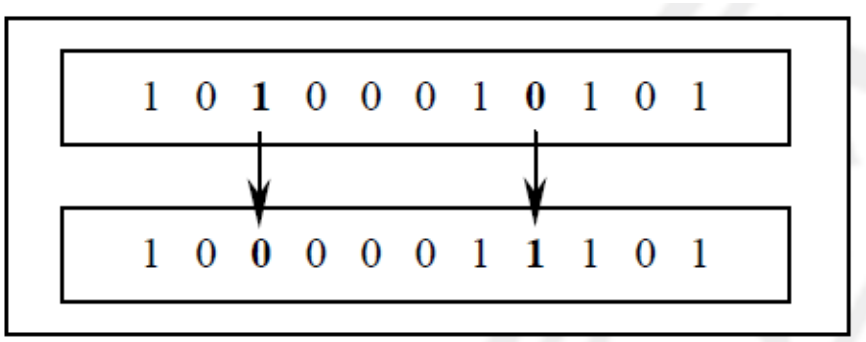

Figura 3.16 Operación mutación estándar (Pérez, 2010)

- Mutación no uniforme: Es el que se utiliza para la codificación real. Se diseñó este operador porque el operador estándar introducía demasiados cambios en la codificación real.
  Dado un cromosoma formado por m variables, $S^t = \{v_1, v_2, \dots, v_k, \dots, v_m\}$ cuyo elemento v_k es seleccionado para mutar, el cromosoma resultante es la siguiente: $S^{t+1} = \{v_1, v_2, \dots, v'_k, \dots, v_m\}$ donde:

$$v'_k = \begin{cases} v_k + \Delta(t, UB - v_k) \; si \; un \; número \; aleatorio \; es \; 0 \\ v_k + \Delta(t, v_k - LB) \; si \; es \; 1 \end{cases} \quad (4.15)$$

  En la ecuación 4.15, UB y LB son los límites superior e inferior, respectivamente, del dominio de la variable v_k. El valor Δ(t,y) devuelve un valor comprendido entre 0 e y.

$$\Delta(t, y) = y \cdot \left(1 - r^{\left(1 - \frac{t}{T}\right)^b}\right) \begin{cases} r \; número \; aleatorio \; entre \; 0 \; y \; 1 \\ T \; número \; total \; de \; generaciones \\ b \; grado \; de \; dependencia \; con \; el \; núm. de \; iteración \end{cases} \quad (4.16)$$

  Se recomiendo usar b = 5. De esta manera se incentiva la exploración (diversidad) en las generaciones iniciales y la explotación en las finales.

- Mutación basada en el orden (OBM): Este operador se aplica para codificaciones permutacionales con y sin repetición. Se intercambian dos genes de la cadena entre sí.

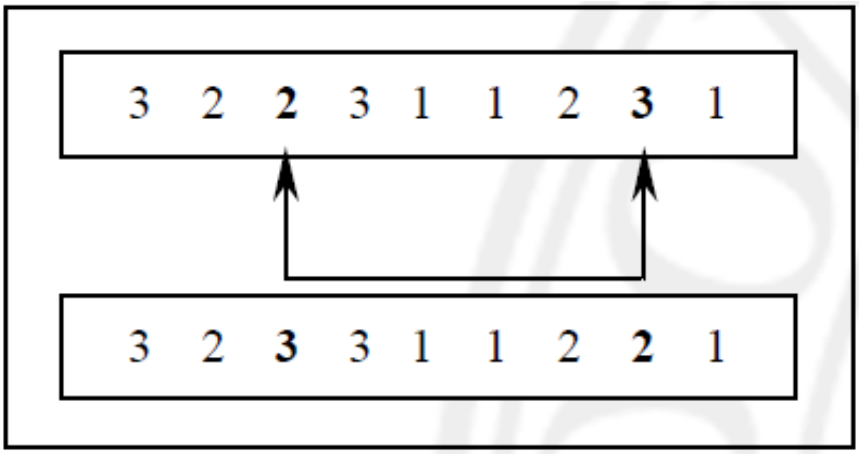

Figura 3.17 Mutación basada en el orden (Pérez, 2010)

- Mutación por permutación de un subconjunto (SSM): Diseñada para la codificación permutacional y aplicable también para la codificación permutacional con repetición. Se seleccionan dos posiciones de la cadena de genes. Los valores comprendidos entre ambas posiciones se permutan aleatoriamente y el resto se mantienen invariantes.

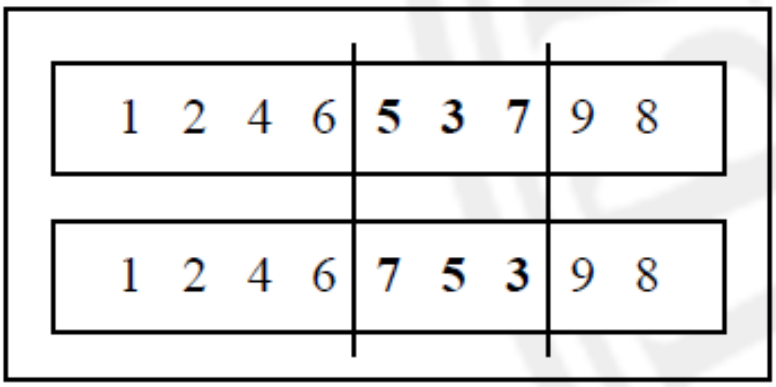

Figura 3.18 Mutación por permutación de un subconjunto (Pérez, 2010)

3.4.7. Sustitución

La forma en la que se aplica sustitución depende del proceso de reproducción. Al respecto, se establece la siguiente distinción de los algoritmos genéticos:

- *Algoritmos genéticos generacionales*: De la población inicial de N individuos se obtienen N padres que se recombinan para obtener N hijos. Estos N hijos forman parte, en su totalidad, de la siguiente generación de padres.
- *Algoritmos genéticos de creación continua*: En este caso, se eligen dos individuos para obtener dos hijos y se introducen en la población sustituyendo uno o dos individuos. Dicha sustitución puede ser:
    - Aleatoria, es decir, todos tienen la misma probabilidad.

- Seleccionando al peor o peores individuos.
- Mediante técnicas de clasificación u ordenación.

Existe una diferencia fundamental entre estos dos tipos de algoritmos expuestos y es que, al crear un buen individuo, para los algoritmos generacionales no estará disponible hasta la siguiente generación mientras que para los algoritmos de creación continua lo estará inmediatamente.

Para evitar que con la selección o los operadores genéticos se pierdan individuos con buenas características de la generación anterior, en los algoritmos genéticos generacionales se usa un proceso elitista que consiste en eliminar al peor individuo de la nueva generación y sustituirlo por el mejor individuo de la generación actual.

4. ALGORITMOS DE OPTIMIZACIÓN MULTIOBJETIVO

Los Algoritmos de Optimización Multiobjetivo (**AGMO**) pertenecen a los algoritmos genéticos introducidos en el apartado anterior y, dado que son los que se van a utilizar en este trabajo, este capítulo va a servir para profundizar sobre ellos.

El objetivo del trabajo consiste en la optimización de los parámetros de un algoritmo multiobjetivo en su utilización para sintonización de TMDs en una estructura siguiendo dos criterios: Estado Límite de Servicio y Estado Límite Último del edificio (en nuestro caso, de la maqueta).

Para la resolución de este tipo de problemas, es necesario utilizar herramientas de optimización multiobjetivo, tales como los AGMO.

En muchos casos, los objetivos de optimización establecidos por el problema son contradictorios, es decir, mejorar desde el punto de vista de un objetivo la solución provoca que se empeore desde el punto de vista del otro. Por lo tanto, la resolución de la mayoría de estos problemas se reduce, de manera muy genérica a dos etapas (Deb, 1999):

1) Encontrar soluciones óptimas de compromiso en un rango amplio de valores para los objetivos planteados.
2) Elegir una de las soluciones obtenidas utilizando información más "avanzada", es decir, mediante criterios más allá de los objetivos principales de la optimización.

A continuación, se van a explicar con más profundidad tanto la optimización multiobjetivo como los algoritmos empleados para ello.

4.1. Optimización Multiobjetivo

Un problema de optimización multiobjetivo (Bagchi, 1999) es aquel que tiene varias funciones objetivo que deben ser maximizadas o minimizadas (en general, optimizadas). Cada una de las funciones tiene asociada una serie de restricciones que debe ser satisfecha por las soluciones, de manera que el problema de optimización multiobjetivo tiene la siguiente forma general:

$$\begin{aligned} &Maximizar/Minimizar\ f_m(x), m = 1, 2, \dots, M; \\ &Sujeto\ a\ g_j(x) \geq 0, j = 1, 2, \dots, J; \\ &h_k(x) = 0, k = 1, 2, \dots, K; \\ &x_i^{(L)} \leq x_i \leq x_i^{(U)}, i = 1, 2, \dots, n; \end{aligned} \tag{4.1}$$

Una solución 'x' es un vector de n variables de decisión. La última línea corresponde a las variables frontera, que restringen cada variable de decisión x_i entre dos límites superior (U) e inferior (L). Estos límites constituyen un espacio de variables de decisión 'D'. Además, asociado al problema, existen J (desigualdades) y K (igualdades) restricciones.

Hay M funciones objetivo que deben ser optimizadas de forma simultánea para que la solución o el conjunto de soluciones obtenido sea válido para el problema, puesto que todas las funciones objetivo deben ser satisfechas.

4.1.1. Fundamentos de la Optimización Multiobjetivo

Para entender los principios de la optimización multiobjetivo, se va a usar un ejemplo gráfico, como son las escalas en los vuelos comerciales.

En ellos, se pueden distinguir dos objetivos claramente diferenciables:

1. Coste del vuelo
2. Comodidad del pasajero

Desde el punto de vista del coste del vuelo, las aerolíneas tienen en cuenta el volumen de pasajeros que se trasladan entre dos puntos. Cuando el volumen es muy elevado, compensa ofrecer un vuelo directo, mientras que, si el viaje entre dos ciudades es utilizado por poca gente, es más rentable realizar escalas en otras ciudades intermedias.

Por otra parte, para los pasajeros es más cómodo realizar un viaje con vuelo directo entre dos puntos puesto que el tiempo invertido en el viaje será menor. Sin embargo, el viaje directo será más caro.

Por lo tanto, entre estos dos objetivos es necesario encontrar una solución de compromiso que satisfaga a ambas partes de la mejor manera posible.

La conclusión es que para problemas multiobjetivo no existe una única solución óptima, sino que hay un conjunto de soluciones válidas que satisfacen todos los objetivos.

4.1.1.1. Soluciones Óptimas. Frente de Pareto

El frente de Pareto es una herramienta muy útil para la optimización de problemas de varios objetivos. En concreto, es muy ilustrativo cuando se trata de optimizar dos objetivos. Consiste en un conjunto de soluciones (todas ellas válidas) representadas por una función de aspecto hiperbólico en el que, en cada eje, está representado uno de los objetivos.

Al desplazarse por la función, se mejora la solución desde el punto de vista de un objetivo y se empeora desde el punto de vista del otro y viceversa (Deb, 1999), por lo que no existe una única solución óptima.

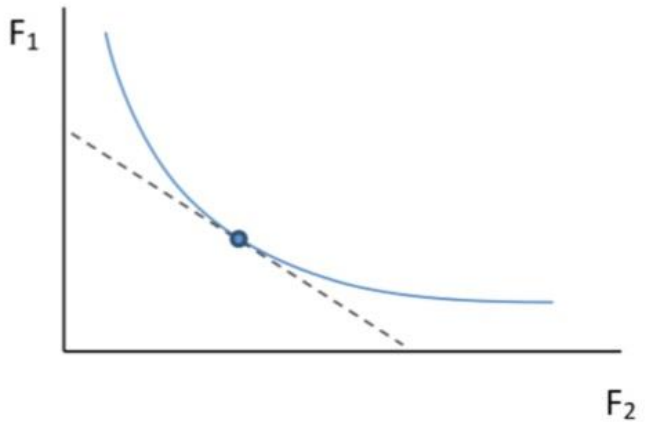

Figura 4.1 Frente de Pareto

En la figura 4.1 puede verse un ejemplo de función de Pareto. En ella está señalado la que podría ser la solución más adecuada al problema (solución de compromiso entre F_1 y F_2).

La única manera de mejorar las soluciones es desplazar el frente de Pareto hacia el origen (figura 4.2), optimizando el conjunto de soluciones (cuando se trata, como en la imagen siguiente, de un problema de minimización). Para ello se utilizan los Algoritmos Genéticos Multiobjetivo.

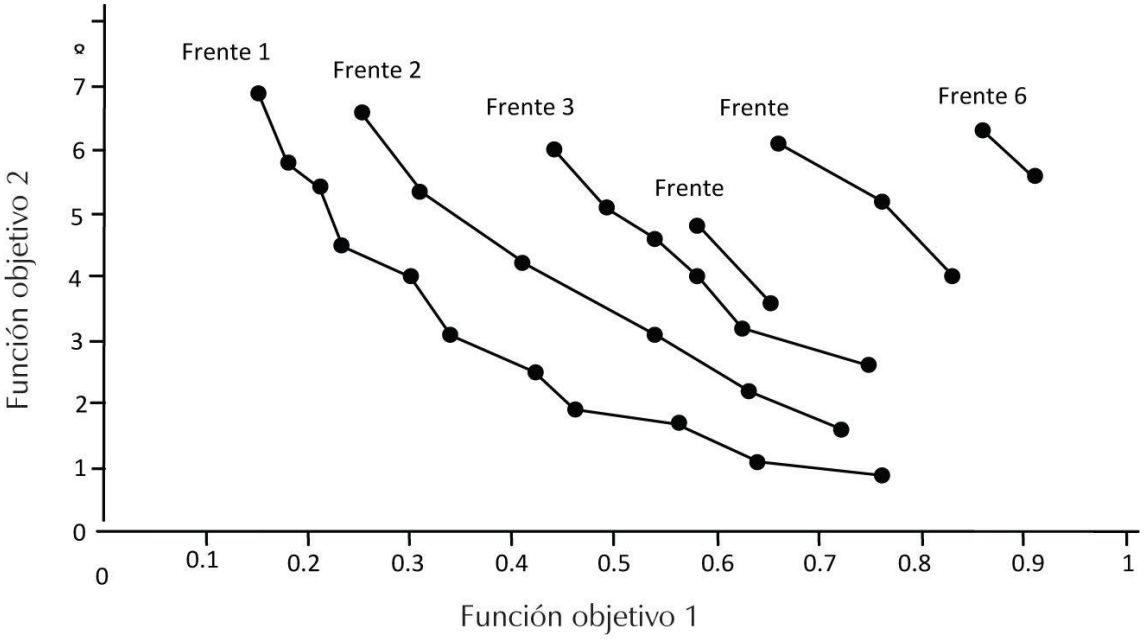

Figura 4.2 Optimización del Frente de Pareto

4.1.1.2. Objetivos en la Optimización Multiobjetivo

En el ejemplo anterior, se está tratando con un problema de dos objetivos, pero el método también es válido cuando hay más de dos funciones a optimizar.

Como se ha visto en el apartado anterior, se obtiene un conjunto de soluciones todas ellas válidas para el problema. Por lo tanto, cuando tenemos múltiples objetivos que, además, son contradictorios entre sí, discernir cuál es la mejor solución es muy complicado.

Por ello, cuando no se dispone de información adicional que pueda permitir realizar una elección de soluciones, todas las soluciones son igualmente importantes. Por esta razón, hay dos metas a cumplir en la optimización multiobjetivo:

a) Encontrar un conjunto de soluciones tan cercano como sea posible al frente de Pareto óptimo.

b) Que el conjunto de soluciones sea tan diverso como sea posible.

El primero de los objetivos es evidente cuando se trata de un problema de optimización, mientras que el segundo permite obtener un conjunto de soluciones lo suficientemente grande como para conseguir un conjunto de soluciones de compromiso al problema planteado cumpliendo todas las funciones objetivo.

4.1.1.3. Objetivos no conflictivos

Un aspecto a tener en cuenta es que el frente de Pareto es útil solo si las funciones objetivo del problema son conflictivas, puesto que, si no lo son, la solución óptima para uno de los objetivos coincidirá con la del resto de objetivos, reduciéndose el frente de Pareto a un punto.

4.1.2. Dominancia. Optimización de Pareto

Asumiendo que existen M funciones objetivo y para considerar tanto la maximización como la minimización, se usará el operador ◊ entre dos soluciones como i◊j para indicar que i es mejor solución que j o j◊i para indicar que j es mejor que i. Análogamente, dicho operador puede ser negado para indicar que la solución i no es mejor que la j (i~~◊~~j). En problemas de minimización ◊ corresponderá al operador '<' mientras que en problemas de maximización ◊ representará el operador '>'.

Se dice que la solución $x^{(1)}$ domina a la solución $x^{(2)}$ cuando las dos condiciones siguientes se satisfacen:

1. La solución $x^{(1)}$ no es peor que la solución $x^{(2)}$ en ninguno de los objetivos, es decir, $f_j(x^{(1)})$ ~~◊~~ $f_j(x^{(2)})$ para todo j = 1, 2, …, M.
2. La solución $x^{(1)}$ es estrictamente mejor que la solución $x^{(2)}$ en, al menos, un objetivo, o $f_{\bar{j}}(x^{(1)})$ ◊$f_{\bar{j}}(x^{(2)})$ para, al menos, un $\bar{j} \in \{1, 2, \ldots, M\}$.

Teniendo en cuenta lo anterior, se puede saber cuándo una solución domina a otra y, por lo tanto, descartar dichas soluciones.

Comparando pares de soluciones obtenidas obtendremos finalmente un conjunto de soluciones no dominadas entre sí siendo todas ellas válidas para el problema planteado.

4.1.3. Formación de nichos y especiación

Para terminar con el desarrollo teórico de los Algoritmos Genéticos Multiobjetivo, es importante conocer los conceptos de nicho y especiación.

En los Algoritmos Genéticos, la formación de nichos y la especiación son interesantes porque permiten obtener todas las soluciones óptimas (todos los picos) de las funciones consideradas en el problema, evitando así que el AG converja siempre hacia el mismo pico o solución del problema, sobre todo en funciones multiobjetivo.

(Deb y Goldberg, 1989) consideran dos formas de formación de nichos: agrupamiento (crowding) y compartición (sharing).

- Agrupamiento: Propuesto por (De Jong, 1975), consiste en crear diferentes nichos reemplazando miembros existentes en otro nicho de acuerdo a su similitud con otros miembros en una población superpuesta.
- Intercambio: Este método ocurre en la naturaleza cuando determinados recursos son compartidos dentro de un entorno. Cuando se habla de un Algoritmo Genético, el recurso que se comparte es la aptitud (fitness) usada como criterio de apareamiento

del AG. Para introducir el intercambio en el AG se rebaja la aptitud del miembro una cantidad proporcional al miembro más cercano de su entorno. De esta manera se reduce la tendencia de que un miembro con una aptitud muy alta domine al resto de miembros del mismo nicho durante la selección.

Un sistema de intercambio, propuesto por (Goldberg and Richardson, 1987) consiste en dividir la población en subpoblaciones de acuerdo a la similitud entre los individuos. Dicha similitud se controla comparando la distancia ($d(x_i, x_j)$) de los miembros entre sí con un parámetro denominado σ_{share}. La cantidad de material genético que se comparte entre dos miembros depende de la función de intercambio de la ley de poder $Sh(d(x_i, x_j))$ según la ecuación siguiente:

$$Sh\left(d(x_i, x_j)\right) = 1 - \left[\frac{d(x_i, x_j)}{\sigma_{share}}\right]^{\alpha} \; si \; d(x_i, x_j) < \sigma_{share} \qquad (4.2)$$
$$= 0 \; en \; cualquier \; otro \; caso$$

Donde α es el exponente de poder, seleccionado adecuadamente.

Para promover el comportamiento de formación de nichos, los individuos de cada nicho comparten su aptitud con sus vecinos. La forma en la que el individuo x_i comparte el fitness se determina teniendo en cuenta la suma de los valores de la función de intercambio aportados por los otros miembros de la población a x_i. Los miembros cercanos a x_i provocarán un alto grado de intercambio mientras que los lejanos causarán un grado bajo de intercambio.

El cálculo de la aptitud (fitness) actual del miembro del nicho se obtiene dividiendo la aptitud original del individuo entre la suma de las funciones de intercambio de dicho miembro (i) con cada miembro (j), con j = 1, …, n en una población de tamaño n (ecuación 4.3).

$$f_s(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} Sh\left(d(x_i, x_j)\right)} \qquad (4.3)$$

Este mecanismo anterior simula el intercambio de recursos limitados por organismos de un nicho en un ecosistema. La organización en nichos implica que los recursos deben ser compartidos, evitando el crecimiento incontrolado de individuos con una aptitud muy elevada.

Para resumir, la formación de nichos favorece que los individuos compartan los recursos siguiendo las bases de la especialización y reduciendo la competición entre especies por dichos recursos. Esto último impulsa la diversidad dentro del entorno y, por lo tanto, permite al algoritmo obtener todos los picos del problema (soluciones óptimas).

Por otra parte, Deb y Goldberg advirtieron que la especiación mejora el rendimiento de la búsqueda de soluciones del AG. Una vez que el mecanismo de intercambio agrupa conjuntos de soluciones en los picos de la función multiobjetivo, la recombinación entre miembros asentados en diferentes picos puede producir nuevos individuos que no pertenecen a ningún pico, lo cual provoca desperdiciar esfuerzos de cálculo, puesto que se trata de individuos con una aptitud baja que no sobrevivirán a la selección.

Para evitar esto, se introdujo el concepto de especiación, creando especies a partir de las agrupaciones de individuos en los picos de la función y restringiendo el apareamiento entre miembros de distintas especies.

La forma de realizar la especiación es prácticamente igual a la de formar nichos, utilizando una distancia $d(x_i, x_j)$ entre dos individuos y comparándola con un valor σ_{mating} para saber si son de la misma especie o no. Si se satisface la condición, los individuos se aparean y, si no es así, se prueba con otro individuo.

El uso de la especiación tiene como resultado la obtención con cierta rapidez de los picos de una función multiobjetivo.

En resumen, la formación de nichos mediante intercambio minimiza las opciones de que todas las soluciones obtenidas converjan a un único pico de la función multiobjetivo y la especiación reduce las posibilidades de formar miembros inadecuados por apareamientos indeseables, a la vez que mejora la velocidad a la que el AG encuentra soluciones aceptables.

4.2. Métodos Clásicos

En este apartado se van a explicar los métodos clásicos utilizados para enfrentarse a problemas de optimización multiobjetivo. A grandes rasgos, estos métodos clásicos (o algoritmos), pueden clasificarse en dos grupos principales (Cohon, 1985):

- Métodos de generación
- Métodos basados en preferencia

Los métodos de generación obtienen soluciones no dominadas para el problema y posteriormente se elige una de ellas. A priori no se utilizan conocimientos de importancia relativa entre los diferentes objetivos.

Por otro lado, los métodos basados en preferencia tienen en cuenta la información disponible acerca de las prioridades respecto a los distintos objetivos.

Otra posible clasificación es la que propusieron (Hwang y Masud, 1979) y, más tarde (Miettinen, 1999), afinando la propuesta de Cohon:

- Métodos sin preferencia: No utilizan información sobre la importancia relativa de los objetivos
- Métodos a posteriori: Usan las preferencias de cada objetivo e iterativamente obtienen un conjunto de soluciones en forma de frente de Pareto.
- Métodos a priori: Usan los conocimientos acerca de todos los objetivos y, habitualmente, obtienen una solución de Pareto óptima.
- Métodos interactivos: Los métodos interactivos van usando la información acerca de los objetivos de forma progresiva durante la optimización.

4.2.1. Método de Sumas Ponderadas

Este método consiste en ponderar cada objetivo en función de un peso suministrado por el usuario. De esta manera se aproxima el problema multiobjetivo por uno de único objetivo.

Para utilizar este método es necesario tener información acerca de la importancia relativa de los distintos objetivos para asignarles una ponderación adecuada y teniendo en cuenta el orden de magnitud de cada uno de los mismos.

Una vez se realiza lo anterior, el problema de optimización de único objetivo que se obtiene se describe de la siguiente manera:

$$\begin{aligned} Minimize\ F(x) &= \sum_{m=1}^{M} w_m f_m(x), \\ sujeto\ a\ g_j(x) &\geq 0, j = 1, 2, \dots, J; \\ h_k(x) &= 0, k = 1, 2, \dots, K; \\ x_i^{(L)} &\leq x_i \leq x_i^{(U)}, i = 1, 2, \dots, n. \end{aligned} \tag{4.4}$$

En la ecuación anterior, el peso del m-ésimo objetivo está representado por w_m (comprendido entre 0 y 1). Normalmente, la asignación de la ponderación se realiza de tal manera que la suma de todos los pesos sea la unidad.

A partir de la ecuación anterior se obtiene la solución del problema. Hay dos teoremas fundamentales asociados a la misma (Miettinen, 1999):

1) *La solución del problema representada por la ecuación 4.4 es el óptimo de Pareto si los pesos de todos los objetivos son positivos.*
2) *Si x^+ es una solución óptima de Pareto de un problema de optimización multiobjetivo convexo, entonces existe un vector **w** de pesos no nulos positivos de manera que x^+ es una solución al problema dada por la ecuación 4.4.*

La principal ventaja de este método es que es el más simple para resolver problemas de optimización multiobjetivo, puesto que garantiza encontrar soluciones en el frente de Pareto.

Sin embargo, el método presenta problemas cuando los objetivos a optimizar son duales (unos de maximización y otros de minimización), o cuando se aplica a problemas no lineales.

Otro aspecto a tener en cuenta es que el método está programado para encontrar una solución que satisfaga el criterio de optimización primario, pero es necesario realizar comprobaciones para ver si la solución encontrada es, efectivamente, un mínimo, por lo que es necesario invertir más tiempo de cálculo computacional.

En conclusión, se trata de un método sencillo de aplicar, pero tiene ciertas desventajas a la hora de encontrar las soluciones óptimas al problema en algunos casos.

4.2.2. Método de Restricción ε

Este método se basa en mantener tan sólo un objetivo y restringir el resto con un valor fijo determinado por el usuario. El problema se reduce a:

$$\begin{gathered} Minimizar\ f_{\mu}(x), \\ Sujeto\ a\ f_m(x) \leq \varepsilon_m, m = 1, 2, \dots, M\ y\ m\ \neq\ \mu; \\ g_j(x) \geq 0, j = 1, 2, \dots, J; \\ h_k(x) = 0, k = 1, 2, \dots, K; \\ x_i^{(L)}\ \leq x_i \leq x_i^{(U)}, i = 1, 2, \dots, n. \end{gathered} \tag{4.5}$$

El parámetro ε_m representa el límite superior de la función f_m. Es la restricción impuesta por el algoritmo para la resolución del problema. Se pueden realizar varios experimentos modificando los valores de la restricción y obtener así distintas soluciones al problema.

Este método permite obtener soluciones óptimas de Pareto en problemas no convexos, al contrario que el método estudiado anteriormente.

El siguiente teorema determina la utilidad del método de restricción ε:

- *La única solución del problema de restricción ε fijado en la ecuación 4.5 es una solución óptima de Pareto para cualquier vector de límites superiores dado:*

$$\varepsilon = (\varepsilon_1, \dots, \varepsilon_{\mu-1}, \varepsilon_{\mu+1}, \dots, \varepsilon_M)^T$$

Este método tiene como ventajas que obtiene soluciones óptimas de Pareto diferentes mediante la utilización de distintos valores de restricción, ε_m. Puede utilizarse tanto para problemas convexos como para no convexos.

En cuanto a la cantidad de información necesaria para este método, es similar al de suma ponderada.

Como desventaja, se puede destacar que la solución del problema obtenida depende del vector de restricciones elegido (ε). Al establecer unos límites a las funciones objetivo, no se están cumpliendo las condiciones originales establecidas por el problema, por lo que la solución obtenida es válida pero no es la única. Además, puede haber casos en los que la elección de las restricciones sea tal que no haya soluciones factibles al problema.

4.2.3. Método Métricos Ponderados

Se basa en combinar varios objetivos en uno sólo. Para ello se utilizan métricas ponderadas como l_p o l_∞. Para pesos no negativos, la distancia l_p medida de cualquier solución x procedente de la solución ideal z^* puede ser minimizada como sigue:

$$\begin{aligned} Minimizar\ l_p(x) &= \left(\sum_{m=1}^{M} w_m |f_m(x) - {z_m}^*|^p\right)^{1/p}, \\ Sujeto\ a\ g_j(x) &\geq 0, j = 1, 2, \dots, J; \\ h_k(x) &= 0, k = 1, 2, \dots, K; \\ {x_i}^{(L)} &\leq x_i \leq {x_i}^{(U)}, i = 1, 2, \dots, n. \end{aligned} \tag{4.6}$$

El parámetro p puede tomar cualquier valor entre 1 e infinito. Cuando p = 1, el problema resultante es equivalente al de sumas ponderadas. Cuando p = 2, una distancia euclídea ponderada de cualquier punto en el espacio de objetivos desde el punto ideal es minimizada.

Si se utiliza un valor de p elevado, el problema se reduce a minimizar la desviación más grande $|f_m(x) - z_m^+|$. Este problema particular se denomina problema Tchebycheff ponderado, siendo la función a minimizar la siguiente:

$$Minimizar\ l_\infty(x) = \ max_{m=1}^{M} w_m |f_m(x) - {z_m}^*| \tag{4.7}$$

Sujeto a las mismas condiciones que la ecuación 4.6.

Cuando se utilizan p = 1 o 2, las soluciones que se obtienen son óptimas, pero limitadas. Sin embargo, al usar el valor infinito, se puede obtener cualquier solución óptima del frente de Pareto. Para este último caso (problema Tchebycheff ponderado), existe un teorema especial:

- *Siendo x^+ una solución óptima de Pareto. Existe un vector positivo de ponderación tal que x^+ es solución del problema Tchebycheff ponderado mostrado en la ecuación 4.7, donde el punto de referencia es el vector utópico z^{++}.*

El método es muy útil para encontrar soluciones óptimas. En concreto, en el caso del problema Tchebycheff ponderado, se pueden encontrar todas las soluciones óptimas de Pareto. Sin embargo, en este método hay que tener en cuenta si los objetivos tienen distintos órdenes de magnitud para que funcione bien. El método también requiere la solución ideal z^x. Para ello, todos los objetivos deben ser optimizados independientemente antes de optimizar la métrica l_p.

4.2.4. Método Benson

Este procedimiento es similar al de las métricas ponderadas, excepto que la solución de referencia se selecciona como una solución óptima factible pero no de Pareto. Una solución z^0 es seleccionada aleatoriamente de la región posible de soluciones. Posteriormente, se calcula la distancia no negativa entre la solución z^0 y cada objetivo y la suma de todas ellas es maximizada:

$$\begin{gathered} Maximizar \sum_{m=1}^{M} max\left(0, \left(z_m^0 - f_m(x)\right)\right), \\ Sujeto\ a\ f_m(x) \leq z_m^0, m = 1, 2, \dots, M; \\ g_j(x) \geq 0, j = 1, 2, \dots, J; \\ h_k(x) \geq 0, k = 1, 2, \dots, K; \\ x_i{}^{(L)} \leq x_i \leq x_i{}^{(U)}, i = 1, 2, \dots, n. \end{gathered} \tag{4.8}$$

La maximización de la función anterior es similar a encontrar un hipercubo con el máximo perímetro. Como el frente de Pareto se encuentra en el extremo de la región factible de soluciones, la solución óptima del problema de optimización es un miembro del frente de Pareto.

El método tiene la ventaja de que la normalización de todas las diferencias puede realizarse previamente a realizar la suma. Para la obtención de las soluciones óptimas de Pareto, las diferencias pueden ser ponderadas antes de la suma y, después de esto, cambiando el vector de ponderaciones, se pueden obtener diferentes soluciones óptimas de Pareto. Adicionalmente, si la elección de la solución inicial z^0 es adecuada, el método puede ser aplicable a problemas no convexos.

El problema de optimización establecido necesita un número adicional de restricciones en la región que domine la solución elegida z^0. La función objetivo obtenida no es diferenciable, por lo que los métodos de resolución basados en el gradiente tienen muchas dificultades para obtener la solución.

4.2.5. Método de la Función de Valor

En este método, el usuario proporciona una función de valor U = $\mathbb{R}^M \rightarrow \mathbb{R}$ que relaciona los M objetivos. Esta función de valor debe ser válida en todo el espacio de posibles soluciones. El problema se convierte en una maximización de la función de valor:

$$\begin{gathered} Maximizar\ U\big(f(x)\big), \\ Sujeto\ a\ g_j(x) \geq 0, j = 1, 2, \dots, J; \\ h_k(x) = 0, k = 1, 2, \dots, K; \\ {x_i}^{(L)} \leq x_i \leq {x_i}^{(U)}, i = 1, 2, \dots, n. \end{gathered} \tag{4.9}$$

En la ecuación 4.9, f(x) es un vector que contiene todas las funciones objetivo del problema multiobjetivo. Tal y como está definido el problema si U(f(i)) > U(f(j)), la solución i será mejor que la j, puesto que queremos maximizar U. (Rosenthal, 1985) expone que la función U debe ser fuertemente decreciente para poder ser usada en problemas multiobjetivo. Eso significa que la preferencia de una solución debe aumentar si uno de los objetivos se reduce mientras se mantiene otro con el mismo valor. A raíz de esto, (Miettinen, 1999) propuso el siguiente teorema:

- *Siendo la función de valor* U = $\mathbb{R}^M \rightarrow \mathbb{R}$ *fuertemente decreciente. Si U alcanza su máximo en* f^+*. Entonces,* f^+ *es una solución óptima de Pareto.*

Hay que destacar que este método permite obtener una sola solución para cada conjunto de parámetros elegido. Variando los mismos se encuentran distintas soluciones.

Las principales ventajas de este método es que es sencillo e ideal cuando se tiene suficiente información sobre la función de valor. Este método es utilizado en problemas multiobjetivo para obtener un conjunto discreto de soluciones, aunque puede ser aplicado en espacios de búsqueda continuos.

Por otra parte, como se ha indicado anteriormente, la solución obtenida depende de la función de valor elegida. Por ello, necesita que el usuario utilice una función aplicable globalmente sobre el espacio completo de búsqueda.

4.2.6. Métodos de Programación de Objetivos

La principal idea de la programación de objetivos es encontrar una solución que logre una meta predefinida para una o varias funciones objetivo. Si no es posible conseguir lo anterior, el siguiente paso es encontrar soluciones que maximicen la derivada de la meta marcada. SI, por el contrario, se consigue encontrar directamente una solución a la meta predefinida, la tarea del método es identificar dicha solución particular.

Para un problema de único objetivo, el problema se resuelve con la ecuación 4.10:

$$\left.\begin{array}{c} Objetivo\ (f(x) = t) \\ x \in S \end{array}\right\} \tag{4.10}$$

Donde S es el espacio de posibles soluciones. Si la meta 't' es menor que el valor óptimo del objetivo, no existe ninguna solución factible que satisfaga el objetivo. Por lo tanto, la finalidad del método sería encontrar la solución que minimice la desviación (d) entre el resultado y la meta 't'. La solución del problema es x^+ y la sobreestimación es $d = f(x^+) - t$. Análogamente, si la meta 't' establecida es mayor que el máximo factible de la función objetivo, la tarea del método es encontrar el valor x que maximiza la función: $f(x) = f_{max}$. Por último, si la meta 't' se encuentra en el rango $[f(x^+), f_{max}]$, la solución del problema de programación de objetivos es exactamente igual a 't', aunque no sea la solución óptima.

Aunque se ha descrito el problema de único objetivo, este método es muy interesante para aplicarlo en problemas multiobjetivo.

En el caso de la ecuación 4.10, la meta planteada es una igualdad, pero hay cuatro tipos diferentes de criterios de meta:

1) Menor o igual: $f(x) \leq t$
2) Mayor o igual: $f(x) \geq t$
3) Igual: $f(x) = t$
4) Perteneciente a un rango: $f(x) \in [t^l, t^u]$

En cualquier caso, para resolver un problema de programación de objetivos, es necesario convertir cada meta en, al menos, una restricción de igualdad, y el objetivo es minimizar todas las desviaciones, normalmente identificadas por las letras p y n. Hay varios métodos de programación de objetivos, los cuales difieren en la forma de minimizar las desviaciones.

4.2.7. Métodos Interactivos

El último de los métodos clásicos que se va a explicar es el interactivo. Son métodos que no requieren muchos conocimientos a priori. Cuando se encuentra una solución óptima de Pareto, su localización e interacciones son analizadas.

El principal aspecto a tener en cuenta en estos métodos es que, de vez en cuando, durante la optimización es necesario aportar algo de información sobre la dirección de búsqueda, el vector de ponderaciones, los puntos de referencia y otros factores.

Algunos de los métodos interactivos más populares son:

- Método de compensación del sustituto interactivo importante (en inglés, **I**nteractive **S**urrogate **W**orth **T**rade-off method, ISWT) (Chankong y Haimes, 1983).
- Método de etapa (Benayoun et al., 1971).
- Método del punto de referencia (Wierzbicki, 1980).
- Método de suposición (Buchanan, 1997).
- Aproximación del sistema de optimización multiobjetivo interactivo no diferenciable basado en haces (NIMBUS, en inglés) (Miettinen y Mäkelä, 1995).
- Aproximación del haz de luz (Jaszkiewicz y Slowinsky, 1999).

4.3. Algoritmos Genéticos Multiobjetivo

Los métodos clásicos explicados con anterioridad son de gran utilidad en la resolución de problemas en los que la complejidad no es muy elevada. Además, no se pueden utilizar cuando se trata de problemas en los que es necesario cumplir más de un objetivo. Por esta razón, se han desarrollado los Algoritmos Genéticos Multiobjetivo.

En el siguiente apartado se indican algunas modificaciones a los algoritmos genéticos simples cuando es necesario resolver problemas multiobjetivo para que sean más efectivos en la búsqueda de soluciones (Deb, 2001).

4.3.1. Función de Optimización

Al tratar con problemas multiobjetivo, es necesario encontrar múltiples soluciones óptimas. Debido a esto, es complicado que un algoritmo encuentre de manera sencilla el óptimo global del problema. Por esta razón, existen varias modificaciones sobre los algoritmos genéticos básicos para encontrar múltiples soluciones óptimas de Pareto.

4.3.1.1. Diversidad mediante Mutación

En los problemas multiobjetivo, es posible que un algoritmo encuentre soluciones cercanas a las óptimas en generaciones tempranas. Se hace necesario mantener estas soluciones "buenas" en las sucesivas generaciones, por lo que se hace necesario tener un operador de preservación de diversidad.

El operador mutación se usa a menudo en los algoritmos genéticos. Junto con la selección y el cruce, puede ayudar a encontrar diferentes soluciones óptimas y, además, permite mantener soluciones válidas a lo largo de muchas generaciones.

El problema del operador mutación es que puede tener un efecto constructivo en el problema, pero también destructivo. Puede generar una mejor solución o empeorarla. Comprobar si el efecto de la mutación ha sido positivo o negativo supone invertir tiempo, por lo que, en los algoritmos genéticos, la mutación se aplica con una probabilidad muy pequeña.

4.3.1.2. Preselección

Se trata de otro método para mantener diversidad en un algoritmo genético. Cuando se genera descendencia a partir de dos soluciones padre, se compara la aptitud (fitness) del hijo con la de los padres. Si la aptitud del hijo es superior a la del peor de los padres, lo sustituye. Si el hijo tiene una aptitud similar a la de ambos padres, sustituir a uno de los padres permite coexistir varias soluciones diferentes en la población.

Esta operación se realiza con todas las parejas de padres. De esta manera se consiguen obtener múltiples soluciones importantes dentro de la población.

4.3.1.3. Modelo de Agrupamiento (Crowding)

También es un método de conservación de diversidad. Como su nombre indica, consiste en agrupar soluciones en cualquier lugar del espacio de búsqueda rechazado, promoviendo la diversidad necesaria para mantener múltiples soluciones óptimas.

DeJong propone un modelo de agrupamiento basado en usar un concepto de población superpuesta y una estrategia de agrupamiento. En su algoritmo genético, solo una proporción G (llamada generación hueco, *gap generation*) de la población puede reproducirse en cada generación. Cuando un descendiente va a ser introducido en la población superpuesta, se seleccionan CF soluciones (factor de agrupamiento o *crowding factor*) de forma aleatoria. El descendiente es comparado con esas CF soluciones y la solución más parecida a la del descendiente es reemplazada.

Esta forma de realizar la sustitución de soluciones permite mantener la diversidad a lo largo de las generaciones. Por lo tanto, la principal contribución de este modelo es que se mantienen múltiples soluciones óptimas reemplazando una solución por otra similar en la población.

4.3.1.4. Modelo de la Función de Intercambio

(Goldberg y Richardson, 1987) sugirieron un concepto revolucionario consistente en degradar la aptitud de soluciones similares en lugar de reemplazarla. Se trata de un modelo muy utilizado para la resolución de problemas de optimización multiobjetivo. A continuación, se va a describir el modelo con más detalle.

Suponiendo que queremos encontrar Q soluciones óptimas a un problema multiobjetivo. La única información disponible es una población finita de N individuos. Se asume que q << N, por lo que cada óptimo puede considerarse dentro de un nicho independiente. Para poder encontrar las soluciones óptimas, la población N debe tener suficientes individuos representativos de cada nicho.

Para cada solución óptima se determina un número de soluciones representativas. Si en un óptimo hay más individuos que los esperados, es necesario degradar la aptitud de manera que, en una competición global, el operador de selección no considere todas las soluciones (se evita priorizar unas soluciones sobre otras). Análogamente, si en un óptimo hay menos individuos de los esperados, dicho óptimo debe ser enfatizado por el operador selección. Lo anterior se indica en la ecuación 4.11:

$$\frac{f_1}{m_1} = \frac{f_2}{m_2} = \cdots = \frac{f_q}{m_q} \tag{4.17}$$

En la que f_i es una zona óptima representada por m_i soluciones: $m \propto f$

Se define la función de la aptitud intercambiada (*shared fitness function*) como:

$$f_i' = \frac{f_i}{m_i} \tag{4.18}$$

La ecuación 4.12 implica que todos los óptimos (ya sean locales o globales) tendrán la misma aptitud. Esto permite que se promocione cada óptimo por igual y se mantengan las múltiples soluciones óptimas en las generaciones sucesivas.

El modelo expuesto anteriormente tiene un problema debido a que es necesario tener mucha información sobre el problema para saber si una solución pertenece, efectivamente, a un óptimo real. Para ello, (Goldberg y Richardson, 1987) usaron una función de intercambio (ecuación 4.13) que permitía estimar el número de soluciones que pertenecen a cada óptimo.

$$Sh(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{share}}\right)^{\alpha}, si\ d \leq \sigma_{share} \\ \\ 0, en\ cualquier\ otro\ caso \end{cases} \tag{4.19}$$

El parámetro d es la distancia entre dos soluciones. La función toma valores entre 0 y 1. El valor 1 significa que la solución es idéntica y el valor 0 indica que dos soluciones están demasiado alejadas y, por lo tanto, no hay ningún efecto de intercambio entre ellas.

A partir de esta función, es posible calcular una estimación de la extensión de un agrupamiento alrededor de una solución. Para ello, se calcula la función de intercambio de una solución con respecto al resto de la población y se realiza el sumatorio de todos los valores de la siguiente manera:

$$nc_i = \sum_{j=1}^{N} Sh(d_{ij}) \tag{4.20}$$

El valor de nc_i es siempre mayor o igual a uno, puesto que hay que tener en cuenta la contribución de la solución sobre sí misma, Sh(0.0) = 1. Por último, se calcula la aptitud de intercambio como $f'_i = f_i/nc_i$. La aptitud de intercambio será igual para aquellos óptimos con mayor número de soluciones y para aquellos con menor número de soluciones como se describió anteriormente.

Una vez se han calculado todas las aptitudes de intercambio, un método de selección proporcional se us para determinar la región de apareamiento (los operadores cruce y mutación se utilizan de manera habitual).

La utilización de la aproximación de la función de intercambio conlleva ciertos problemas. En primer lugar, la elección de un σ_{share} adecuado. Para ello, es necesario determinar el número de óptimos (q). Si q es mayor que el número de óptimos del problema en la región de búsqueda, la función de intercambio va a intentar crear más nichos de los que el problema puede tener, lo que provoca que se encuentre soluciones no óptimas. Por otra parte, si q es menor que el número de óptimos existentes en la región de búsqueda del problema, algunos óptimos no serán encontrados. Para evitar este problema, es aconsejable utilizar dos 'q' distintas.

También es importante considerar el tamaño de la población necesario para que la función de intercambio sea capaz de encontrar los óptimos de forma independiente. Se propone usar, al menos, la población mínima necesaria representada por N = k·q, donde k es una constante cuyo producto con q permite obtener la población mínima indicada.

A lo largo de las generaciones, el AG puede ser mejorado teniendo en cuenta que la suma de cada nicho proporciona una idea del número de soluciones alrededor de un óptimo, permitiendo actualizar σ_{share} a un valor, cada vez, más adecuado. Se mejor así el rendimiento del algoritmo.

Para terminar, otra consideración a tener en cuenta es que pueden tenerse soluciones no óptimas cuando éstas no están rodeadas por muchas soluciones, puesto que, como se ha explicado, la función de intercambio favorecerá a estas soluciones para evitar la pérdida de diversidad.

4.3.1.5. Algoritmos Genéticos Ecológicos

La propuesta de (Davidor, 1991) es un AG ecológico basado en la formación de nichos y especies. Para conseguirlo, sitúa los miembros de la población en una malla bidimensional y cada miembro está rodeado por otros ocho miembros. Los operadores genéticos se aplican sobre subpoblaciones de nueve individuos (el individuo seleccionado junto a los ocho que le rodean) para crear descendencia. La nueva generación creada compite con los padres y, si su aptitud es mejor, reemplaza a uno de ellos.

Lo normal es que la descendencia tenga una aptitud similar a la de sus padres, por lo que se espera que los miembros de una subpoblación se mantengan cerca, formando islas de diferentes soluciones óptimas.

De esta manera se demuestra que el método es útil para su utilización en problemas de optimización multiobjetivo, puesto que el algoritmo permite mantener picos locales durante muchas generaciones.

4.3.2. Algoritmos Genéticos Multiobjetivo

Una vez se ha realizado una introducción a los problemas de optimización multiobjetivo y a las bases de resolución, se va a proceder a explicar algunos de los algoritmos genéticos más utilizados en la resolución de este tipo de problemas.

Se trata de algoritmos con un nivel de desarrollo elevado capaces de encontrar soluciones óptimas a problemas con una complejidad alta. Para ello, están programados de manera que mantienen conjuntos de soluciones no dominadas usando un operador conservador de nichos como los descritos en la sección 4.3.1. Debido a esto, se pueden mantener múltiples soluciones buenas en la población y, después de varias generaciones, el proceso converge encontrando soluciones óptimas del frente Pareto.

Es importante recalcar que el interés de encontrar soluciones sobre el frente de Pareto radica en que cada una de dichas soluciones es óptima en el problema a resolver. La elección posterior de una solución del conjunto obtenido requiere de información más avanzada acerca de los objetivos planteados para el problema.

4.3.2.1. Non-Dominated Sorting Genetic Algorithm (NSGA)

Es un algoritmo que usa estrategias elitistas de preservación de individuos y un mecanismo explícito para mantener la diversidad.

Parte de una población de padres P_t para generar la descendencia Q_t (ambas poblaciones de N miembros). En lugar de encontrar las soluciones no dominadas de la descendencia, este algoritmo utiliza la población R_t (formada por “N individuos), formada por las dos poblaciones P_t y Q_t. Esto requiere un mayor esfuerzo que seleccionar las soluciones no dominadas de Q_t sencillamente. Sin embargo, se obtiene un conjunto de soluciones no dominadas entre padres e hijos.

Una vez se han ordenado todas las soluciones (padres e hijos), la nueva población se rellena a partir de soluciones de diferentes frentes, eligiendo siempre las mejores soluciones. Como la nueva población debe tener N elementos, las “peores” soluciones son eliminadas.

En este algoritmo, sin embargo, existe el problema de que hay regiones del frente de soluciones que no pueden considerarse debido a que no hay espacio disponible en la nueva población. Para evitarlo, en lugar de descartar arbitrariamente ciertas soluciones, se utiliza una estrategia de nicho para elegir los miembros del último frente, basado en la región menos poblada de dicho frente.

Esta estrategia no es importante en las primeras etapas del algoritmo. Sin embargo, a medida que se avanza, la población R_t contiene cada vez más soluciones no dominadas procedentes del mejor frente, hasta el punto en el que más de N soluciones contenidas en R_t (se recuerda que tiene un tamaño 2N) pertenezcan a este frente. La estrategia de nicho garantiza que se elija un conjunto de soluciones variado dentro del conjunto contenido en R_t. Cuando la población converge al frente óptimo de Pareto, este algoritmo asegura una mejor proliferación de las soluciones.

El algoritmo se inicia generando una población aleatoria P_0. Esta población se ordena en niveles de no-dominancia y a cada solución se le asigna una aptitud (o “fitness”) en función de

dicha no-dominancia (siendo 1 el mejor nivel). Se tiene que minimizar ese nivel. Para ello, se utiliza la selección por torneo, la recombinación y la mutación para crear una descendencia Q_0.

El procedimiento del NSGA es el siguiente:

1) Combinar los padres y la descendencia y crear $R_t = P_t \cup Q_t$. Ordenar R_t e identificar diferentes frentes: F_i, i = 1, 2, …, etc.
2) Establecer la nueva población $P_{t+1} = \emptyset$. Establecer un contador i = 1.

   Mientras $|P_{t+1}|+|F_i| < N$, aplicar $P_{t+1} = P_{t+1} \cup F_i$ e i = i + 1.
3) Aplicar el procedimiento de ordenación por agrupamiento ($F_i, <_c$) e incluir la máxima variedad de soluciones ($N - |P_{t+1}|$) usando los valores de la distancia de agrupamiento en el orden de F_i en P_{t+1}.
4) Crear la población Q_{t+1} a partir de la P_{t+1} usando la selección por torneo, el cruce y la mutación.

El operador comparador de agrupaciones ($<_c$) compara dos soluciones e indica la ganadora partiendo de que cada solución i tiene dos atributos:

1. Un rango de no-dominancia r_i en la población.
2. Una distancia local de agrupamiento (d_i) en la población.

La distancia local de agrupamiento de una solución es la medida del espacio de búsqueda alrededor de dicha solución que no está ocupado por otra solución. Se define el operador de selección por torneo de agrupamiento de la siguiente manera:

- *Operador de Selección por Torneo de Agrupamiento: Una solución i gana el troneo contra otra solución j si cualquiera de las siguientes condiciones es verdadera:*
  - *Si la solución i tiene mejor rango, es decir: $r_i < r_j$.*
  - *Si tienen el mismo rango, pero la solución i tiene una distancia local de agrupamiento mejor que la solución j, es decir: $r_i = r_j$ y $d_i > d_j$.*

La primera condición asegura que la solución elegida está en un frente mejor de no-dominancia. La segunda condición resuelve el empate de dos soluciones que pertenecen al mismo frente mediante la distancia de agrupamiento. La solución que se encuentra en una región menos poblada gana el torneo. La distancia de agrupamiento es un indicador que permite saber la densidad de soluciones que rodea a una en particular.

Una ventaja de la utilización de esta distancia, es que no es necesario usar un parámetro de nicho (como σ_{share}) en etapas posteriores del algoritmo, puesto que ya está considerado.

El algoritmo, no obstante, presenta una desventaja importante. Cuando el operador de comparación de agrupamiento se utiliza para restringir el tamaño de la población, el algoritmo pierde si propiedad de convergencia. En el momento en el que el primer frente de no-dominancia excede el tamaño de la población, el algoritmo hace que soluciones más cercanas al óptimo de Pareto tengan que dejar su lugar a otras soluciones no dominadas que no pertenecen al óptimo de Pareto. Se genera así un conjunto de soluciones que, en las etapas sucesivas, puede resultar en un ciclo que genere soluciones óptimas y no óptimas, en lugar de converger en un conjunto de soluciones óptimas de Pareto.

4.3.2.2. Strength Pareto Evolutionary Algorithm (SPEA)

Se trata de un algoritmo evolutivo elitista propuesto por (Zitzler y Thiele, 1998) conocido como SPEA. El algoritmo introduce el elitismo manteniendo explícitamente una población externa $\bar{P}$. Esta población almacena un número fijo de soluciones no dominadas encontradas hasta el comienzo de la simulación.

En cada generación, las nuevas soluciones encontradas son comparadas con las existentes en la población externa $\bar{P}$ y las soluciones no dominadas resultantes son conservadas. Además de preservar la élite, el algoritmo usa dicha élite en los operadores genéticos para guiar la búsqueda hacia las mejores regiones del espacio.

El algoritmo comienza con una población aleatoria inicial P_0 de tamaño N y una población externa vacía $\overline{P_0}$ con una capacidad máxima $\bar{N}$. En cada generación, las mejores soluciones no dominadas de la población P_t son copiadas en la población externa $\bar{P}_t$. De esta manera, las soluciones de la población externa que sean dominadas por nuevas soluciones encontradas son eliminadas. Así se almacenan las mejores soluciones.

Sin embargo, para evitar que la población externa se llene de soluciones no dominadas, el tamaño de la población externa se limita ($\bar{N}$). Mientras la población externa no esté llena, se siguen almacenando las mejores soluciones. No obstante, una vez está llena, el criterio para mantener soluciones es el de conservar aquellas soluciones que pertenezcan a un frente menos poblado, manteniendo así la diversidad de soluciones.

Una vez se han introducido en la población exterior las nuevas soluciones obtenidas, el algoritmo aplica los operadores genéticos para obtener una nueva población. El primer paso es asignar la aptitud ("fitness") a cada solución en la población obtenida, así como a las soluciones contenidas en la población exterior.

La aptitud de los individuos de la población exterior (S_i) es proporcional al número (n_i) de miembros de la población actual que una solución exterior i domina:

$$S_i = \frac{n_i}{N+1} \tag{4.21}$$

Cuantas más soluciones domina una solución externa, mejor será su aptitud. El denominador garantiza que la aptitud nunca sea uno o superior.

Por otra parte, la aptitud de un miembro de la población actual j se asigna como uno más el sumatorio de la aptitud de todos los miembros de la población externa a las que domina j:

$$F_j = 1 + \sum_{i \in \bar{P}_t \wedge i \preccurlyeq j} S_i \tag{4.22}$$

El método provoca que la aptitud de un miembro de la población actual sea siempre mayor la aptitud de cualquier miembro de la población exterior. Este método sugiere que las soluciones con menor aptitud son las mejores.

Con los valores de la aptitud asignados, se aplica la selección por torneo binario a la población conjunta $\bar{P}_t \; U \; P_t$ para elegir las soluciones con menor aptitud. Por lo tanto, el procedimiento favorece las élites exteriores durante el torneo. De forma usual, se aplican los operadores cruce y mutación para crear una nueva población P_{t+1} de tamaño N.

El desarrollo paso a paso de una iteración del algoritmo es el siguiente, partiendo de: $\bar{P}_t = \emptyset$.

1) Encontrar el mejor conjunto no dominado $F_1(P_t)$ de P_t. Copiar las soluciones en $\bar{P}_t$ o aplicar $\bar{P}_t = \bar{P}_t \; U \; F_1(P_t)$.
2) Encontrar las mejores soluciones no dominadas $F_1(\bar{P}_t)$ de la población modificada $\bar{P}_t$ y borrar todas las soluciones dominadas, o aplicar $\bar{P}_t = F_1(\bar{P}_t)$.
3) Si $|\bar{P}_t| > \bar{N}$, usar una técnica de agrupamiento (clustering) para reducir el tamaño a $\bar{N}$. En caso contrario, mantener $\bar{P}_t$ sin cambios. La población resultante es la población externa de la siguiente generación $\bar{P}_{t+1}$.
4) Asignar la aptitud a cada solución de la élite $i \in \bar{P}_{t+1}$ usando la ecuación 4.20. Posteriormente, asignar la aptitud a cada individuo de la población $j \in P_t$usando la ecuación 4.21.
5) Aplicar la selección por torneo binario con los valores de la aptitud (en sentido de minimización), el cruce y la mutación para crear una nueva población $\bar{P}_{t+1}$ de tamaño N de la población combinada $\bar{P}_{t+1} \; U \; P_t$ de tamaño $\bar{N} + N$.

En cuanto al algoritmo de agrupamiento (*clustering algorithm*), éste se encarga de reducir el tamaño de la población externa. Para ello, considera cada solución en un grupo diferente. Posteriormente, se calcula la distancia de agrupamiento entre cada para de grupos, definida como la distancia euclídea media de todos los pares de soluciones ($i \in C_1$, $j \in C_2$):

$$d_{12} = \frac{1}{|C_1||C_2|} \sum_{i \in C_1, j \in C_2} d(i,j) \tag{4.23}$$

Una vez que todas las distancias han sido calculadas, los dos grupos (clusters) más cercanos forman un grupo más grande. A continuación, se recalculan las distancias y se repite el proceso hasta que el número de grupos en la población externa se reduce a $\bar{N}$.

En cada grupo, la solución con una distancia media menor a otras soluciones del mismo grupo se mantiene y el resto son eliminadas.

Como principal ventaja de este método, destaca que, una vez que se encuentra una solución óptima de Pareto, se almacena en la población exterior y solo es eliminada si se encuentra otra solución mejor.

Por otra parte, introducir un nuevo parámetro como el tamaño de la población exterior $\bar{N}$ supone una complicación. Es debido a que debe haber un balance entre N y $\bar{N}$ para que el algoritmo funcione satisfactoriamente. Se propone una ratio de 1:4 entre el tamaño de la población exterior y el tamaño de población del algoritmo.

4.3.2.3. Strength Pareto Evolutionary Algorithm II (SPEA2)

El algoritmo SPEA2 es una evolución del anterior en el que se han introducido algunos cambios respecto al algoritmo SPEA (Ziztler et al., 2001).

El algoritmo SPEA usa una población inicial y un archivo vacío en el que almacena soluciones no dominadas exclusivamente. Dicho archivo se actualiza en cada iteración, eliminándose los individuos duplicados o dominados se eliminan del mismo. Cuando el archivo está lleno de soluciones no dominadas, la técnica de selección de soluciones mantiene la diversidad del frente óptimo.

Una vez se tiene el archivo con los individuos que corresponda, se calcula el fitness tanto de los individuos de la población como de los del archivo tal y como se explicó en el apartado anterior.

A continuación, se realiza la selección de soluciones mediante torneo binario entre los individuos de la población y del archivo y se procede con la siguiente iteración (Antón, 2012).

Resumiendo, el algoritmo SPEA presenta los principales problemas siguientes:

- Asignación del fitness: Los individuos que son dominados por los mismos miembros tienen el mismo valor de aptitud, lo cual hace muy difícil discernir que individuos son mejores sin saber si se dominan entre sí.
- Densidad: Cuando hay muchos individuos que no se dominan entre sí, no se tiene suficiente información para saber cuáles de ellos debe seleccionar el algoritmo. El cálculo de la densidad de soluciones en la región de búsqueda permite mantener la diversidad y guiar la búsqueda más adecuadamente.
- Truncamiento del archivo: En el algoritmo SPEA, la técnica de agrupamiento reduce el número de soluciones del archivo para que no se exceda el límite de soluciones establecido. Sin embargo, se eliminan las soluciones más alejadas, lo cual impide obtener un frente de soluciones distribuido en toda la región de búsqueda.

El algoritmo SPEA2 mejora a su predecesor en estos tres aspectos, obteniéndose un conjunto de soluciones de mayor calidad.

Fitness y Densidad

En este apartado se va a explicar la forma en la que el algoritmo SPEA2 realiza la asignación de la aptitud y calcula la densidad para la selección posterior de soluciones.

Para evitar que pueda haber varios individuos con el mismo valor de fitness, SPEA2 tiene en cuenta tanto los individuos a los que domina cada solución como los individuos por los que es dominado.

A cada miembro del archivo $\bar{P}_t$ y de la población P_t se le asigna un valor de importancia S(i) que representa el número de soluciones a las que domina:

$$S(i) = |\{j | j \in P_t \cup \bar{P}_t \wedge i \succ j\}| \tag{4.24}$$

Con estos valores, se obtiene un fitness inicial R(i), como el sumatorio de los valores S(j) de los individuos que dominan a i, sean del archivo o de la población:

$$R(i) = \sum_{j \in P_t \cup \bar{P}_t, j \succ i} S(j) \tag{4.25}$$

Evidentemente, cuanto menor es el fitness, mejor, puesto que se tratará de un individuo dominado por muy pocos miembros.

Este valor de fitness se completa utilizando la densidad. De esta manera, se puede llegar a discernir qué solución es mejor que otra con igual aptitud en base a la acumulación de soluciones que haya en una determinada región del espacio de búsqueda.

Para individuo, se calculan las distancias a todos los individuos en el archivo y la población y se almacenan y ordenan de forma creciente para compararlos con una distancia referencial identificada por σ_i^r.

En SPEA2 la r utilizada se calcula como: $r = \sqrt{N + \bar{N}}$ con N el tamaño de la población y $\bar{N}$ el tamaño del archivo. La densidad de cada individuo se obtiene con la ecuación 4.25:

$$D(i) = \frac{1}{\sigma_i^r + 2} \tag{4.26}$$

La densidad siempre será mayor que cero y menor que 1.

Finalmente, el valor del fitness se calcula sumando los valores de R(i) y D(i):

$$F(i) = R(i) + D(i) \tag{4.27}$$

Actualización del archivo de soluciones

El método de truncamiento utilizado en el SPEA2 evita que se eliminen las soluciones más extremas. El primer paso es copiar en el archivo de la siguiente generación las soluciones con un fitness menor que 1 (estas son las soluciones no dominadas):

$$\bar{P}_{t+1} = \{i | i \in P_t \cup \bar{P}_t \wedge F(i) < 1\} \tag{4.28}$$

Si el número de soluciones no dominadas coincide con el tamaño del archivo $\bar{N}$, la operación se da por finalizada. Por el contrario, puede haber dos posibilidades:

1. El número de soluciones no dominadas es menor que $\bar{N}$. En este caso, el archivo se completa con aquellas soluciones dominadas que tengan mejor fitness.

2. El archivo no puede albergar tantas soluciones no dominadas, por lo que es necesario recurrir al método de truncamiento. Se basa en elegir el individuo que tiene la distancia mínima a otro individuo y, en caso de empate, se acude a las siguientes distancias para decidirlo. Al final, se debe cumplir $|\bar{P}_{t+1}| = \bar{N}$. Matemáticamente, el método de truncamiento es:

$$\begin{aligned} i \leq_d j \leftrightarrow \quad & \forall\, 0 < r < |\bar{P}_{t+1}|: \quad \sigma_i^r = \sigma_j^r \ \vee \\ & \exists\, 0 < r < |\bar{P}_{t+1}|: \quad \left[\left(\forall\, 0 < l < r: \sigma_i^l = \sigma_j^l\right) \wedge \sigma_i^r = \sigma_j^r\right] \end{aligned} \tag{4.29}$$

En la siguiente figura se muestra el método de truncamiento en un archivo de tamaño $\bar{N} = 5$. A la izquierda el conjunto total de soluciones no dominadas y a la derecha las soluciones eliminadas.

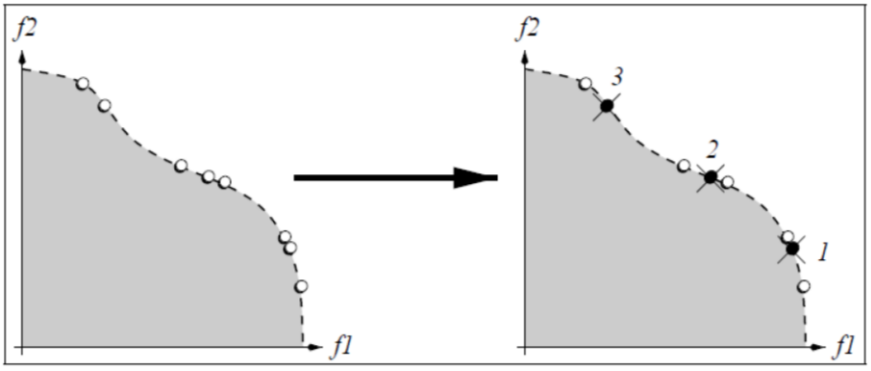

Figura 4.3 Método de truncamiento

Algoritmo SPEA2

A continuación, se muestra el algoritmo general del SPEA2 de forma esquemática:

1. Generar la población inicial P_0
2. Crear un archivo vacío $\bar{P}_0 = \emptyset$
3. $t = 0$
4. **While** $t < T$ (criterio genérico) **do**
    a. Calcular los valores de fitness de los individuos de P_t y $\bar{P}_t$
    b. Copiar todos los individuos no dominados en P_t y $\bar{P}_t$ a $\bar{P}_{t+1}$
    c. **If** $|\bar{P}_{t+1}| > \bar{N}$ **then**
        i. Reducir $\bar{P}_{t+1}$ con el método de truncamiento
    d. **else if** $|\bar{P}_{t+1}| < \bar{N}$ **then**
        i. Completar $\bar{P}_{t+1}$ con individuos dominados en P_t y $\bar{P}_t$
    e. **end if**
    f. Aplicar el operador de selección por torneo binario con reemplazamiento en $\bar{P}_{t+1}$ hasta completar el conjunto de individuos que se reproducirán
    g. Aplicar los operadores cruce y mutación y establecer en P_{t+1} la población resultante $t = t + 1$
5. **end while**
6. Devolver $\bar{P}_t$

5. MÉTODOS DE COMPARACIÓN DE FRONTERAS DE PARETO

Mediante los algoritmos genéticos se van a realizar distintos experimentos para la obtención de resultados del problema de optimización de TMD's en edificios.

Dichos experimentos van a realizarse modificando distintos parámetros para ver cómo afectan a los resultados y, posteriormente, se va a estudiar el impacto de dichos cambios sobre la frontera de Pareto.

Para poder analizar los cambios en la frontera de Pareto, es necesario utilizar métodos de comparación a través de los cuales se pueda discernir qué conjunto de soluciones es mejor.

En cuanto a los métodos de comparación, hay una clasificación fundamental según el tipo de estudio que realizan sobre los conjuntos de soluciones y que se indican a continuación (Deb, 2001).

5.1. Métodos que evalúan la cercanía a la frontera de Pareto óptima

5.1.1. Error Ratio

Este método simplemente analiza cuántos elementos del conjunto de soluciones Q no pertenecen al conjunto de soluciones óptimo, P (Veldhuizen, 1999). Se define como:

$$ER = \frac{\sum_{i=1}^{|Q|} e_i}{|Q|} \tag{5.1}$$

Donde $e_i = 1$ si $i \notin P$ y $e_i = 0$ si $i \in P$.

Cuanto menor sea el valor de ER, mejor convergencia tendrá Q al frente de Pareto óptimo. Para que este método sea útil, se requiere que el conjunto P de soluciones tenga una gran cantidad de soluciones para evitar que, si un punto de Q pertenece a P, pero en el conjunto P dicho punto no existe, el método considere que la solución en cuestión no pertenece a P.

5.1.2. Set Coverage Metric

Este método es similar al anterior, pero permite obtener una idea de la dispersión entre dos conjuntos de soluciones, puesto que, dados dos conjuntos A y B el método calcula la proporción de soluciones de B que son dominadas por el conjunto A (C(A,B)) o viceversa (C(B,A)) (Ziztler, 1999):

$$C(A,B) = \frac{|\{b \in B | \exists a \in A : a \preccurlyeq b\}|}{|B|} \tag{5.2}$$

Si se obtiene C(A,B) = 1, todos los miembros de B están dominados por A y si se obtiene 0, ningún elemento estará dominado.

Se trata de una operación que no es simétrica (es decir, no se cumple necesariamente que: C(A,B) = 1 – C(B,A)), por lo que hay que calcular ambos valores.

A través de este indicador podemos discernir qué conjunto de soluciones es mejor teniendo en cuenta que si C(A,B) < C(B,A), el conjunto de soluciones B será mejor que el A, puesto que habrá menos soluciones de B dominadas por A, que soluciones de A dominadas por B.

5.1.3. Generational Distance

En lugar de averiguar si una solución del conjunto Q pertenece al frente de Pareto óptimo (P), este método calcula la distancia media de las soluciones de Q al conjunto P (Veldhuizen, 1999), de la siguiente manera:

$$GD = \frac{\left(\sum_{i=1}^{|Q|} d_i^p\right)^{\frac{1}{p}}}{|Q|} \tag{5.3}$$

Con p = 2, la distancia calculada es la euclídea entre la solución i ∈ Q y el individuo más cercano de P:

$$d_i = min_{k=1}^{|P|} \sqrt{\sum_{m=1}^{M} \left(f_m^{(i)} - f_m^{(k)}\right)^2} \tag{5.4}$$

Donde $f_m^{(k)}$ es el k-ésimo valor de la m-ésima función objetivo del conjunto P.

Este método tiene dificultades para mostrar la verdadera distancia cuando hay una gran distancia entre los elementos de Q. También es recomendable, en este sentido, que el conjunto P tenga un número grande de individuos.

5.2. Métodos que determinan la diversidad a lo largo del conjunto de soluciones

5.2.1. Spacing

Este indicador, propuesto por (Schott, 1995), calcula la distancia entre soluciones consecutivas del conjunto no dominado obtenido:

$$S = \sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}\left(d_i - \bar{d}\right)^2} \tag{5.5}$$

Donde: $d_i = \min_{k \in Q \wedge k \neq i} \sum_{m=1}^{M} \left| f_m^i - f_m^k \right|$ y $\bar{d}$ es el valor medio calculado como: $\bar{d} = \sum_{i=1}^{|Q|} \frac{d_i}{|Q|}$.

Este método calcula las desviaciones estándar de los diferentes valores, d_i. Cuanto más uniforme sea la distribución de las soluciones del conjunto Q, menor será el valor de S y, por lo tanto, mejor será el conjunto de soluciones desde el punto de vista de la diversidad.

El problema de este método es que es necesario comprobar las distancias entre cada solución i con el resto de soluciones para encontrar el menor valor, lo cual requiere un gran esfuerzo computacional.

5.2.2. Spread

Para evitar el problema del método anterior, (Deb et al., 2000) sugirieron este nuevo indicador:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} \left| d_i - \bar{d} \right|}{\sum_{m=1}^{M} d_m^e + |Q-1|\bar{d}} \tag{5.6}$$

En esta ecuación, d_i puede ser cualquier distancia entre soluciones vecinas y $\bar{d}$ es la media de las anteriores distancias. Puede utilizarse, por ejemplo, la distancia euclídea para este indicador.

El parámetro d_m^e es la distancia entre las soluciones extremas de los conjuntos P y Q correspondientes al m-ésimo objetivo.

Este indicador tiene un valor nulo cuando la distribución es ideal, para lo que es necesario que se den dos situaciones:

1. Todos los valores d_j^e son nulos.
2. Todas las distancias d_i son idénticas a la media, $\bar{d}$.

La primera condición indica que el conjunto de soluciones obtenido (Q) contiene las soluciones extremas del conjunto de Pareto óptimo. La segunda condición significa que la distribución entre las soluciones es uniforme e igual a la distancia media.

El valor del indicador estará comprendido entre 0 y 1, siendo mejor cuanto menor sea.

5.2.3. Maximum Spread

Para el cálculo de este indicador se obtiene la longitud de la diagonal de un "hipercubo" formado por los valores extremos de las funciones objetivo del problema en el conjunto no dominado (Ziztler, 1999):

$$D = \sqrt{\sum_{m=1}^{M} \left(max_{i=1}^{|Q|} f_m^i - min_{i=1}^{|Q|} f_m^i \right)^2} \tag{5.7}$$

Para tener una versión normalizada de dicha distancia, la ecuación puede modificarse obteniéndose lo siguiente:

$$\overline{D} = \sqrt{\frac{1}{M} \sum_{m=1}^{M} \left(\frac{max_{i=1}^{|Q|} f_m^i - min_{i=1}^{|Q|} f_m^i}{F_m^{max} - F_m^{min}} \right)^2} \tag{5.8}$$

Para esta última ecuación, F_m^{max} y F_m^{min} son el máximo y el mínimo valor del m-ésimo objetivo en el conjunto óptimo de Pareto elegido.

El método da una idea general del conjunto de soluciones, pero no permite calcular la distribución exacta de las soluciones intermedias.

5.2.4. Chi-Square-Like Deviation Measure (Deb, 2001)

Este indicador permite evaluar la capacidad de los algoritmos de optimización multiobjetivo. El método requiere la utilización de un parámetro ε para determinar el número de soluciones n_i que cada solución óptima de Pareto tiene como vecinas (Deb, 1989).

El indicador se calcula con la siguiente ecuación:

$$\iota = \sqrt{\sum_{i=1}^{|P|+1} \left(\frac{n_i - \bar{n}_i}{\sigma_i} \right)^2} \tag{5.9}$$

Como na hay ninguna preferencia entre las soluciones óptimas de Pareto, se suele considerar una distribución uniforme entre las mismas, de manera que $\bar{n}_i = |Q|/|P|$, es decir, para cada solución óptima se considera que va tener, idealmente, $\bar{n}_i$ soluciones a su alrededor.

Por otro lado, se calcula: $\sigma_i^2 = \bar{n}_i(1 - \bar{n}_i/|Q|)\ para\ i = 1, \dots, |P|$, mientras que para el índice $i = |P| + 1$, se obtiene como: $\sigma_{|P|+1}^2 = \sum_{i=1}^{|P|} \sigma_i^2 = |Q|(1 - 1/|P|)$.

Finalmente, hay que tener en cuenta que $\bar{n}_{|P|+1} = 0$, puesto que se espera que no haya soluciones alejadas de las soluciones óptimas.

Un algoritmo será tanto mejor cuanto menor sea el valor de este indicador, puesto que eso significa que las soluciones estarán cerca de la frontera óptima.

Este método no es útil cuando las soluciones están muy alejadas del frente de Pareto óptimo.

5.3. Métodos que analizan tanto la cercanía como la diversidad

5.3.1. Hypervolume

Este indicador calcula el volumen cubierto por los miembros de Q en problemas en los que todos los objetivos deben ser minimizados. Para ello, se construye un "hipercubo" v_i para cada solución i ∈ Q con un punto de referencia W (Veldhuizen, 1999; Ziztler y Thiele, 1998).

La elección del punto de referencia es arbitraria y puede obtenerse simplemente con un vector con valores de las funciones objetivos muy alejados del origen (solución lejana al óptimo).

El cálculo se realiza con la siguiente ecuación:

$$HV = volume\left(\cup_{i=1}^{|Q|} v_i\right) \tag{5.10}$$

El método depende de la escala de los objetivos, de manera que, si se tienen funciones de distinto orden de magnitud, una mejora en una función de mayor orden de magnitud reducirá mucho más el valor HV que una mejora en una función de menor orden de magnitud. Para evitar este problema, lo más adecuado es normalizar el valor de las funciones objetivo.

Otra manera de hacerlo es utilizar el frente de Pareto óptimo (si se dispusiera de él) para obtener:

$$HVR = \frac{HV(Q)}{HV(P)} \tag{5.11}$$

Cuanto más cercano a 1 es el valor de HVR, mejor será el algoritmo.

Teniendo en cuenta las características de los diferentes indicadores y la experimentación que se va a realizar, se van a utilizar cuatro de los ocho criterios mostrados anteriormente:

- Set Coverage Metric
- Spacing
- Maximum Spread
- Hypervolume

Con estas cuatro métricas, seremos capaces de discernir qué frontera de Pareto es mejor en la comparación que se hará en la experimentación.

6. EXPERIMENTACIÓN

La experimentación, como ya se ha adelantado en el capítulo anterior, va a consistir en la modificación de los parámetros de entrada del algoritmo y el estudio posterior de los resultados para ver qué valores son mejores.

Dicha experimentación se va a realizar sobre tres problemas distintos:

1. Edificio de dos plantas en el que se va a instalar un TMD (2P-1TMD).
2. Edificio de dos plantas en el que se va a instalar dos TMD (2P-2TMD).
3. Edificio de 5 plantas en el que se va a instalar un TMD (5P-1TMD).

Por otra parte, el estudio que se va a realizar consistirá en tres experimentos referidos a la modificación de tres parámetros con el objetivo de ver con qué valores se obtienen mejores resultados.

Los tres experimentos tienen unos datos de entrada similares, que se detallan a continuación:

- Número de variables: dependerá de si se utiliza un TMD (3 variables) o dos (8 variables).
- Valor máximo de las variables: límite superior establecido para las variables del TMD (masa, frecuencia y factor de amortiguamiento crítico) y, en el caso de tener dos TMD, planta más elevada en el que pueden estar situados (en este caso sería 2).
- Valor mínimo de las variables: límite inferior de las variables del TMD y, si procede, planta más baja en la que pueden estar situados los TMD (1 para este problema).
- Tipo de variable: 0 si es continua, 1 si es discreta y 2 si es fija.
- Tamaño de población.
- Número de evaluaciones de cada ejecución.
- Número de ejecuciones.
- Probabilidad de cruce.
- Probabilidad de mutación.
- Cruce 1: BLX.
- Cruce 3: SBX.
- Radio de nicho: puede tener dos valores:
    - 0 – Si no se permite que haya diferentes copias de una misma solución en la población.
    - -1 – Algoritmo SPEA2 tradicional.

Para los experimentos se va a estudiar lo siguiente:

1. Qué radio de nicho es mejor: 0 ó -1.
2. Qué tamaño de población es mejor: 100, 200 ó 500.
3. Qué cruce es mejor: cruce 1 (BLX) o cruce 3 (SBX).

6.1. Edificio 2P-1TMD

En primer lugar, se va a realizar la experimentación sobre un edificio de dos plantas en el que se va a instalar un TMD.

6.1.1. Experimento 1

Para este primer experimento se van a fijar todos los valores y se va a modificar tan solo el radio utilizando los valores 0 y -1. Los datos de entrada son los siguientes:

Número de variables		3
Valores máximos de las variables	Masa	0.16
	Frecuencia	1000
	Factor de amortiguamiento crítico	0.5
Valores mínimos de las variables	Masa	0.01
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tipo de variable	Masa	0
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tamaño de población		200
Número de evaluaciones		300000
Número de ejecuciones		5
Probabilidad de cruce		90
Probabilidad de mutación		10
Cruce	1: BLX	1
	2: PNX	0
	3: SBX	0
Radio de nicho		0 ó -1

Tabla 6.1 Variables 2P-1TMD del experimento 1

Todos los valores se mantienen fijos y se actúa sobre el valor del radio. Se realiza la experimentación con ambos valores y se analizan los resultados.

Dado que en cada experimento se realizan 5 ejecuciones, el programa obtiene 5 conjuntos de soluciones. Para obtener la frontera de Pareto, hay que hallar tantas soluciones no dominadas como el tamaño de población fijado (en este caso 200) utilizando todos los valores obtenidos. Para ello se tiene un archivo programado en Matlab que se encarga de hacerlo. De esta manera, se consiguen sendos conjuntos de soluciones, de 200 individuos cada uno. Este procedimiento es igual para todos los experimentos que se van a realizar.

Una vez se ha dado el paso anterior, se tienen las dos fronteras de Pareto correspondientes una a cada radio (0 y -1) y se puede proceder a realizar la comparación de las mismas.

En las siguientes imágenes, puede verse la representación gráfica de las fronteras de Pareto de los radios 0 y -1 por separado y superpuestas:

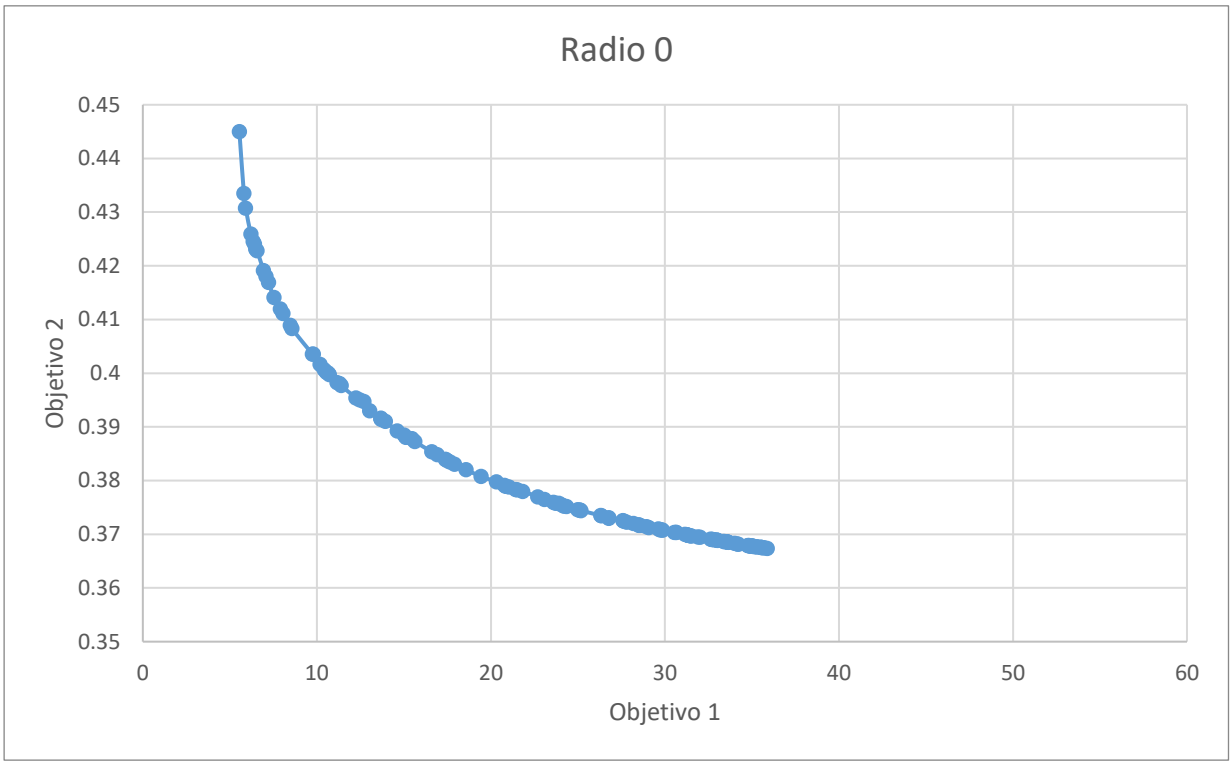

Figura 6.1 Frontera de Pareto de Radio 0 2P-1TMD

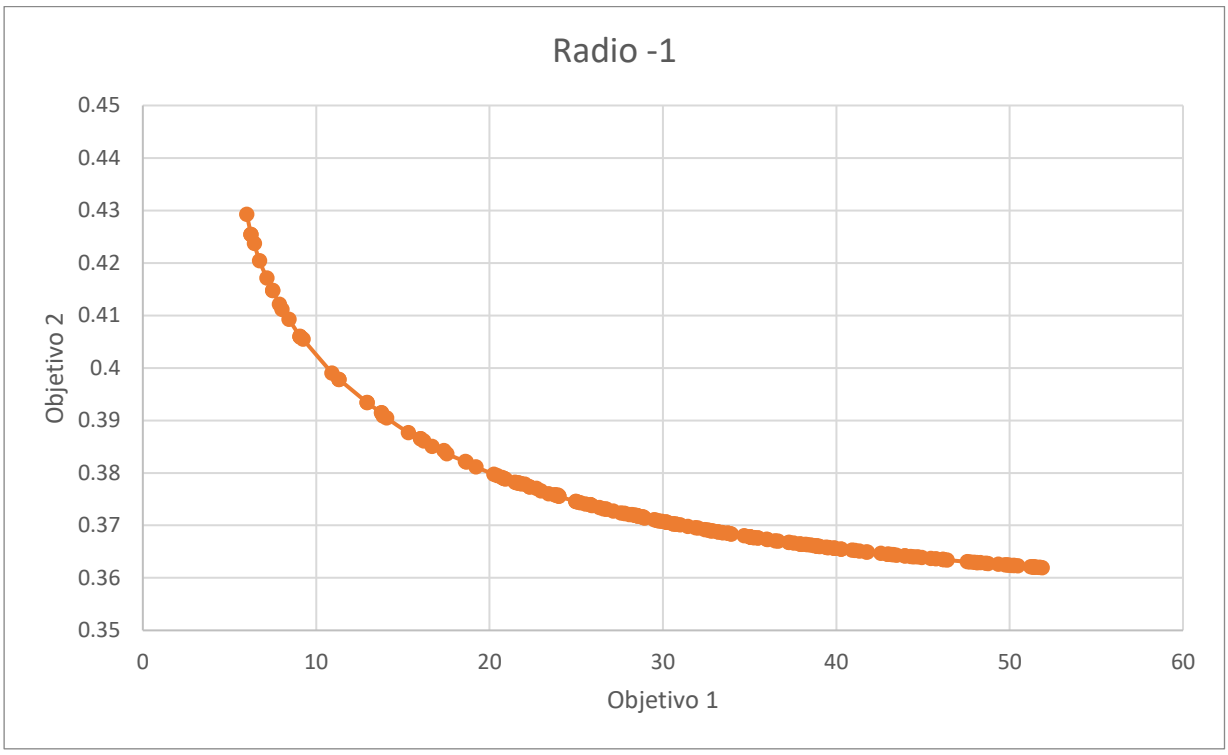

Figura 6.2 Frontera de Pareto de Radio -1 2P-1TMD

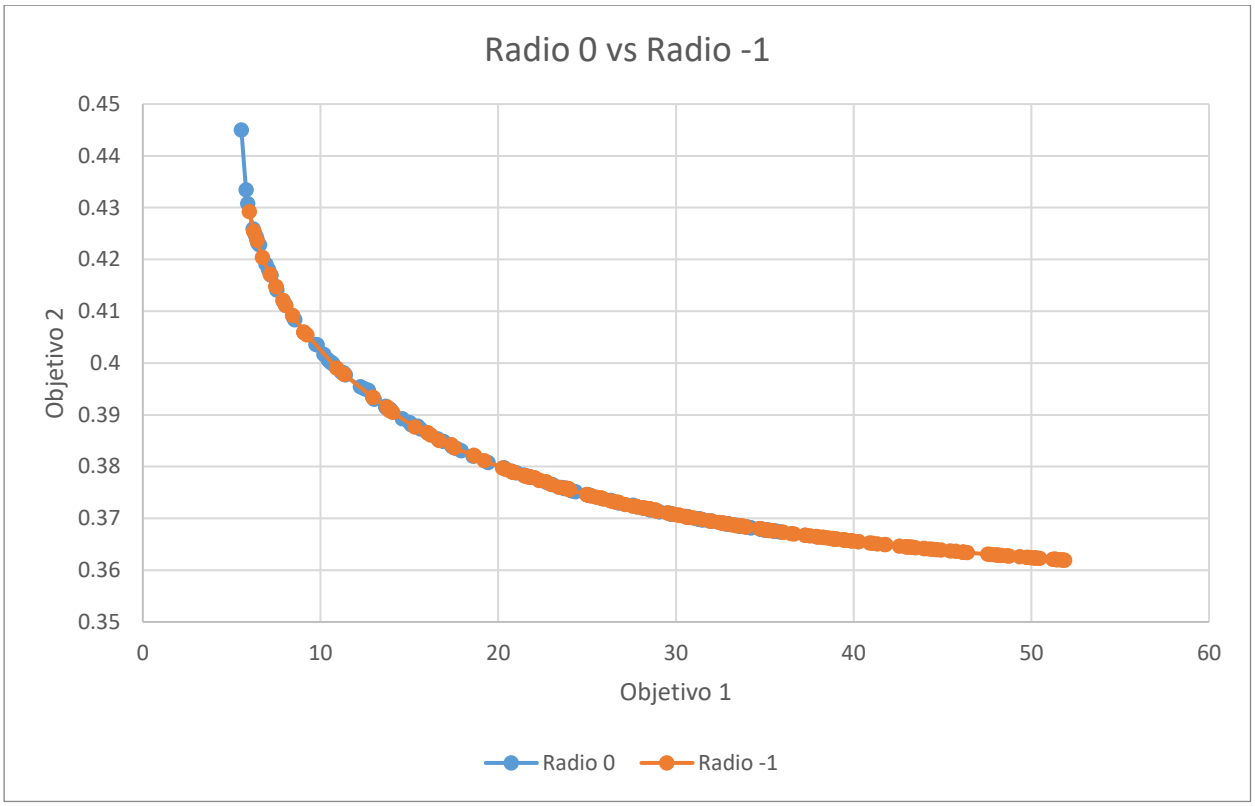

Figura 6.3 Comparación de radios 2P-1TMD

Atendiendo a las imágenes anteriores, no se puede discernir a simple vista qué radio es mejor, puesto que, en la comparación de ambas fronteras, las diferencias son mínimas. Por esta razón, se utilizan los métodos de comparación estudiados en el capítulo anterior.

Los resultados de los cuatro indicadores son los siguientes:

Set Coverage Metric	Radio 0 sobre Radio -1	0.000
	Radio -1 sobre Radio 0	0.010

Spacing	Radio 0	0.267470668213585
	Radio -1	0.117871404350600

Hypervolume	Radio 0	107.19507992041500
	Radio -1	98.27130392939260

Maximum Spread	Radio 0	49.81532414896400
	Radio -1	45.87944626577020

Teniendo en cuenta los criterios, se observa que el radio -1 es mejor en dos de ellos y el radio 0 en los otros dos.

El primer indicador estudia qué porcentaje de las soluciones de un conjunto es dominado por las soluciones del otro. Ambos valores son muy pequeños, lo que significa que los dos conjuntos son muy cercanos el uno al otro. En cualquier caso, el conjunto de radio -1 es menos dominado por el conjunto 0.

El segundo criterio estudia la uniformidad de la distribución de soluciones, es decir, las distancias entre los puntos de la frontera de Pareto. Cuanto más pequeña sea, mejor. En este caso, también es mejor el radio -1, aunque ambos valores son pequeños.

La tercera métrica se centra en el "hipervolumen" del espacio de soluciones que cubre cada conjunto, siendo mejor el radio 0.

Finalmente, el último indicador analiza la diversidad de la frontera de Pareto, es decir, la extensión de la misma a lo largo del espacio de soluciones.

Dado que el primer indicador no aporta demasiada información, que para el segundo indicador las diferencias no son excesivas y que los dos últimos criterios dan una idea de cuál de las dos distribuciones cubre una cantidad mayor del espacio de soluciones, se concluye que el mejor radio es el 0.

6.1.2. Experimento 2

Con el radio 0 como solución del primer experimento, se puede proceder con el siguiente estudio. En este caso, los datos de entrada son:

Número de variables		3
Valores máximos de las variables	Masa	0.16
	Frecuencia	1000
	Factor de amortiguamiento crítico	0.5
Valores mínimos de las variables	Masa	0.01
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tipo de variable	Masa	0
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tamaño de población		100, 200 ó 500
Número de evaluaciones		300000
Número de ejecuciones		5
Probabilidad de cruce		90
Probabilidad de mutación		10
Cruce	1: BLX	1
	2: PNX	0
	3: SBX	0
Radio de nicho		0

Tabla 6.2 Variables 2P-1TMD del experimento 2

Ahora se fijan todos los parámetros (con radio de nicho 0) y se modifica el tamaño de población con tres valores: 100, 200 y 500. El procedimiento a seguir es análogo al anterior: se obtienen 5 ejecuciones para cada tamaño de población y, para cada uno de los tamaños se halla un conjunto de soluciones no dominado que se utilizará para el análisis de los resultados.

Igual que en el primer experimento, se presentan los resultados por separado y superpuestos:

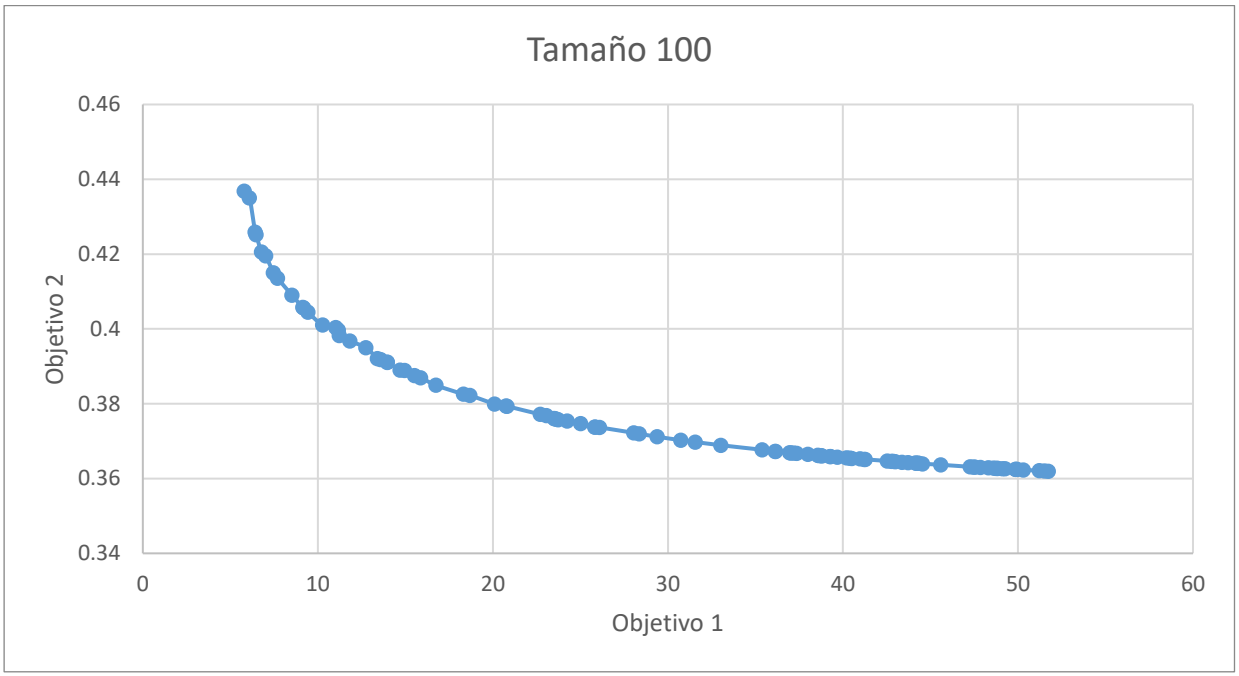

Figura 6.4 Frontera de Pareto de Tamaño 100 2P-1TMD

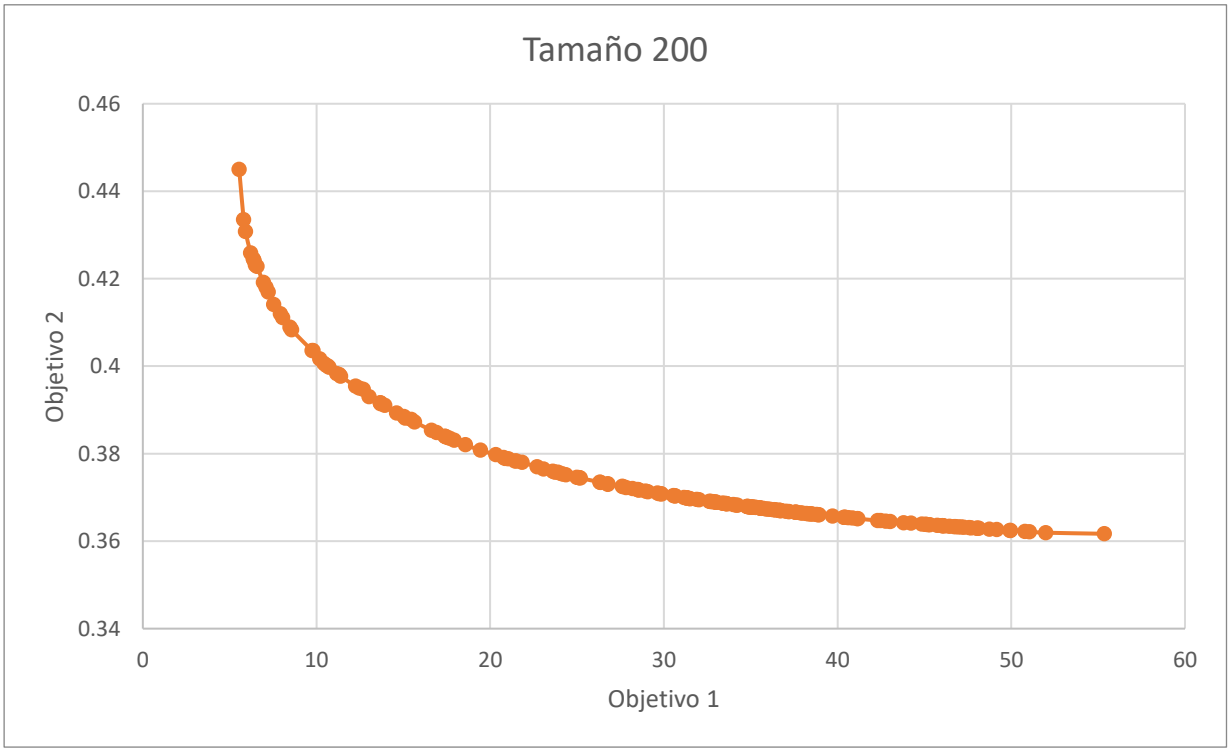

Figura 6.5 Frontera de Pareto de Tamaño 200 2P-1TMD

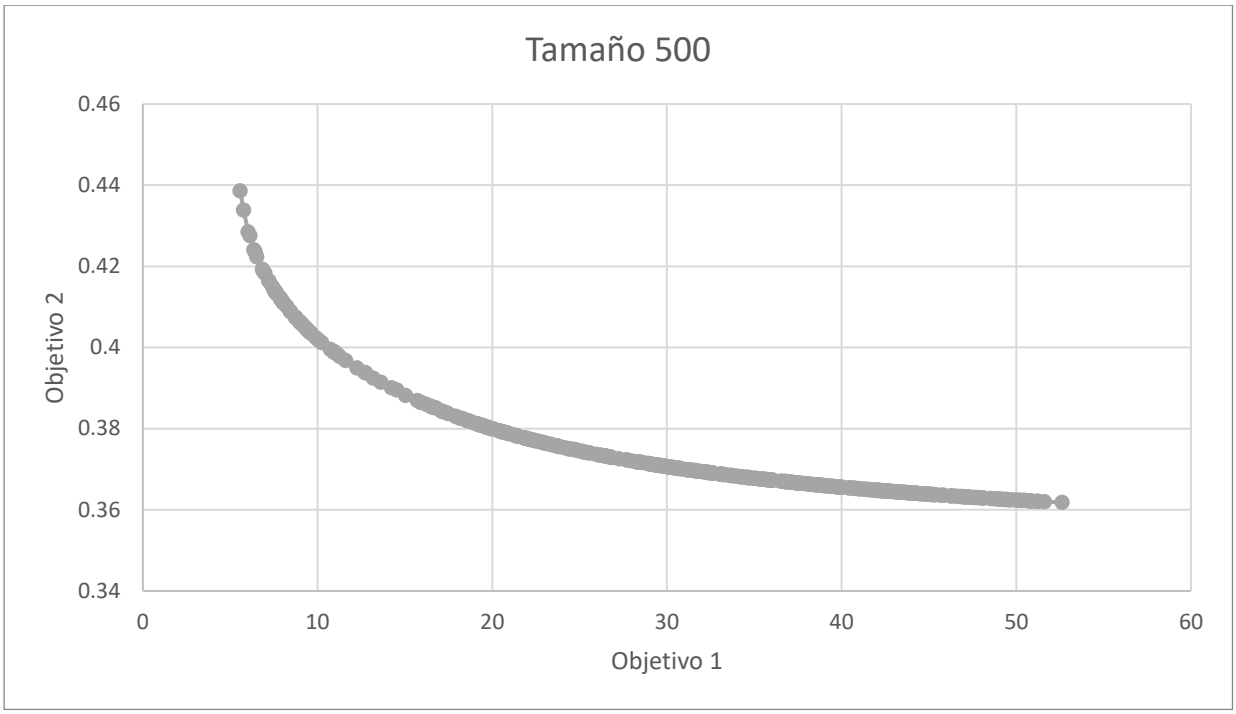

Figura 6.6 Frontera de Pareto de Tamaño 500 2P-1TMD

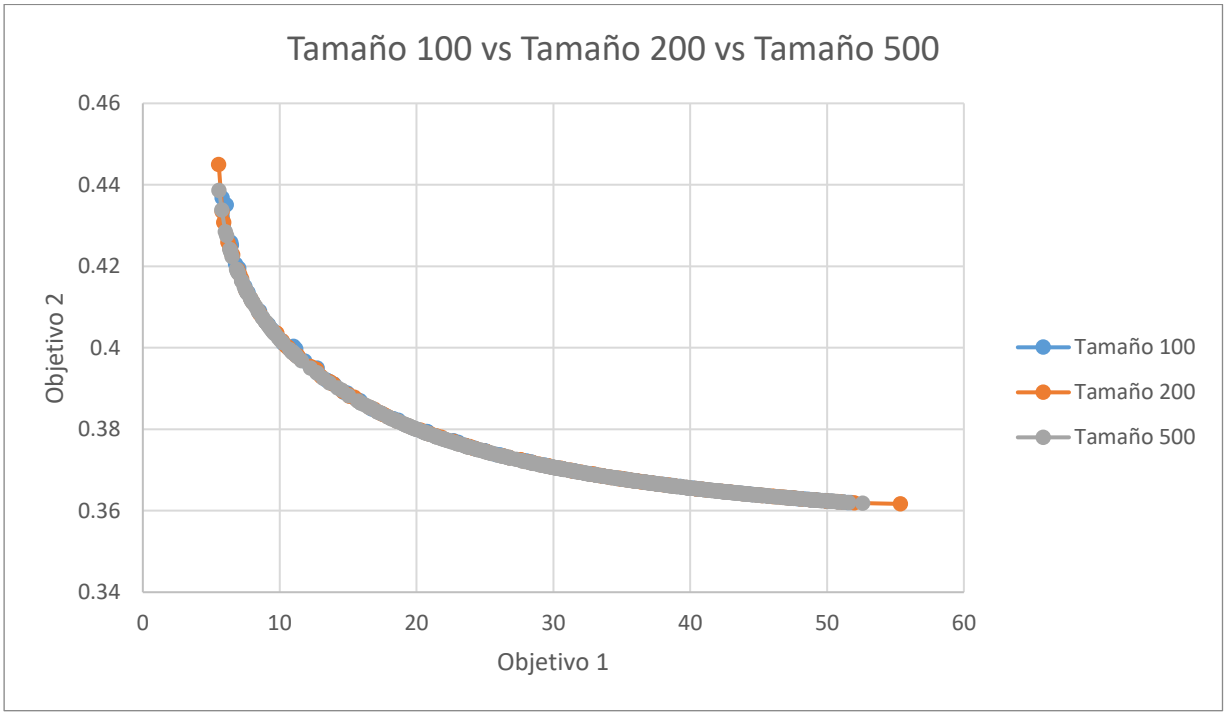

Figura 6.7 Comparación de los tamaños de población 2P-1TMD

Aplicando los métodos de comparación de las fronteras de Pareto, se obtienen los siguientes resultados:

Set Coverage Metric	Tamaño 100 sobre tamaño 200	0.010
	Tamaño 200 sobre tamaño 100	0.000
	Tamaño 100 sobre tamaño 500	0.002
	Tamaño 500 sobre tamaño 100	0.000
	Tamaño 200 sobre tamaño 500	0.000
	Tamaño 500 sobre tamaño 200	0.005

Spacing	Tamaño 100	0.269704869688923
	Tamaño 200	0.267470668213585
	Tamaño 500	0.076058995806359

Hypervolume	Tamaño 100	98.7640909001093
	Tamaño 200	107.195079920415
	Tamaño 500	101.117607431805

Maximum Spread	Tamaño 100	45.9554304598998
	Tamaño 200	49.8153241489640
	Tamaño 500	47.0394733896343

Siguiendo el criterio del primer experimento, se observa que ocurre algo parecido. El tamaño 100 es mejor desde el punto de vista del primer indicador, aunque las tres fronteras de Pareto son muy próximas puesto que los valores son muy pequeños. El tamaño 500 es mejor en el segundo criterio y el tamaño 200 es mejor teniendo en cuenta los dos últimos criterios. En este caso hay algo más de diferencia en el segundo criterio, pero el tamaño 200 cubre una mayor extensión del espacio de soluciones y, aunque su distribución sea menos uniforme que la del tamaño 500, aporta más información sobre el espacio de soluciones estudiado, por lo que se concluye que el mejor tamaño de población es el 200.

6.1.3. Experimento 3

Para terminar, se va a realizar un estudio sobre el cruce más adecuado para la obtención de resultados. En este apartado se van a comparar los cruces 1 (BLX) y 3 (SBX) para ver cuál es mejor. El cruce 2 (PNX) da problemas, por lo que se omitirá.

La tabla de variables del problema es:

Número de variables		3
Valores máximos de las variables	Masa	0.16
	Frecuencia	1000
	Factor de amortiguamiento crítico	0.5
Valores mínimos de las variables	Masa	0.01
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tipo de variable	Masa	0
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tamaño de población		200
Número de evaluaciones		300000
Número de ejecuciones		5
Probabilidad de cruce		90
Probabilidad de mutación		10
Cruce	1: BLX	1 ó 0
	2: PNX	0
	3: SBX	0 ó 1
Radio de nicho		0

Tabla 6.3 Variables 2P-1TMD del experimento 3

Siguiendo el procedimiento ya explicado, se obtienen los dos conjuntos de soluciones de los que se extraen las soluciones no dominadas que conformarán las fronteras de Pareto de este problema.

Se realiza la representación gráfica de las mismas y se procede a realizar el análisis de los resultados.

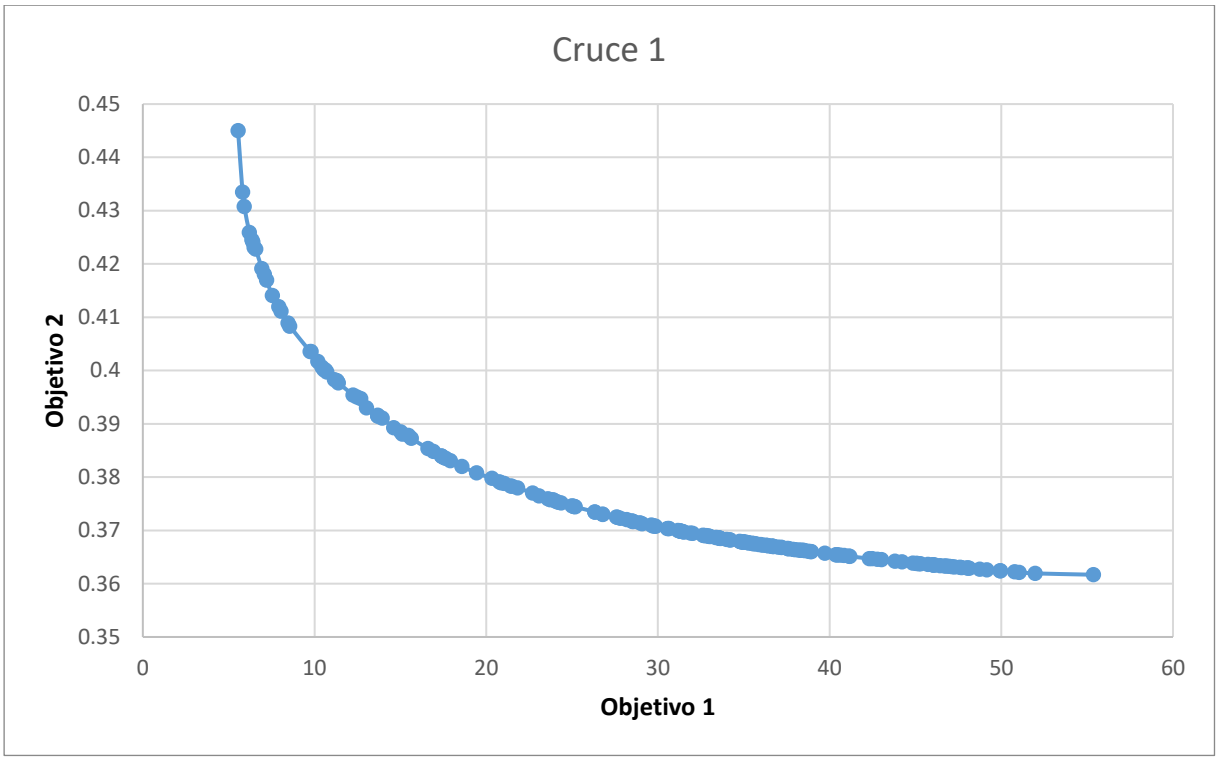

Figura 6.8 Frontera de Pareto del Cruce 1 2P-1TMD

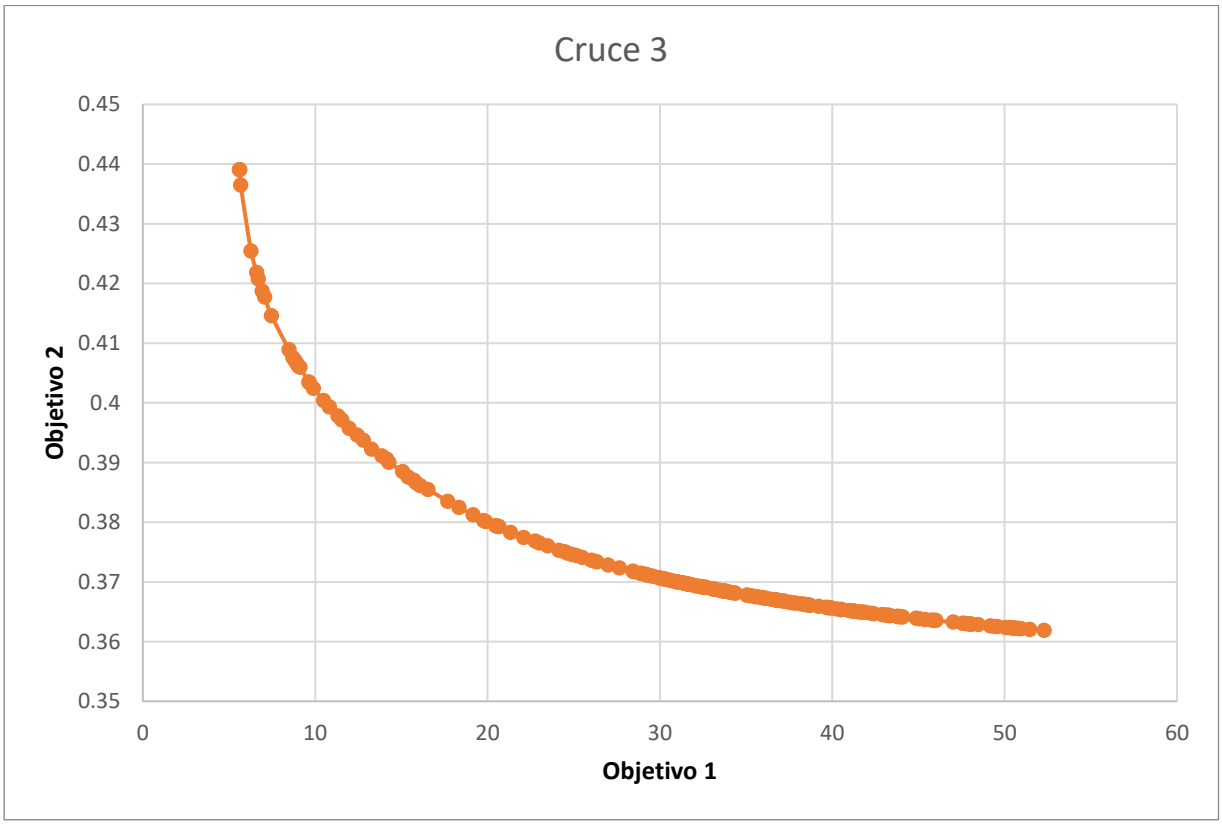

Figura 6.9 Frontera de Pareto del Cruce 3 2P-1TMD

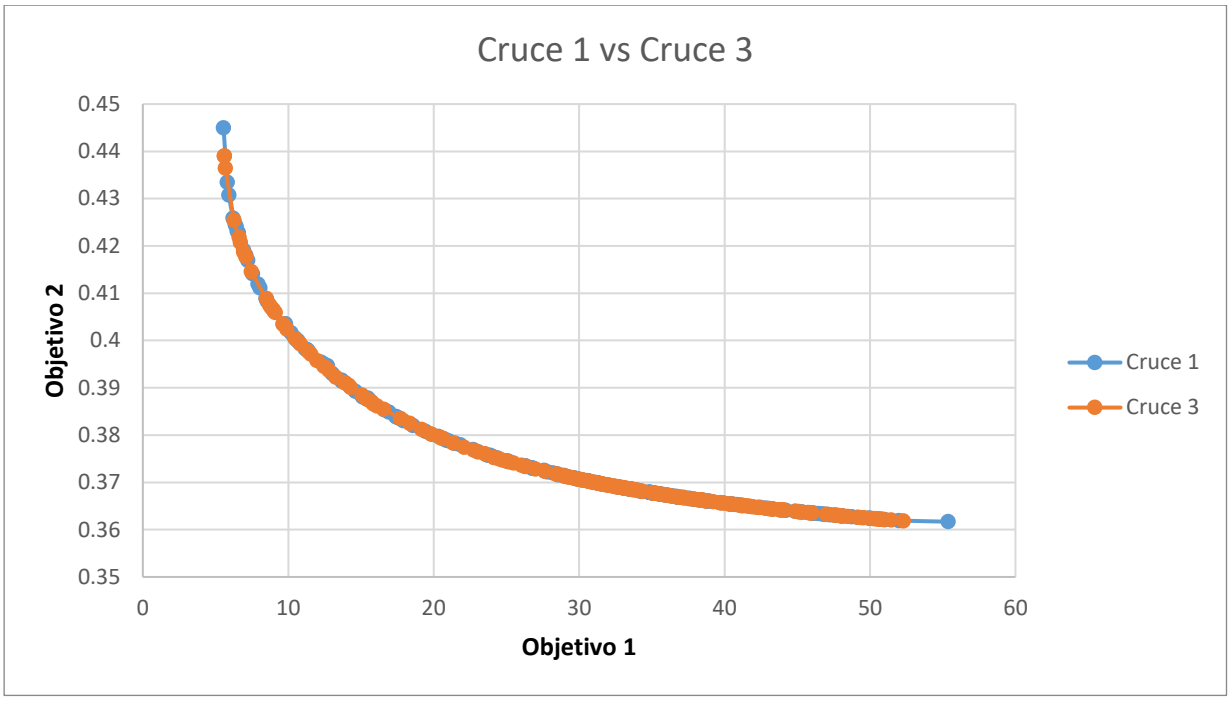

Figura 6.10 Comparación de los cruces 2P-1TMD

Tras la utilización de los métodos de comparación, los resultados son los siguientes:

Set Coverage Metric	Cruce 1 sobre Cruce 3	0.000
	Cruce 3 sobre Cruce 1	0.005

Spacing	Cruce 1	0.267470668213585
	Cruce 3	0.156074874288376

Hypervolume	Cruce 1	107.195079920415
	Cruce 3	100.388570375976

Maximum Spread	Cruce 1	49.81532414896400
	Cruce 3	46.68313031877810

El estudio es similar a los anteriores. El cruce 3 es mejor en los dos primeros indicadores, mientras que el cruce 1 lo es en los dos últimos. Teniendo en cuenta el mismo criterio que en los experimentos anteriores, se concluye que el cruce 1 es mejor puesto que proporciona más información sobre el espacio de soluciones estudiado, aunque su distribución sea menos uniforme que la del cruce 3.

6.1.4. Experimento 4

Para finalizar el estudio, se va a comprobar si hay algún cambio al ejecutar el algoritmo multiplicando el objetivo 2 (de menor orden de magnitud que el objetivo 1) por 100 en él respecto a los resultados obtenidos en la última experimentación, es decir, utilizando los resultados de los tres experimentos anteriores.

La tabla de variables es:

Número de variables		3
Valores máximos de las variables	Masa	0.16
	Frecuencia	1000
	Factor de amortiguamiento crítico	0.5
Valores mínimos de las variables	Masa	0.01
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tipo de variable	Masa	0
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tamaño de población		200
Número de evaluaciones		300000
Número de ejecuciones		5
Probabilidad de cruce		90
Probabilidad de mutación		10
Cruce	1: BLX	1
	2: PNX	0
	3: SBX	0
Radio de nicho		0

Tabla 6.4 Variables 2P-1TMD del experimento 4

Se realiza la ejecución y, posteriormente, se procede con el análisis.

Al igual que antes, los indicadores a utilizar son los mismos, pero para la comparación se va a multiplicar el objetivo 2 del algoritmo original (sin multiplicar) por 100, puesto que, a priori, los resultados no deberían verse modificados y para el estudio se requiere que ambos conjuntos tengan el mismo orden de magnitud.

En las siguientes imágenes se puede apreciar la representación gráfica de los resultados.

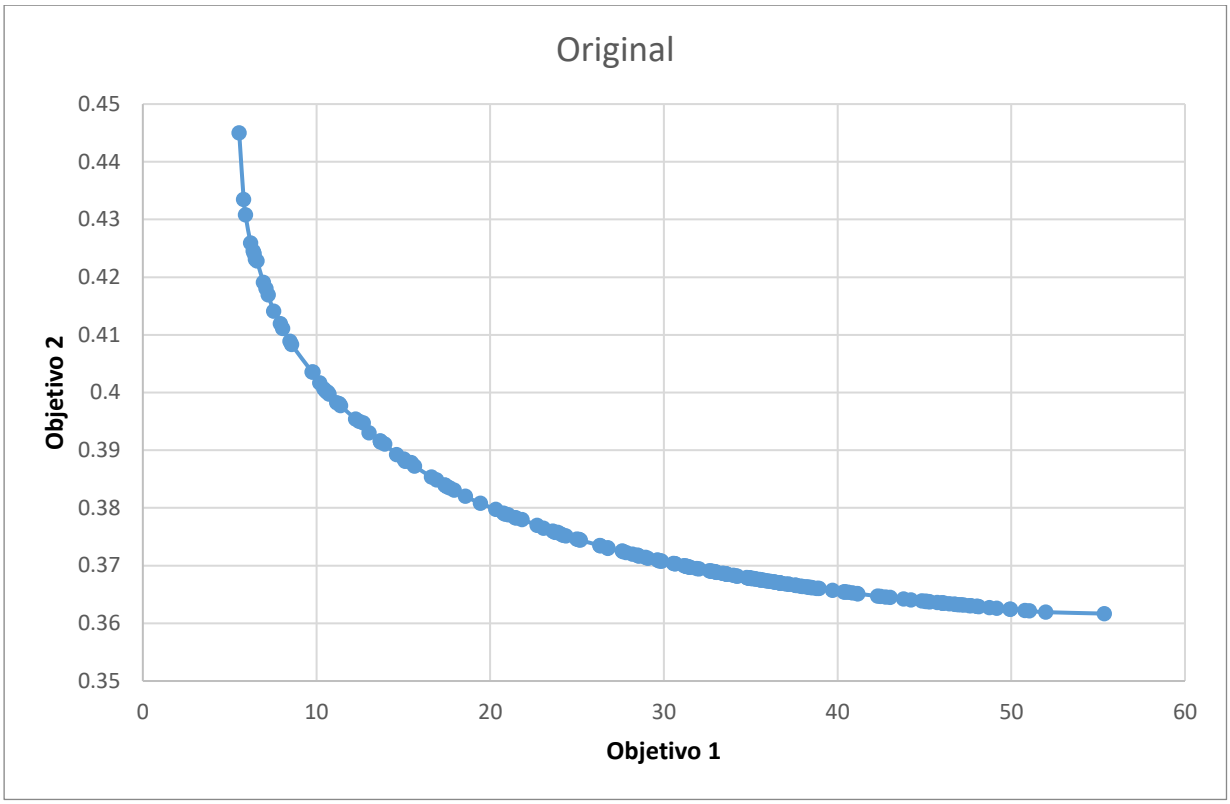

Figura 6.11 Frontera de Pareto Original 2P-1TMD

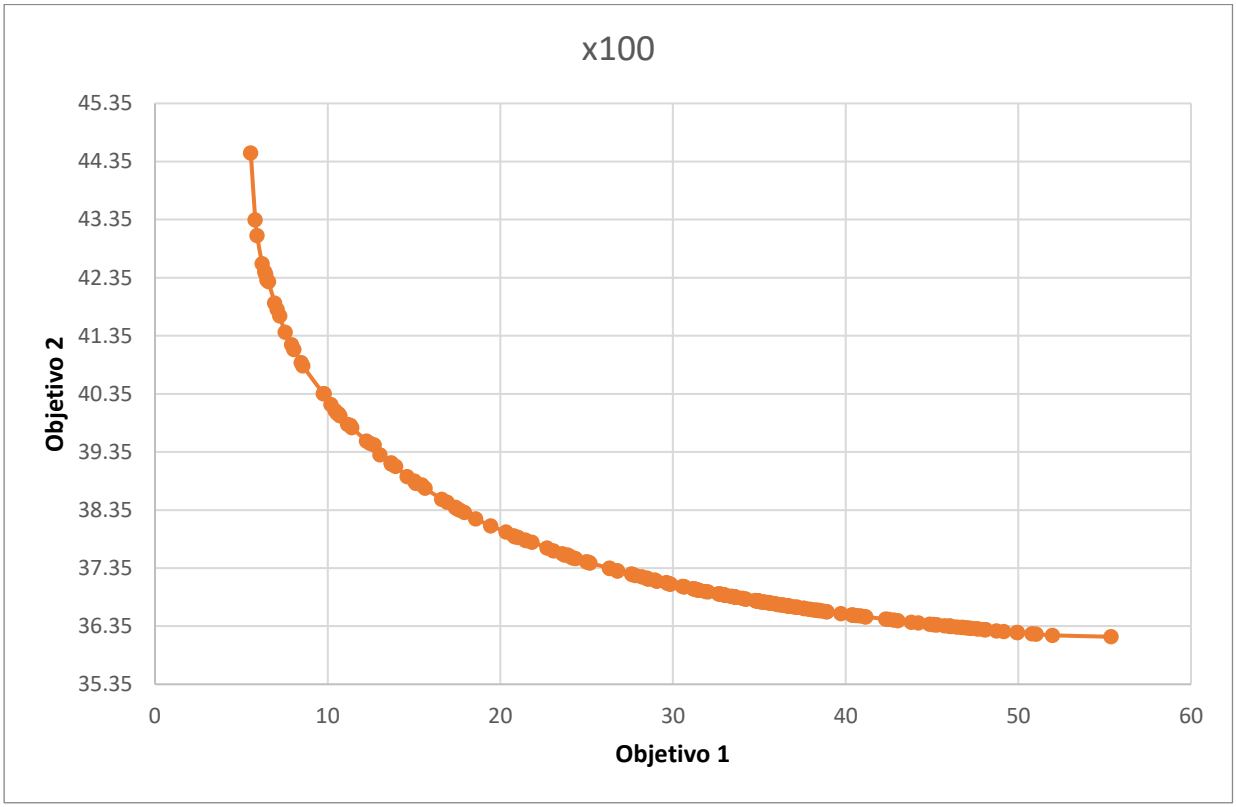

Figura 6.12 Frontera de Pareto x100 2P-1TMD

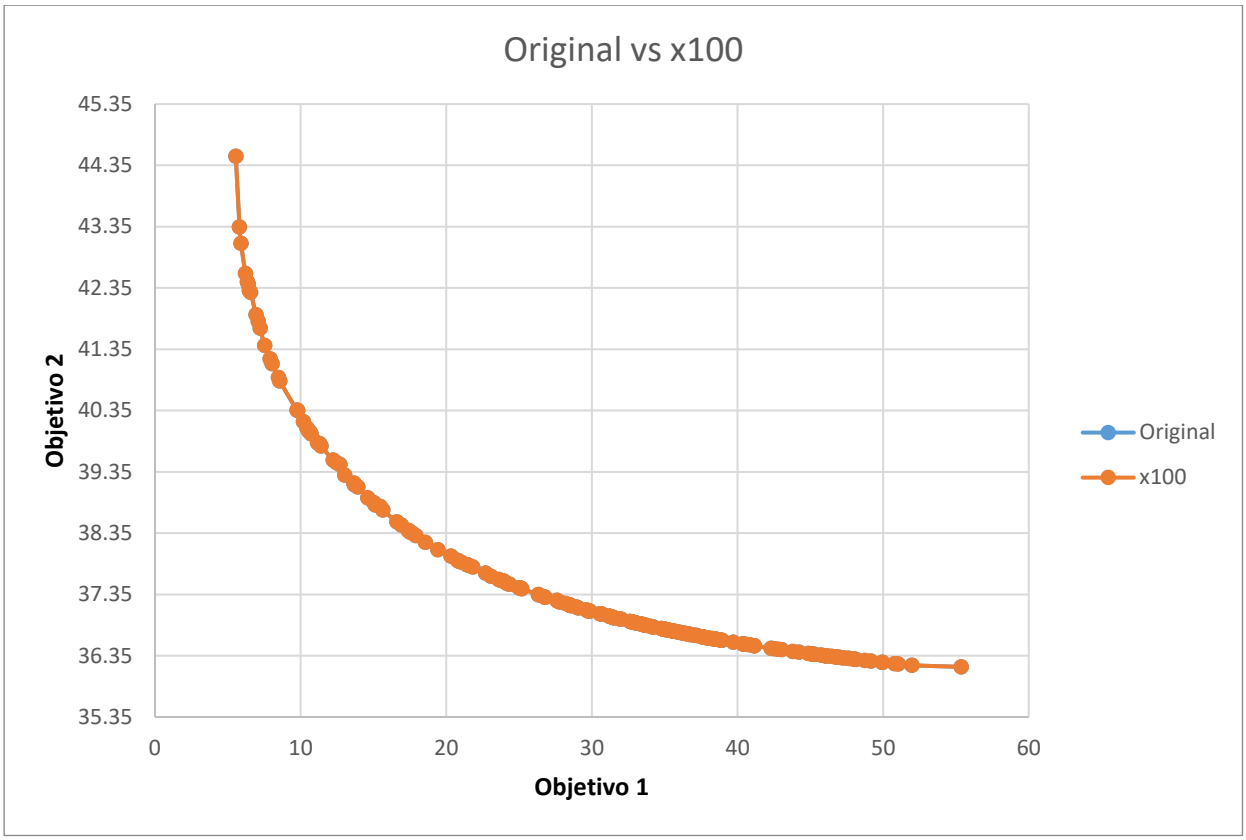

Figura 6.13 Comparación de las ejecuciones 2P-1TMD

Aplicando las métricas de comparación de las fronteras de Pareto obtenidas, los resultados son:

Set Coverage Metric	Original sobre x100	0.000
	x100 sobre original	0.000

Spacing	Original	0.291592997238588
	x100	0.291592995871351

Hypervolume	Original	460.087599457508
	x100	460.087598446403

Maximum Spread	Original	50.50686366436150
	x100	50.50686365941390

A la vista de los mismos, se observa que todos los indicadores son iguales, por lo que no hay ninguna diferencia entre utilizar el algoritmo original y utilizar el algoritmo con el objetivo 2 multiplicado por 100.

6.1.5. Experimento 5

Una vez se han estudiado los parámetros anteriores, es interesante ver cómo afecta un cambio en el valor de la masa máxima. Para eso se ha realizado este último experimento. La tabla de variables de este problema es:

Número de variables		3
Valores máximos de las variables	Masa	0.16, 0.12, 0.08 ó 0.04
	Frecuencia	1000
	Factor de amortiguamiento crítico	0.5
Valores mínimos de las variables	Masa	0.01
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tipo de variable	Masa	0
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tamaño de población		200
Número de evaluaciones		300000
Número de ejecuciones		5
Probabilidad de cruce		90
Probabilidad de mutación		10
Cruce	1: BLX	1
	2: PNX	0
	3: SBX	0
Radio de nicho		0

Tabla 6.5 Variables 2P-1TMD del experimento 5

El modo de operar es análago a los casos anteriores. Se realiza la ejecución, se obtienen los valores no dominados para cada una de las masas y se procede a la representación de las fronteras de Pareto obtenidas.

En este caso, se adjunta tan solo la representación de todas las fronteras de Pareto, puesto que se diferencian claramente unas de otras.

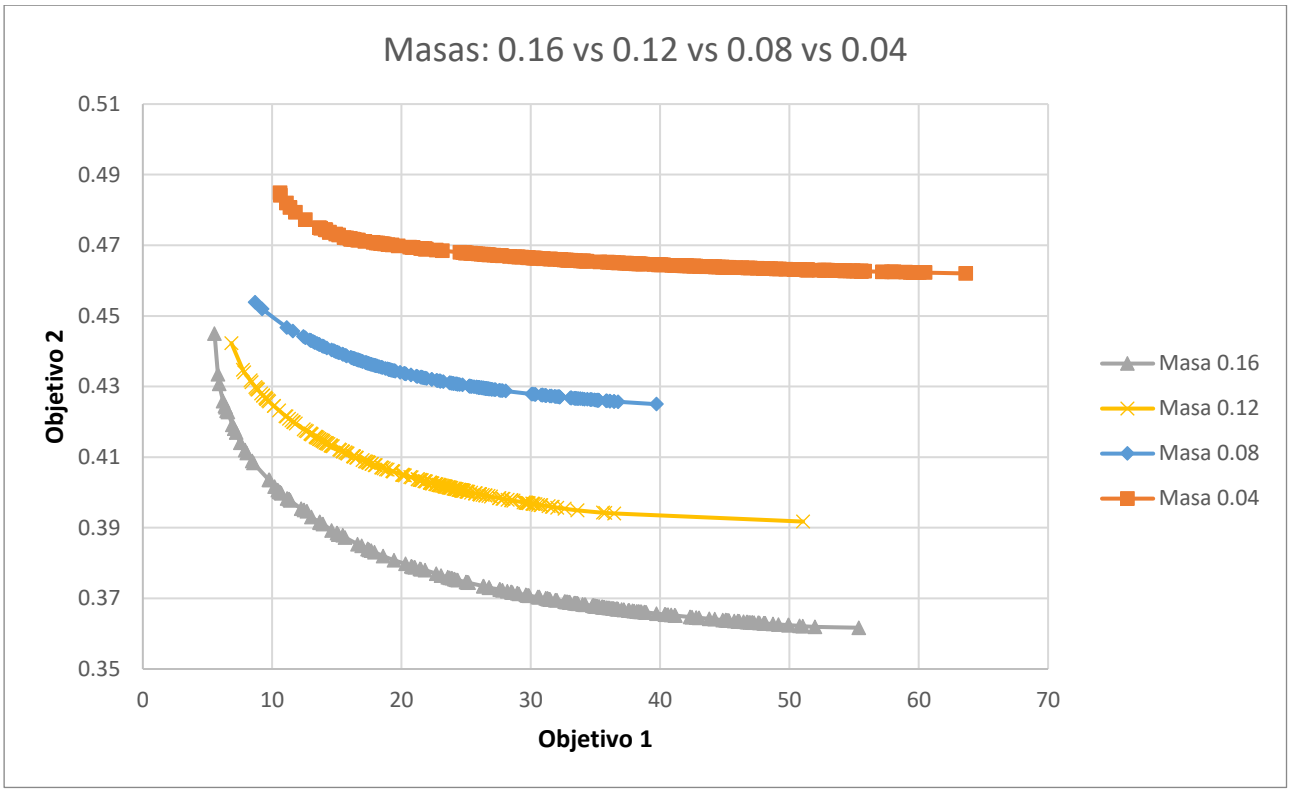

Figura 6.14 Comparación de las masas 2P-1TMD

A simple vista puede verse que, al aumentar la masa, la frontera de Pareto mejora, puesto que se desplaza hacia el origen mejorando ambas funciones objetivo.

Para obtener un valor cuantificable y realizar una comparación de las curvas para ver la mejora que se produce al modificar la masa entre las distintas curvas, se va a obtener la intersección de cada curva con una recta tal como se indica en el siguiente gráfico.

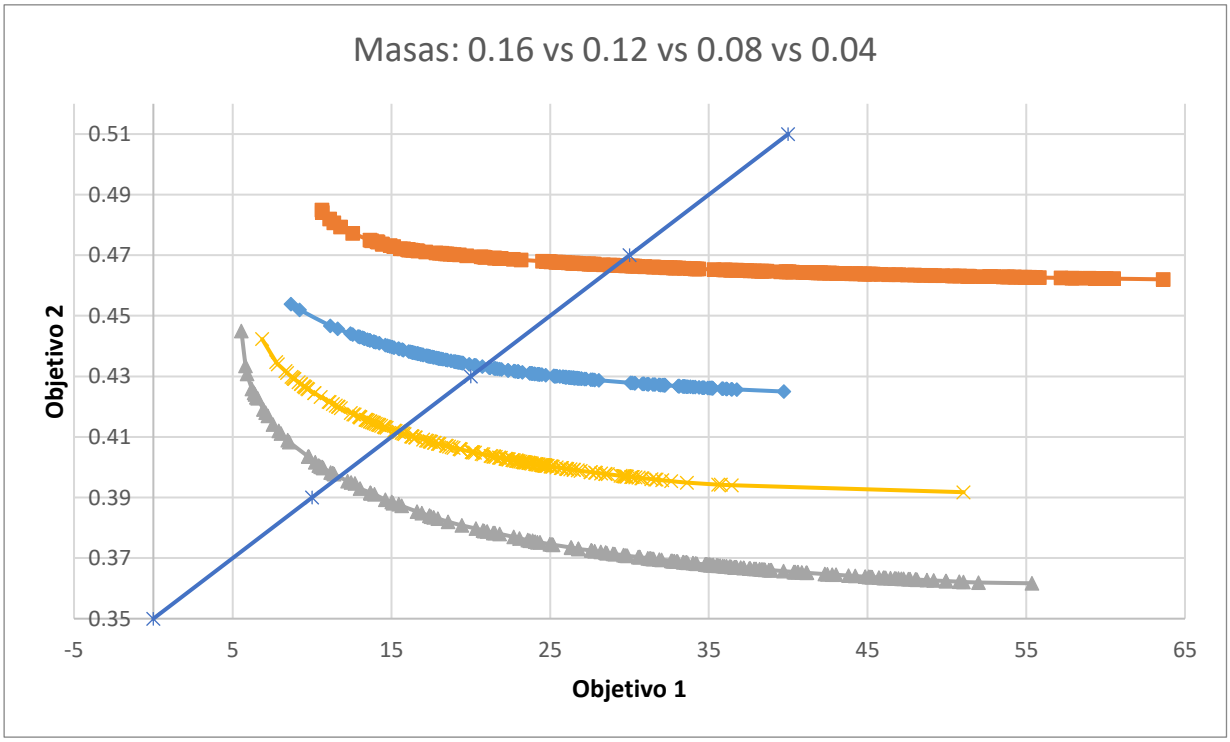

Figura 6.15 Mejora fronteras de Pareto vs Masa

Masa	Objetivo 1	Objetivo 2	Mejora Objetivo 1 (%)	Mejora Objetivo 2 (%)
0,16	11,1642969	0,398249958	26,46	3,37
0,12	15,18132124	0,412120996	24,91	4,97
0,08	20,21981837	0,433712105	30,23	7,07
0,04	28,97864577	0,466706956	-	-

Tabla 6.6 Intersección Curva Auxiliar con Fronteras de Pareto

Como se observa, al aumentar la masa mejoran ambos objetivos. Debido a que el objetivo 2 tiene un menor rango, la mejora es pequeña, pero desde el punto de vista del objetivo 1 las mejoras son significativas.

En cualquier caso, aumentar la masa tiene un impacto positivo sobre las fronteras de Pareto.

6.2. Edificio 2P-2TMD

El modo de proceder es el mismo que en el anterior apartado, solo que, en este caso, al haber 2 TMD, el número de variables aumenta.

6.2.1. Experimento 1

Al igual que en el edificio 2P-1TMD, el primer experimento se centra en determinar con qué radio de nicho se obtienen mejores resultados. La tabla de variables de este experimento es:

Número de variables		3
Valores máximos de las variables	Masa TMD 1	0.21
	Masa TMD 2	0.21
	Frecuencia TMD 1	10
	Frecuencia TMD 2	10
	Factor de amortiguamiento crítico TMD 1	0.5
	Factor de amortiguamiento crítico TMD 2	0.5
	Planta TMD 1	2
	Planta TMD 2	2
Valores mínimos de las variables	Masa TMD 1	0.01
	Masa TMD 2	0.01
	Frecuencia TMD 1	0.5
	Frecuencia TMD 2	0.5
	Factor de amortiguamiento crítico TMD 1	0
	Factor de amortiguamiento crítico TMD 2	0
	Planta TMD 1	1
	Planta TMD 2	1
Tipo de variable	Masa TMD 1	0
	Masa TMD 2	0
	Frecuencia TMD 1	0
	Frecuencia TMD 2	0
	Factor de amortiguamiento crítico TMD 1	0
	Factor de amortiguamiento crítico TMD 2	0
	Planta TMD 1	1
	Planta TMD 2	1
Tamaño de población		200
Número de evaluaciones		300000
Número de ejecuciones		5
Probabilidad de cruce		90
Probabilidad de mutación		10
Cruce	1: BLX	1
	2: PNX	0
	3: SBX	0
Radio de nicho		0 ó -1

Tabla 6.7 Variables 2P-2TMD del experimento 1

Con estos datos se realiza el experimento y se obtienen los dos conjuntos de soluciones. Igual que para el edificio 2P-1TMD, es necesario obtener el conjunto de soluciones no dominadas para hallar el frente de Pareto de ambos radios y proceder a la comparación.

La representación gráfica que se obtiene es la siguiente:

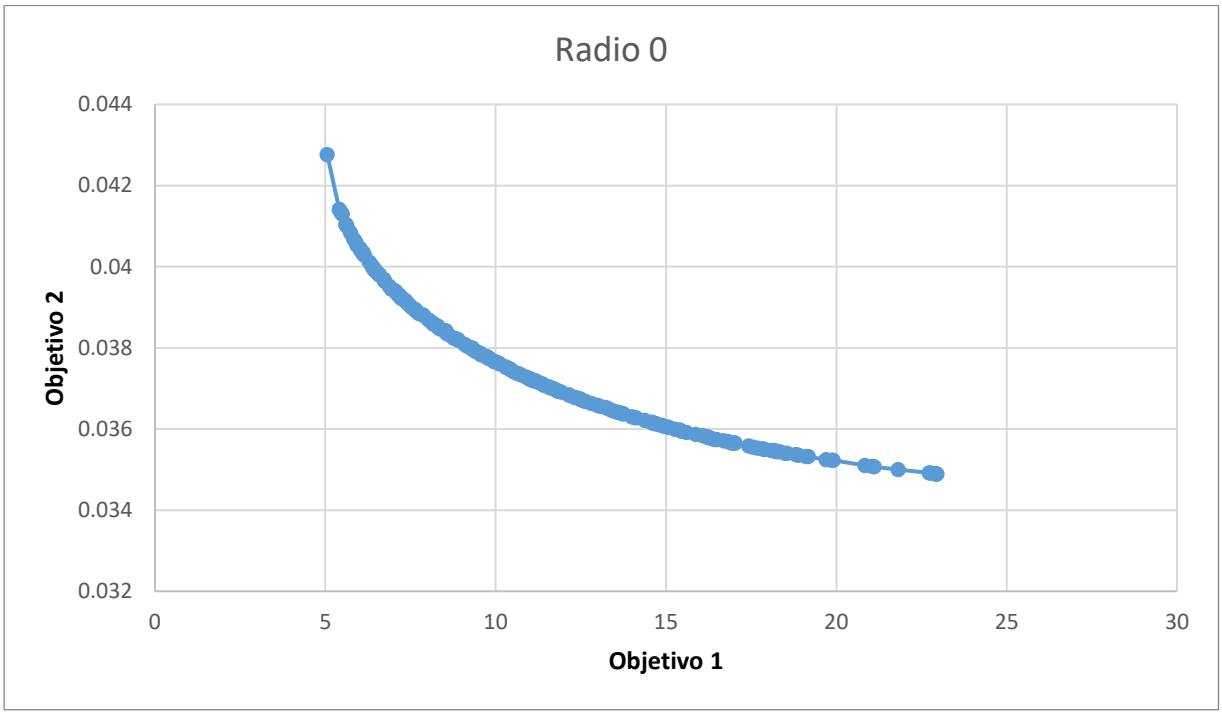

Figura 6.16 Frontera de Pareto del Radio 0 2P-2TMD

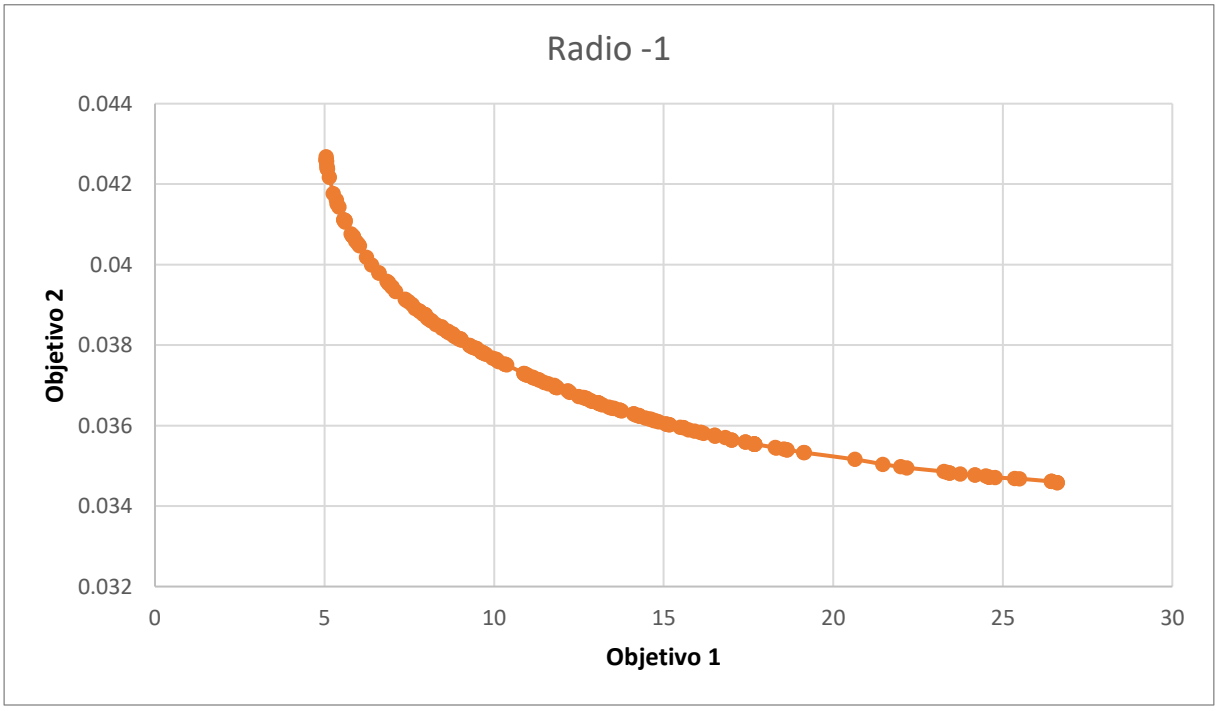

Figura 6.17 Frontera de Pareto del Radio -1 2P-2TMD

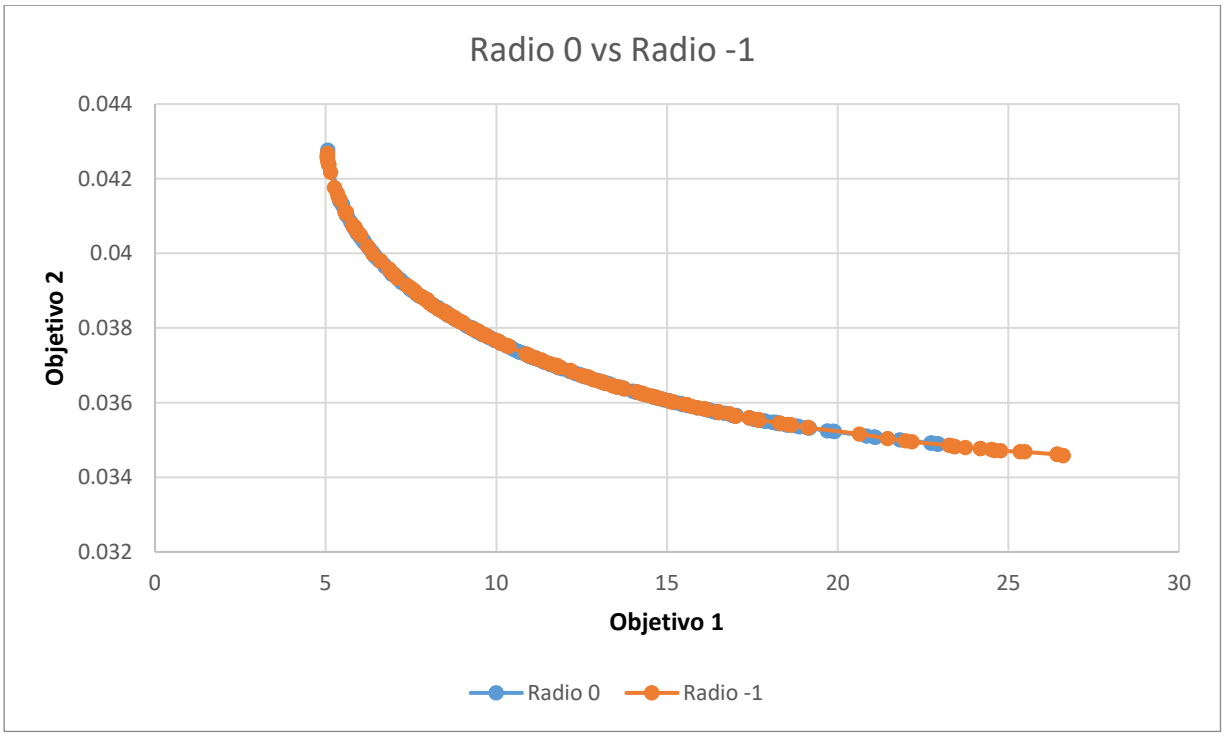

Figura 6.18 Comparación de radios 2P-2TMD

Con los dos conjuntos de soluciones obtenidos, se realiza el estudio de los indicadores para determinar qué radio es mejor. Los resultados son los siguientes:

Set Coverage Metric	Radio 0 sobre Radio -1	0.090
	Radio -1 sobre Radio 0	0.000

Spacing	Radio 0	0.066110788690807
	Radio -1	0.081026446305618

Hypervolume	Radio 0	39.90868154747020
	Radio -1	47.25985562906730

Maximum Spread	Radio 0	17.89395244686530
	Radio -1	21.55563654503310

Siguiendo el mismo procedimiento que en el edificio 2P-1TMD, los dos criterios más relevantes en la toma de decisiones son los dos últimos. Por lo tanto, como el radio -1 es superior en dichos criterios al radio 0, en el segundo indicador las diferencias son muy pequeñas y, como ya se ha explicado en experimentos anteriores que el primer indicador no es muy aclarador, se concluye que el mejor radio es el -1.

6.2.2. Experimento 2

Partiendo del resultado anterior, se prosigue con el estudio de los tamaños de población, quedando la tabla de variables como sigue:

Número de variables		3
Valores máximos de las variables	Masa TMD 1	0.21
	Masa TMD 2	0.21
	Frecuencia TMD 1	10
	Frecuencia TMD 2	10
	Factor de amortiguamiento crítico TMD 1	0.5
	Factor de amortiguamiento crítico TMD 2	0.5
	Planta TMD 1	2
	Planta TMD 2	2
Valores mínimos de las variables	Masa TMD 1	0.01
	Masa TMD 2	0.01
	Frecuencia TMD 1	0.5
	Frecuencia TMD 2	0.5
	Factor de amortiguamiento crítico TMD 1	0
	Factor de amortiguamiento crítico TMD 2	0
	Planta TMD 1	1
	Planta TMD 2	1
Tipo de variable	Masa TMD 1	0
	Masa TMD 2	0
	Frecuencia TMD 1	0
	Frecuencia TMD 2	0
	Factor de amortiguamiento crítico TMD 1	0
	Factor de amortiguamiento crítico TMD 2	0
	Planta TMD 1	1
	Planta TMD 2	1
Tamaño de población		100, 200 ó 500
Número de evaluaciones		300000
Número de ejecuciones		5
Probabilidad de cruce		90
Probabilidad de mutación		10
Cruce	1: BLX	1
	2: PNX	0
	3: SBX	0
Radio de nicho		-1

Tabla 6.8 Variables 2P-2TMD del experimento 2

Ahora se fija el radio en -1 y se varía el tamaño de población entre 100, 200 y 500 individuos. Se obtienen los resultados, se hallan las soluciones no dominadas y se construye el frente de Pareto.

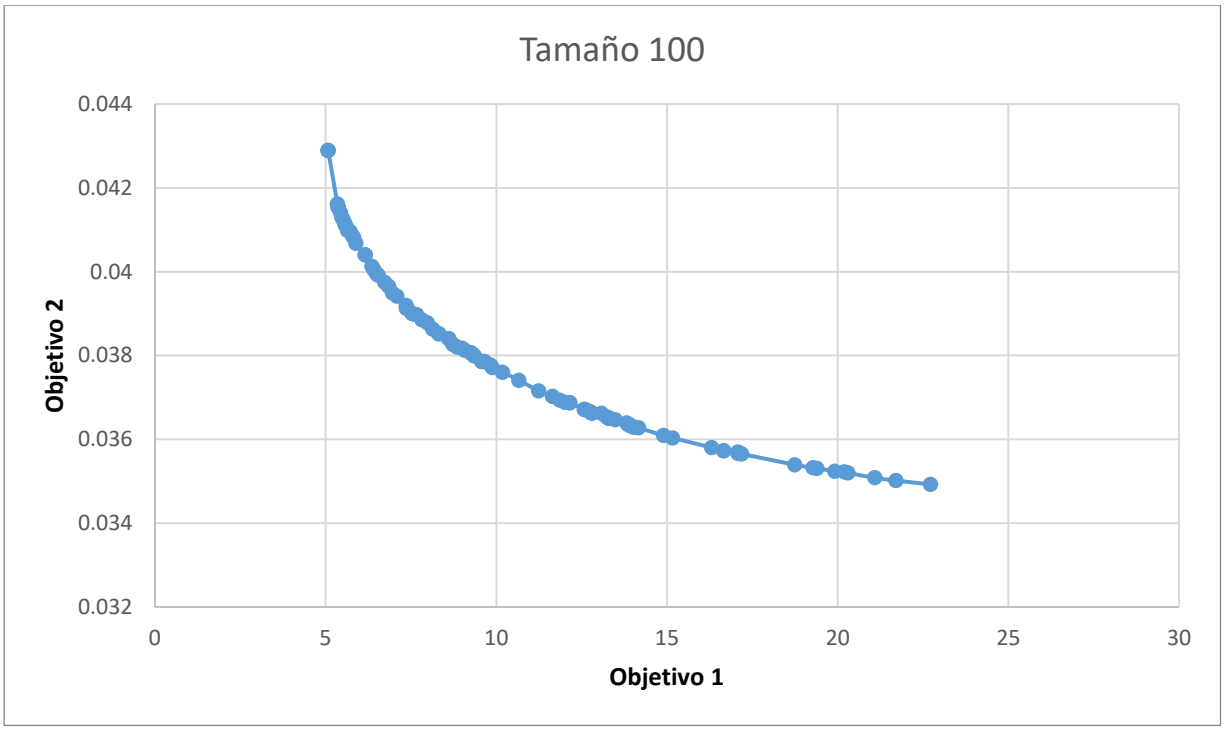

Figura 6.19 Frontera de Pareto de Tamaño 100 2P-2TMD

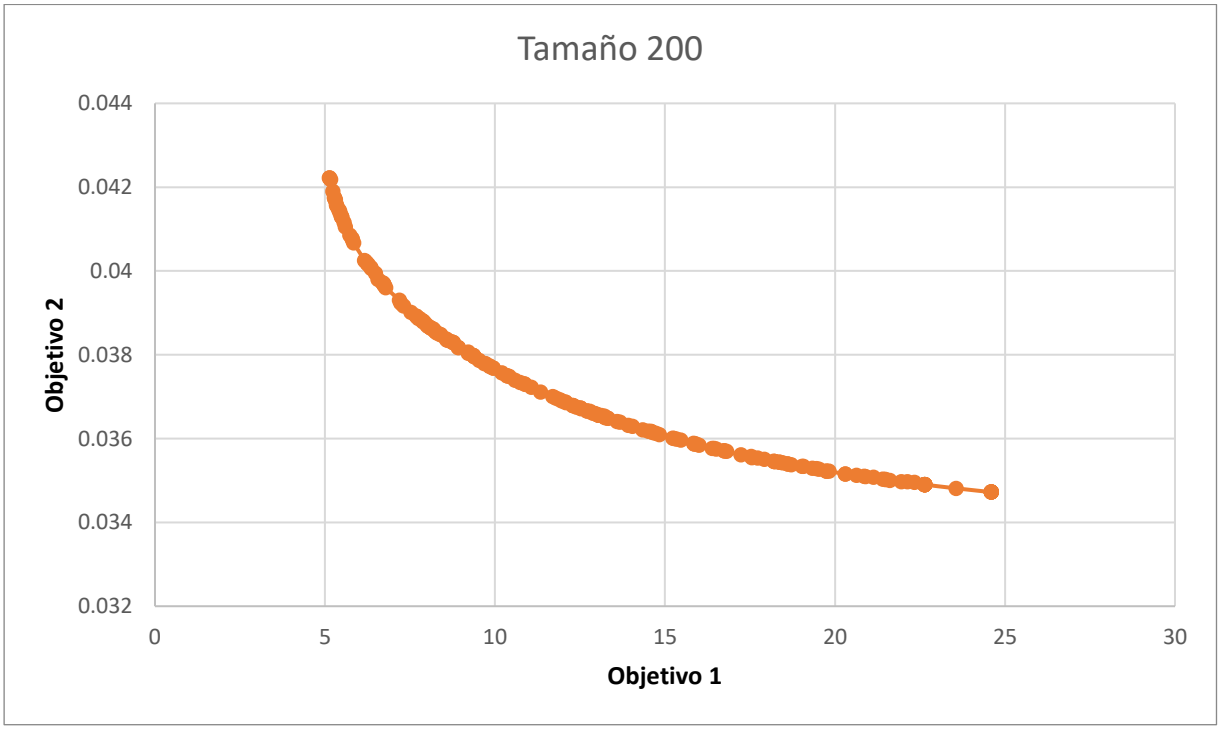

Figura 6.20 Frontera de Pareto de Tamaño 200 2P-2TMD

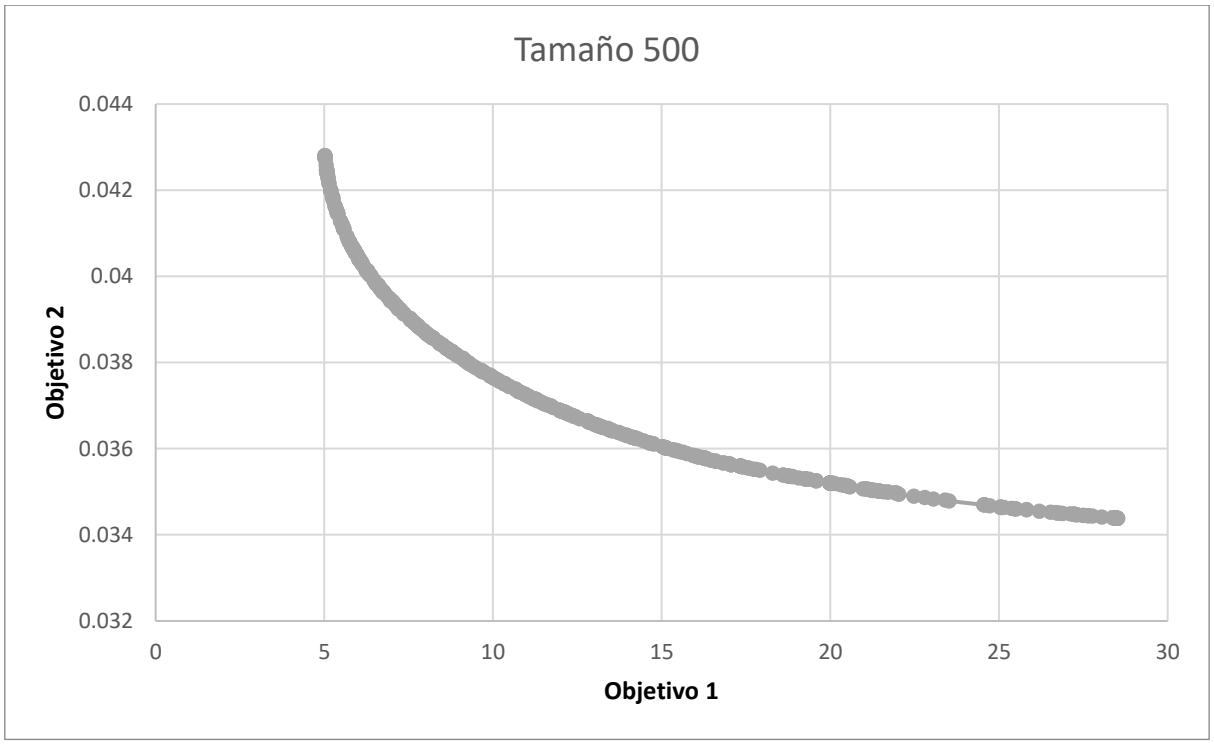

Figura 6.21 Frontera de Pareto de Tamaño 500 2P-2TMD

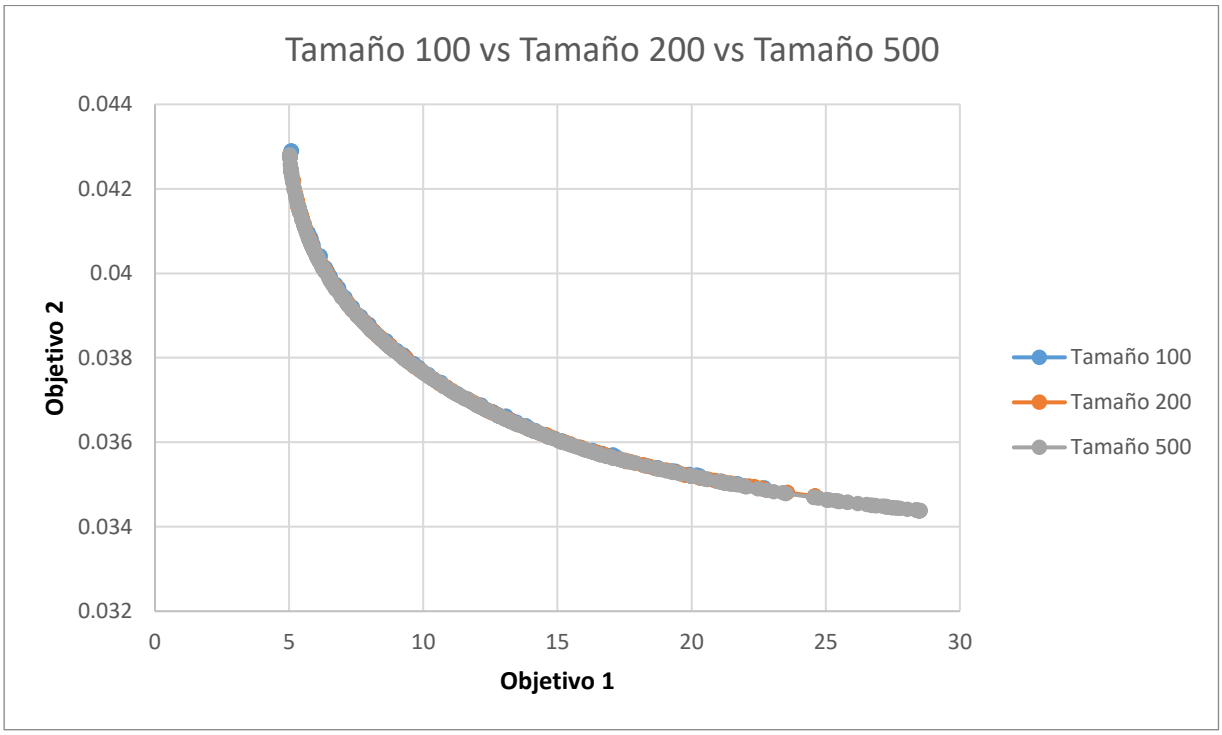

Figura 6.22 Comparación de los tamaños de población 2P-2TMD

Utilizando los métodos de comparación de las fronteras de Pareto, se obtienen los siguientes resultados:

Set Coverage Metric	Tamaño 100 sobre tamaño 200	0.035
	Tamaño 200 sobre tamaño 100	0.000
	Tamaño 100 sobre tamaño 500	0.108
	Tamaño 500 sobre tamaño 100	0.000
	Tamaño 200 sobre tamaño 500	0.096
	Tamaño 500 sobre tamaño 200	0.000

Spacing	Tamaño 100	0.157427129339336
	Tamaño 200	0.083830223805632
	Tamaño 500	0.041117417107199

Hypervolume	Tamaño 100	39.4073672961726
	Tamaño 200	43.057820449596
	Tamaño 500	51.173673007487

Maximum Spread	Tamaño 100	17.6435165865165
	Tamaño 200	19.4675693620255
	Tamaño 500	23.5024376635667

Como se puede observar, en este caso la decisión es más sencilla, puesto que el tamaño 500 es mejor en todos los indicadores a los tamaños 100 y 200 excepto en el primero (que, como ya se ha visto, no aporta mucha información) y, por lo tanto, para la experimentación posterior se procederá con el tamaño de población de 500 individuos.

6.2.3. Experimento 3

Finalmente, el último experimento que se va a realizar es el mismo que en el caso anterior con el objetivo de comparar los cruces 1 (BLX) y 3 (SBX) para ver cuál de los dos arroja mejores resultados.

Las variables del problema son:

Número de variables		3
Valores máximos de las variables	Masa TMD 1	0.21
	Masa TMD 2	0.21
	Frecuencia TMD 1	10
	Frecuencia TMD 2	10
	Factor de amortiguamiento crítico TMD 1	0.5
	Factor de amortiguamiento crítico TMD 2	0.5
	Planta TMD 1	2
	Planta TMD 2	2
Valores mínimos de las variables	Masa TMD 1	0.01
	Masa TMD 2	0.01
	Frecuencia TMD 1	0.5
	Frecuencia TMD 2	0.5
	Factor de amortiguamiento crítico TMD 1	0
	Factor de amortiguamiento crítico TMD 2	0
	Planta TMD 1	1
	Planta TMD 2	1
Tipo de variable	Masa TMD 1	0
	Masa TMD 2	0
	Frecuencia TMD 1	0
	Frecuencia TMD 2	0
	Factor de amortiguamiento crítico TMD 1	0
	Factor de amortiguamiento crítico TMD 2	0
	Planta TMD 1	1
	Planta TMD 2	1
Tamaño de población		500
Número de evaluaciones		300000
Número de ejecuciones		5
Probabilidad de cruce		90
Probabilidad de mutación		10
Cruce	1: BLX	1 ó 0
	2: PNX	0
	3: SBX	0 ó 1
Radio de nicho		-1

Tabla 6.9 Variables 2P-2TMD del experimento 3

Repitiendo el proceso ya explicado, se obtiene la frontera de Pareto al representar las soluciones no dominadas de cada conjunto de soluciones (cruce 1 y cruce 3).

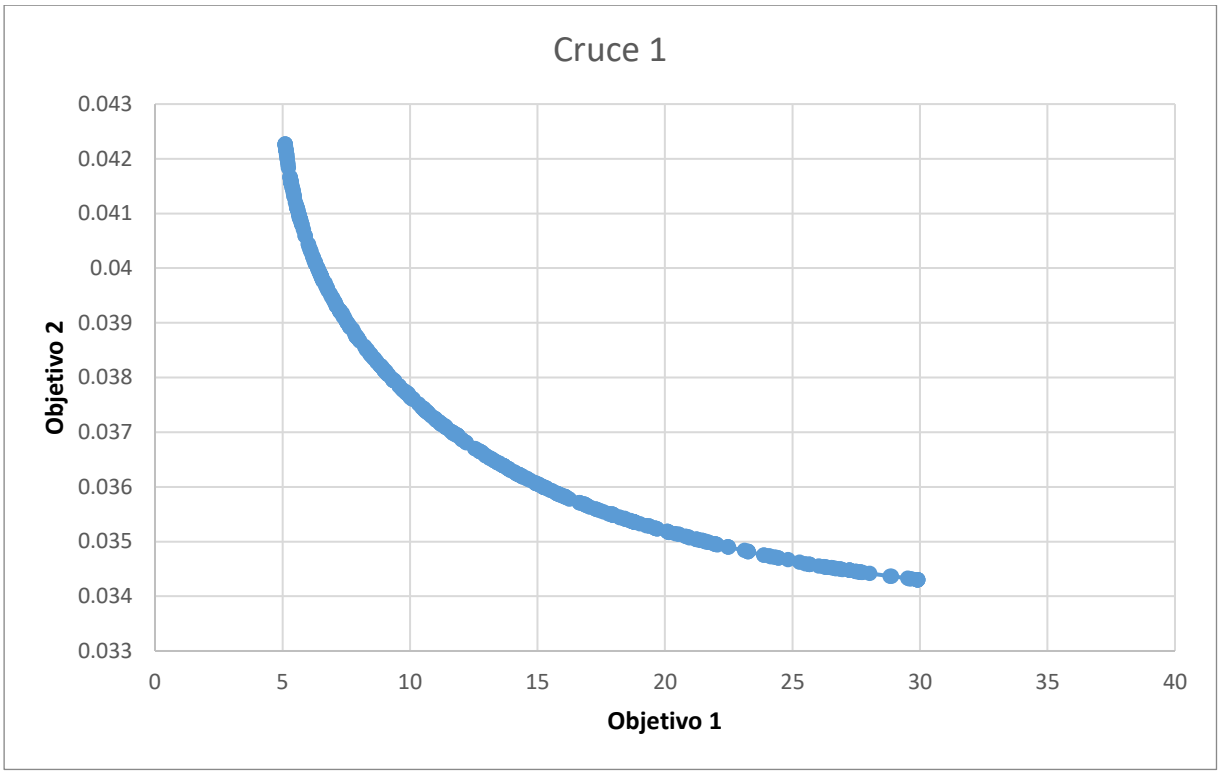

Figura 6.23 Frontera de Pareto del Cruce 1 2P-2TMD

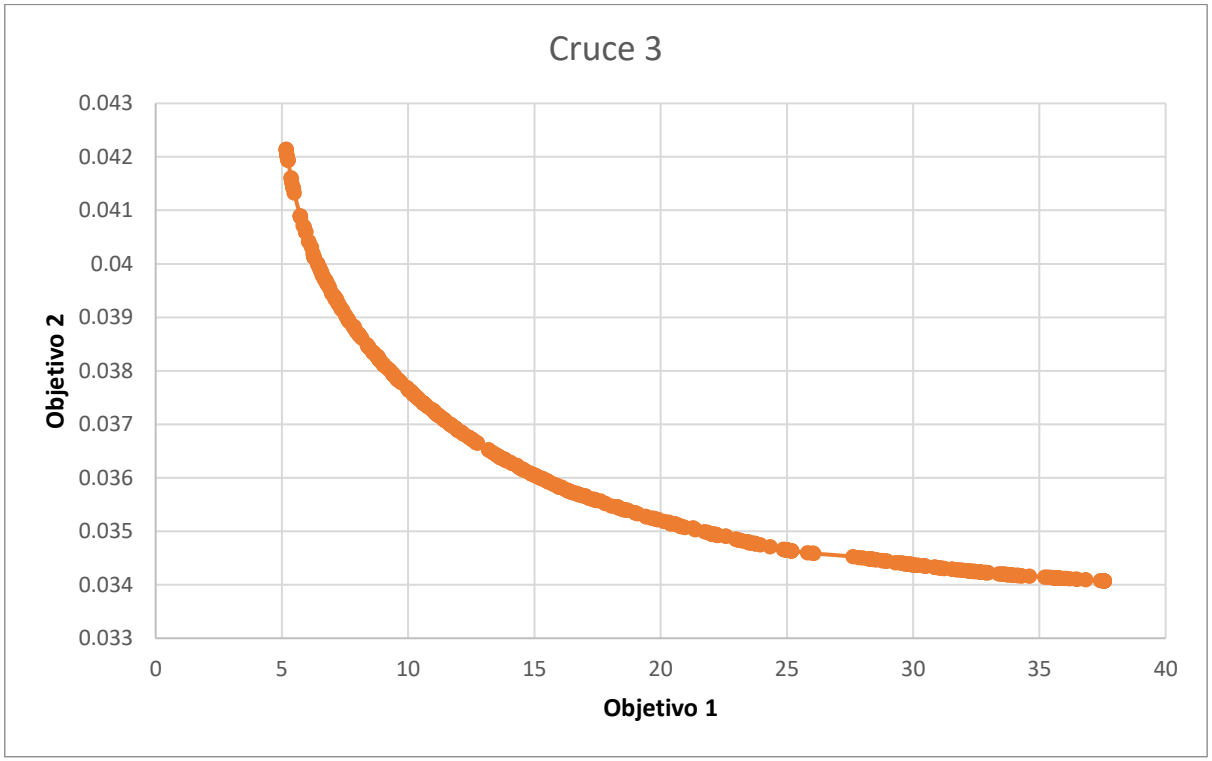

Figura 6.24 Frontera de Pareto del Cruce 3 2P-2TMD

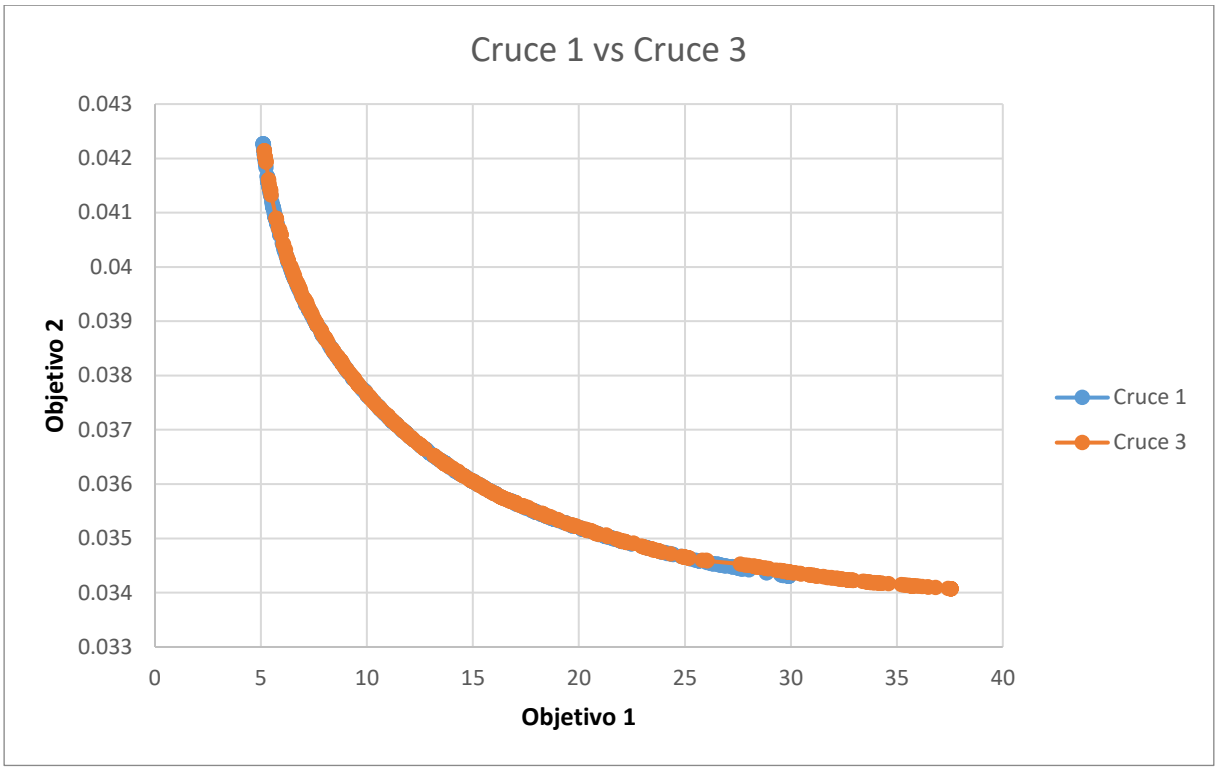

Figura 6.25 Comparación de los cruces 2P-2TMD

Mediante la aplicación de las métricas de comparación de los frentes de Pareto, se hallan los siguientes resultados:

Set Coverage Metric	Cruce 1 sobre Cruce 3	0.194
	Cruce 3 sobre Cruce 1	0.000

Spacing	Cruce 1	0.035301951613882
	Cruce 3	0.042873598179582

Hypervolume	Cruce 1	53.77466244701480
	Cruce 3	69.01547571132400

Maximum Spread	Cruce 1	24.80414455000060
	Cruce 3	32.39600534829350

Para este caso, el cruce 1 es mejor desde el punto de vista del primer indicador. Como ya se ha comentado en otros experimentos anteriores, el primer criterio no es demasiado útil dado que se obtienen valores reducidos debido a que las fronteras de Pareto son próximas entre sí. En cuanto al segundo criterio, los valores son similares siendo algo mejor el cruce 1. Sin embargo, para el tercer y el cuarto indicador, el cruce 3 es superior claramente. Como hasta ahora se han considerado estos dos últimos criterios más fiables y, puesto que en el segundo indicador las diferencias son pequeñas, se concluye que el mejor cruce es el 3.

6.2.4. Experimento 4

Análogamente al edificio 2P-1TMD, aquí también se va a comprobar si multiplicar por 100 en el algoritmo el objetivo 2 provoca algún cambio respecto al algoritmo original.

La tabla de variables es la correspondiente a utilizar toda la información recabada en los tres experimentos previos:

Número de variables		3
Valores máximos de las variables	Masa TMD 1	0.21
	Masa TMD 2	0.21
	Frecuencia TMD 1	10
	Frecuencia TMD 2	10
	Factor de amortiguamiento crítico TMD 1	0.5
	Factor de amortiguamiento crítico TMD 2	0.5
	Planta TMD 1	2
	Planta TMD 2	2
Valores mínimos de las variables	Masa TMD 1	0.01
	Masa TMD 2	0.01
	Frecuencia TMD 1	0.5
	Frecuencia TMD 2	0.5
	Factor de amortiguamiento crítico TMD 1	0
	Factor de amortiguamiento crítico TMD 2	0
	Planta TMD 1	1
	Planta TMD 2	1
Tipo de variable	Masa TMD 1	0
	Masa TMD 2	0
	Frecuencia TMD 1	0
	Frecuencia TMD 2	0
	Factor de amortiguamiento crítico TMD 1	0
	Factor de amortiguamiento crítico TMD 2	0
	Planta TMD 1	1
	Planta TMD 2	1
Tamaño de población		500
Número de evaluaciones		300000
Número de ejecuciones		5
Probabilidad de cruce		90
Probabilidad de mutación		10
Cruce	1: BLX	0
	2: PNX	0
	3: SBX	1
Radio de nicho		-1

Tabla 6.10 Variables 2P-2TMD del experimento 4

Igual que para el 2P-1TMD, hay que tener la precaución de multiplicar por 100 el objetivo 2 de la ejecución del algoritmo original para que la comparación pueda realizarse adecuadamente. La representación gráfica obtenida se adjunta a continuación.

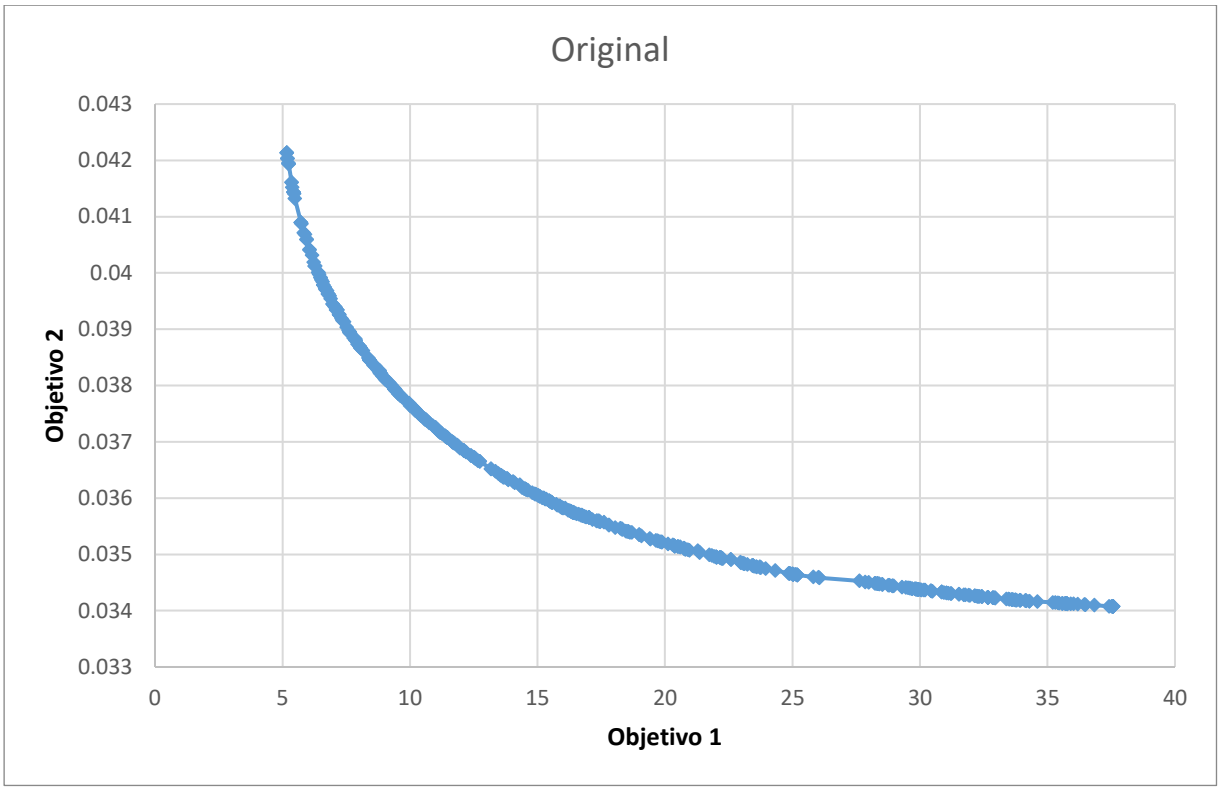

Figura 6.26 Frontera de Pareto Original 2P-2TMD

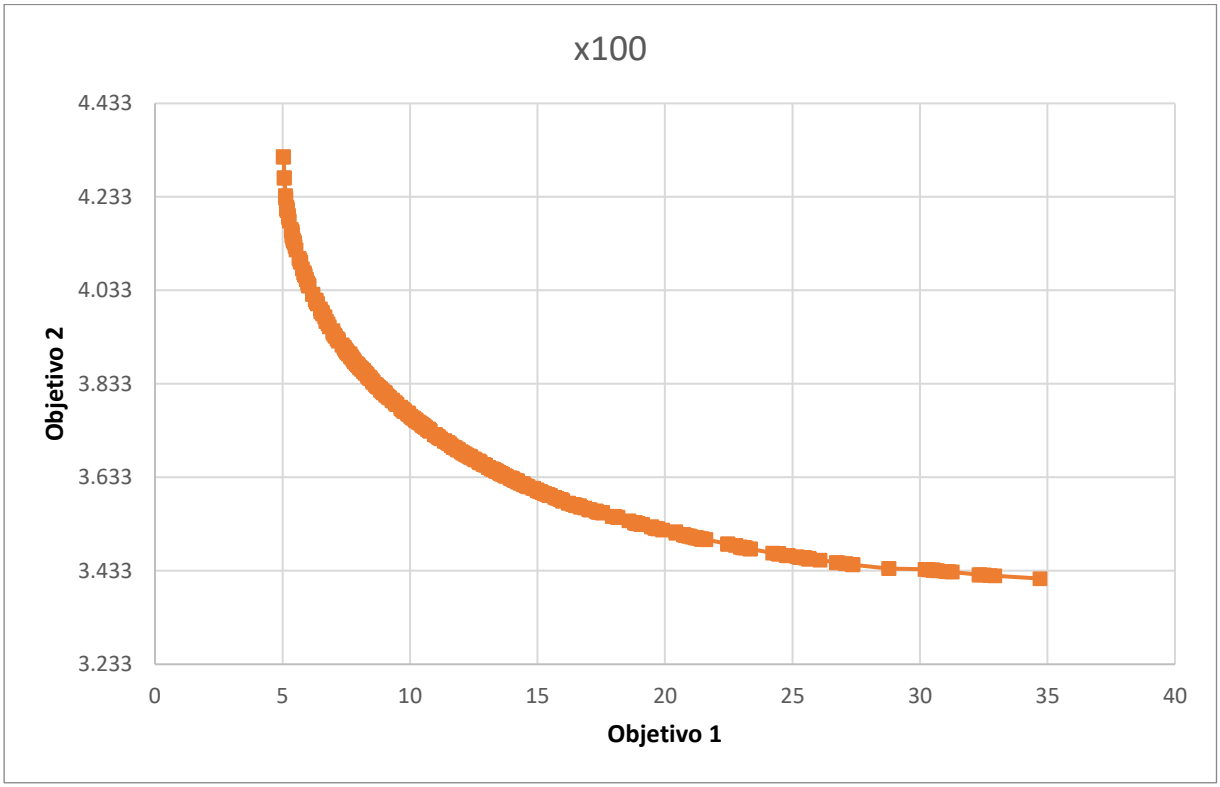

Figura 6.27 Frontera de Pareto Original 2P-2TMD

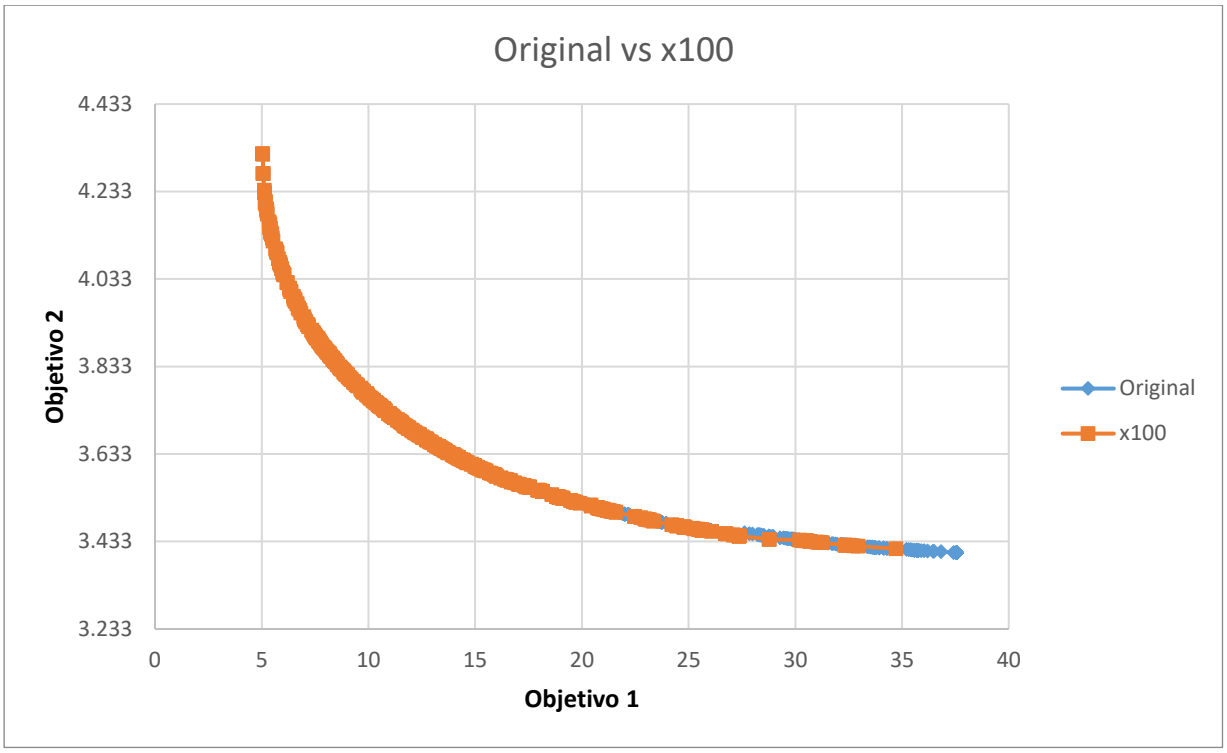

Figura 6.28 Comparación de las ejecuciones 2P-2TMD

Utilizando los métodos de comparación, se obtiene lo siguiente:

Set Coverage Metric	Original sobre x100	0.006
	x100 sobre Original	0.066

Spacing	Original	0.043332500521437
	x100	0.085874919848734

Hypervolume	Original	91.13871082239570
	x100	86.74898147734790

Maximum Spread	Original	32.40603572762480
	x100	29.68645132742740

Respecto al experimento 4 del edificio 2P-1TMD, en este caso hay diferencias en los indicadores, aunque son pequeñas. El primer criterio es muy pequeño en ambos casos, es decir, ambas fronteras son muy cercanas (se puede observar en la figura 6.26). En el resto de indicadores es algo mejor el algoritmo original, por lo que, si hubiera que elegir una opción, sería esta, pero los resultados son muy similares.

6.3. Edificio 5P-1TMD

Para terminar la experimentación, se va a estudiar el comportamiento de las fronteras de Pareto al modificar los parámetros de entrada para un algoritmo programado para un edificio de 5 pisos en el que se va a instalar 1 TMD.

6.3.1. Experimento 1

El experimento 1 es el mismo que en los dos casos anteriores, se modificará el radio entre los valores 0 y -1 para determinar cuál de los dos arroja mejores resultados. La tabla de variables del este experimento es:

Número de variables		3
Valores máximos de las variables	Masa	6340
	Frecuencia	10000
	Factor de amortiguamiento crítico	0.5
Valores mínimos de las variables	Masa	0.001
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tipo de variable	Masa	0
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tamaño de población		200
Número de evaluaciones		300000
Número de ejecuciones		5
Probabilidad de cruce		90
Probabilidad de mutación		10
Cruce	1: BLX	1
	2: PNX	0
	3: SBX	0
Radio de nicho		0 ó -1

Tabla 6.11 Variables 5P-1TMD del experimento 1

Una vez que se obtienen los datos de salida, se procede de forma análoga a los estudios anteriores, es decir, se obtiene el conjunto de soluciones no dominado de cada uno de los radios. A continuación, con dichos datos, se representan las fronteras de Pareto y se procede a realizar la comparación de las mismas.

A continuación, se adjuntan las gráficas obtenidas.

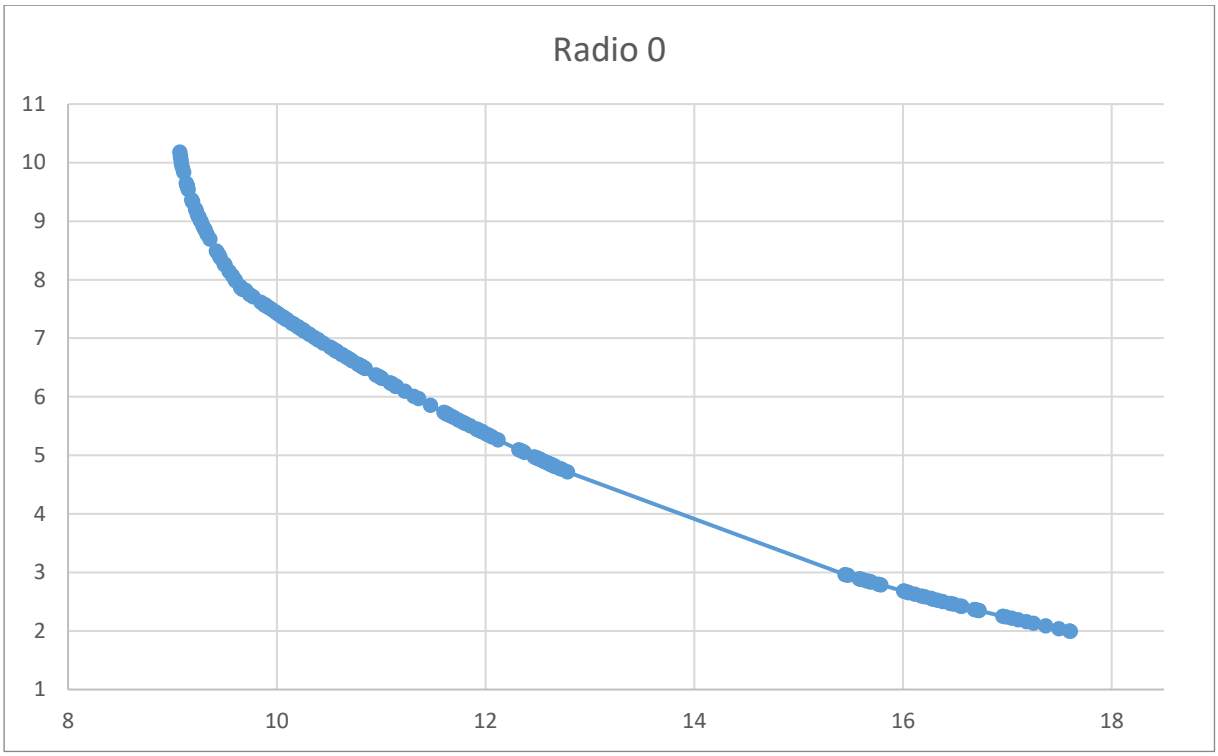

Figura 6.29 Frontera de Pareto de Radio 0 5P-1TMD

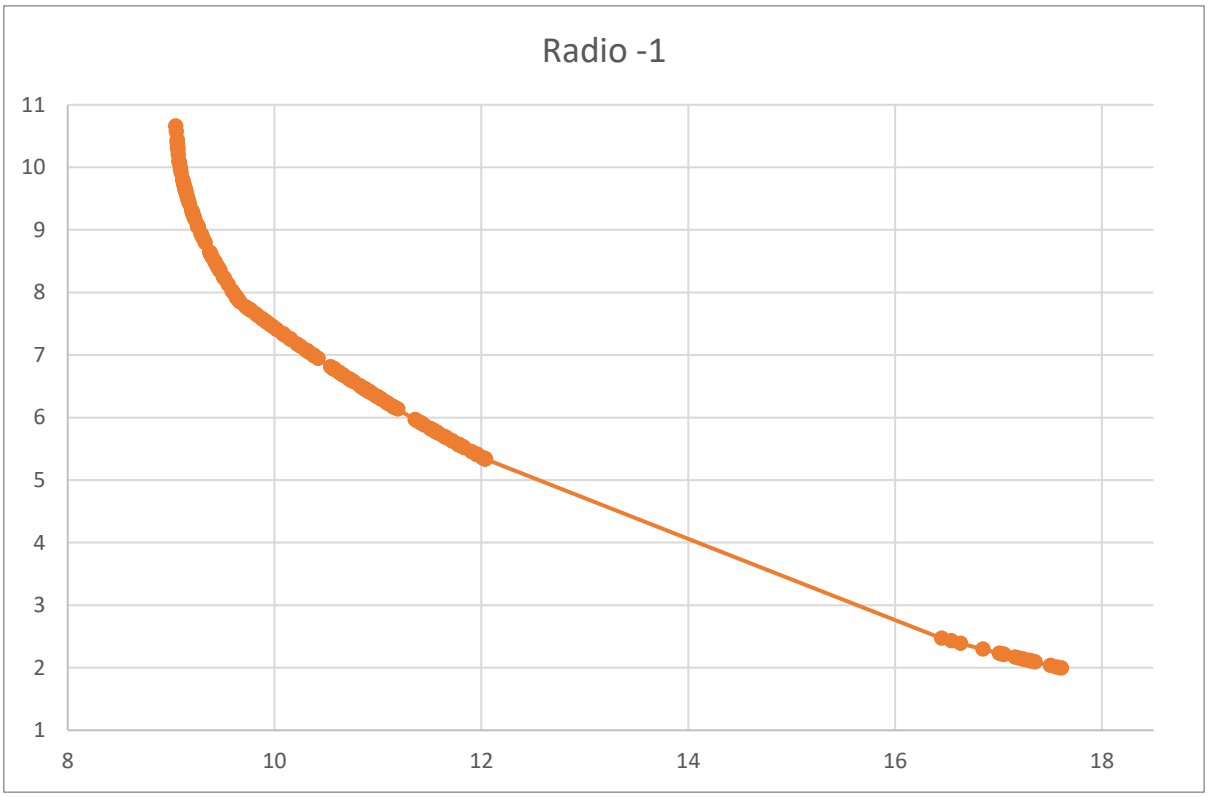

Figura 6.30 Frontera de Pareto de Radio -1 5P-1TMD

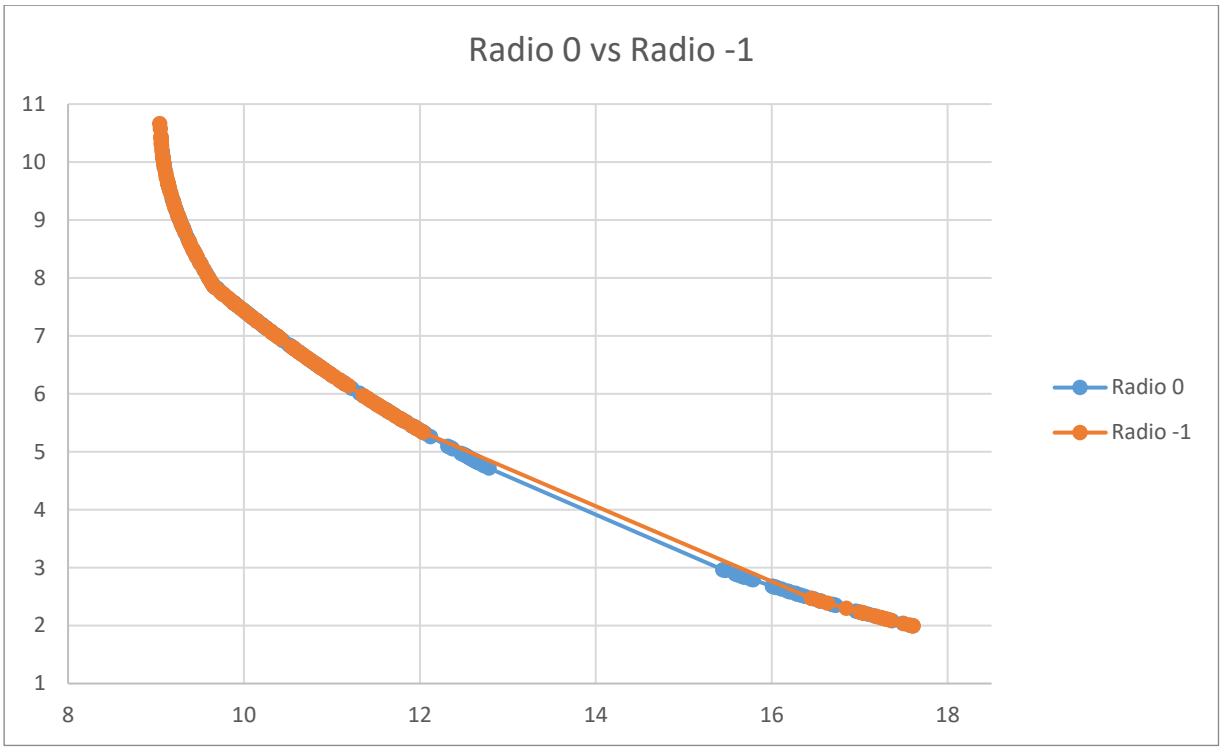

Figura 6.31 Comparación de radios 5P-1TMD

Aplicando los métodos de comparación de Pareto, los resultados son los siguientes:

Set Coverage Metric	Radio 0 sobre Radio -1	0.040
	Radio -1 sobre Radio 0	0.200

Spacing	Radio 0	0.033448634509513
	Radio -1	0.029286564458815

Hypervolume	Radio 0	81.90350187242700
	Radio -1	82.61458604720970

Maximum Spread	Radio 0	11.82453417750510
	Radio -1	12.18599244667900

Aunque las diferencias no son muy elevadas en ninguno de los indicadores, el radio -1 es superior en todos ellos, por lo que para los siguientes experimentos se utilizará el valor de radio -1.

6.3.2. Experimento 2

Una vez que se ha determinado que el radio -1 es mejor, se va a estudiar qué ocurre con el tamaño de población, comprobando entre los tamaños 100, 200 y 500 cuál de ellos es el mejor. La tabla de variables es la siguiente:

Número de variables		3
Valores máximos de las variables	Masa	6340
	Frecuencia	10000
	Factor de amortiguamiento crítico	0.5
Valores mínimos de las variables	Masa	0.001
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tipo de variable	Masa	0
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tamaño de población		100, 200 ó 500
Número de evaluaciones		300000
Número de ejecuciones		5
Probabilidad de cruce		90
Probabilidad de mutación		10
Cruce	1: BLX	1
	2: PNX	0
	3: SBX	0
Radio de nicho		-1

Tabla 6.12 Variables 5P-1TMD del experimento 2

Las gráficas obtenidas son:

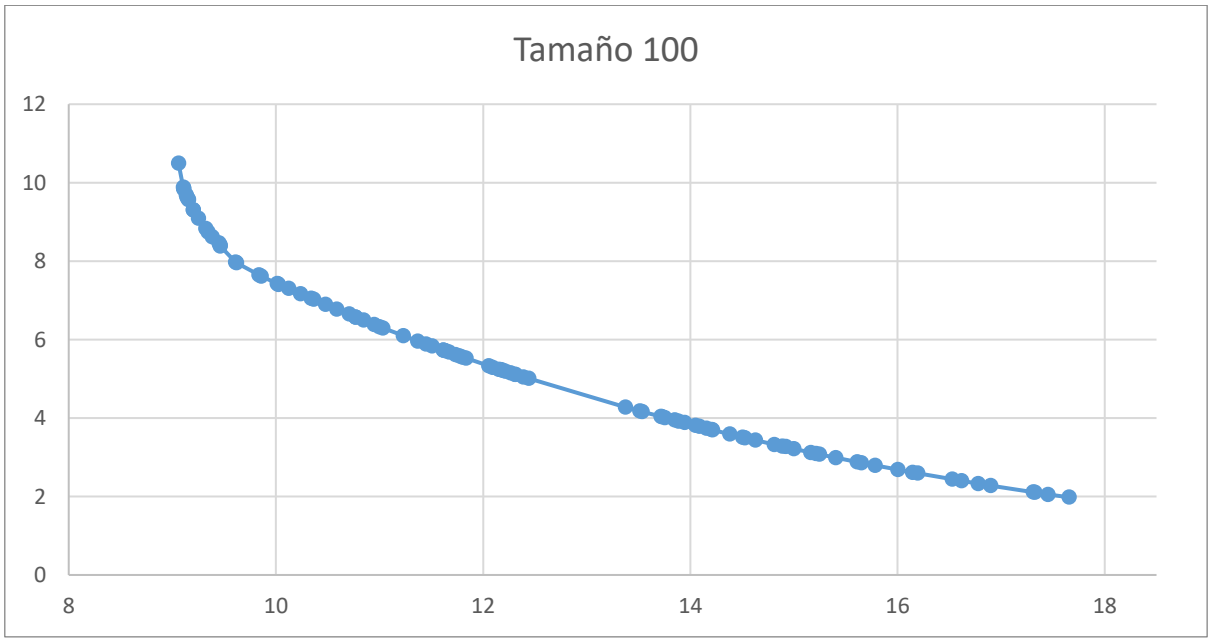

Figura 6.32 Frontera de Pareto de Tamaño 100 5P-1TMD

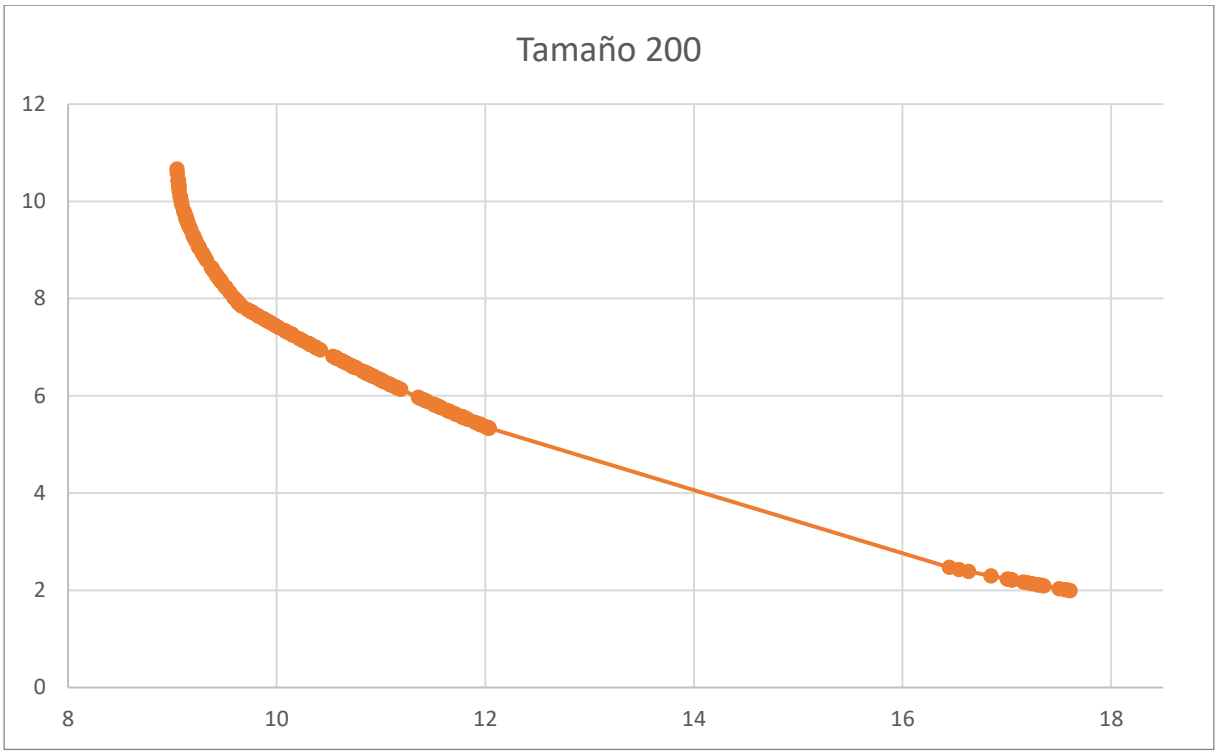

Figura 6.33 Frontera de Pareto de Tamaño 200 5P-1TMD

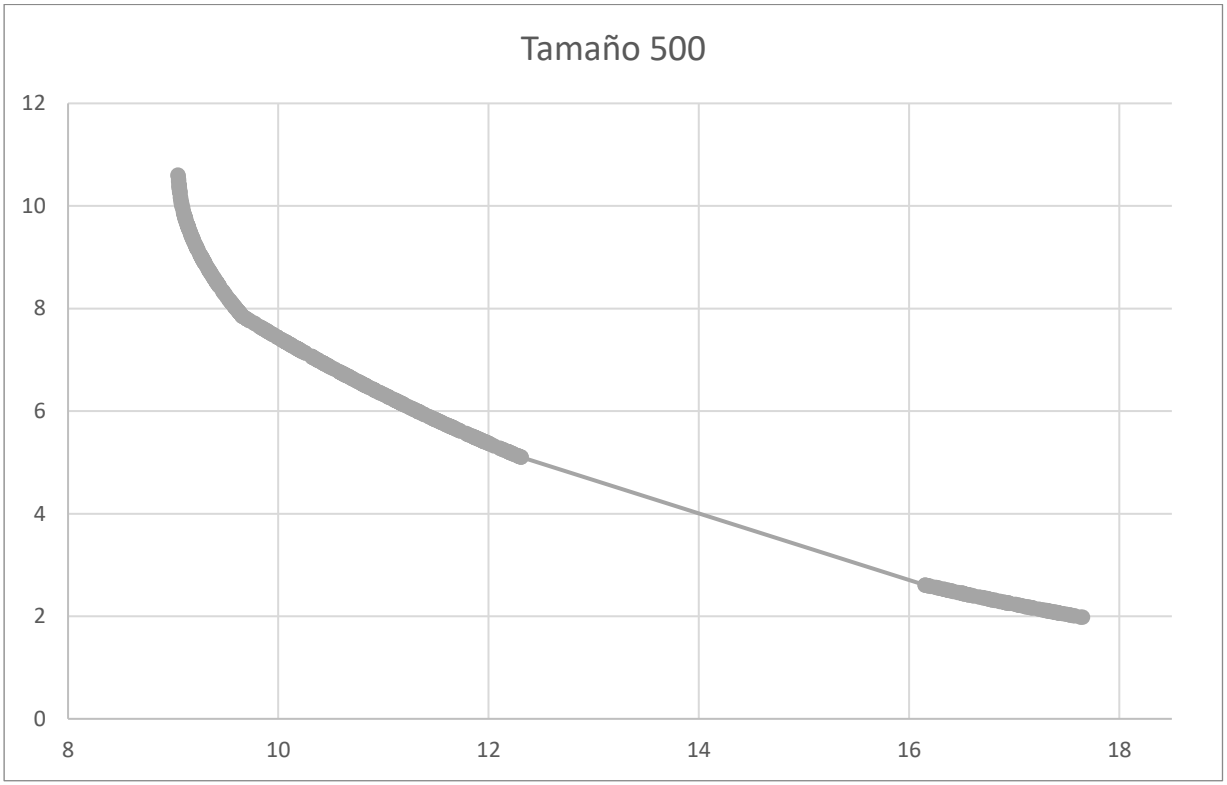

Figura 6.34 Frontera de Pareto de Tamaño 500 5P-1TMD

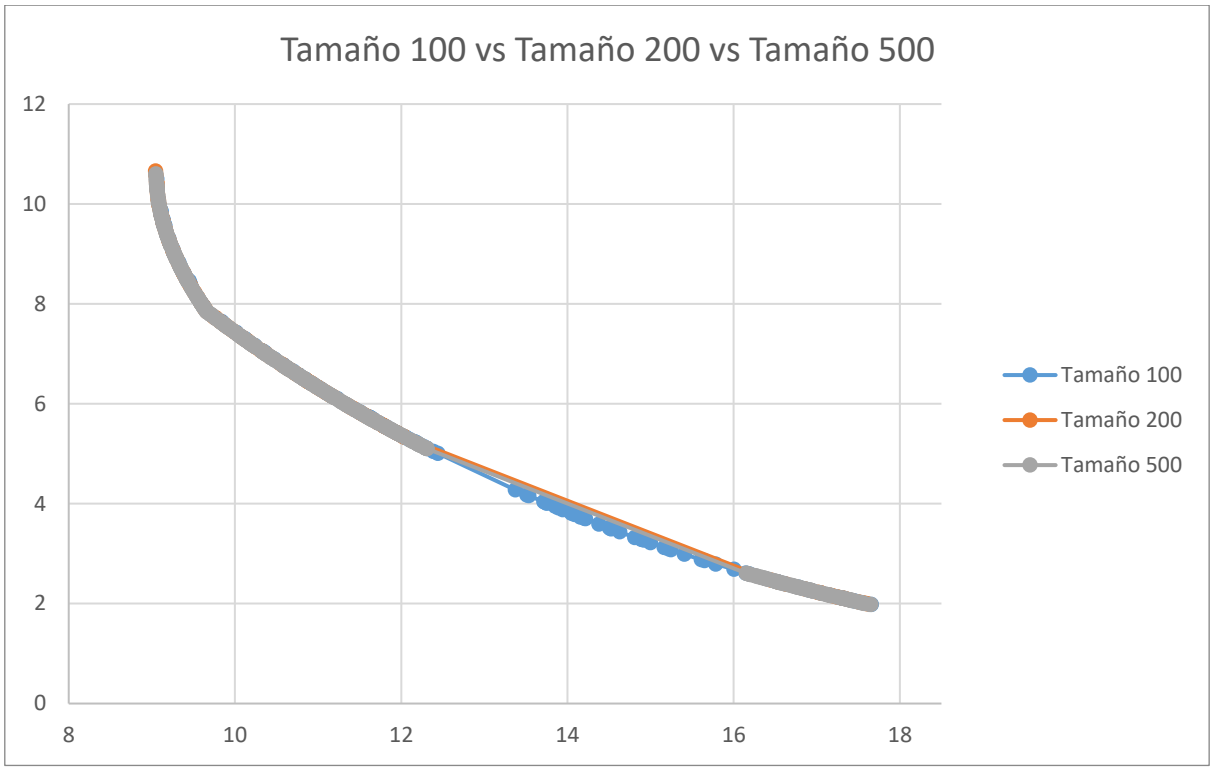

Figura 6.35 Comparación de los tamaños de población 5P-1TMD

Aplicando los criterios de comparación, los resultados son:

Set Coverage Metric	Tamaño 100 sobre tamaño 200	0.030
	Tamaño 200 sobre tamaño 100	0.410
	Tamaño 100 sobre tamaño 500	0.014
	Tamaño 500 sobre tamaño 100	0.300
	Tamaño 200 sobre tamaño 500	0.044
	Tamaño 500 sobre tamaño 200	0.000

Spacing	Tamaño 100	0.091186835870782
	Tamaño 200	0.029286564458815
	Tamaño 500	0.011334538600854

Hypervolume	Tamaño 100	87.6068742710049
	Tamaño 200	82.614586047210
	Tamaño 500	84.263329372062

Maximum Spread	Tamaño 100	12.1010237807874
	Tamaño 200	12.1859924466790
	Tamaño 500	12.1794587070649

Siguiendo los mismos criterios que hasta el momento, en el primer indicador el mejor tamaño de población es el 200. Para la segunda métrica, el mejor tamaño es el 500, aunque con poca diferencia respecto al tamaño 200. En el tercer criterio el mejor tamaño es el 100 y, en el último, las diferencias son tan pequeñas que no nos sirve como factor diferenciador.

Teniendo en cuenta que en el primer indicador el mejor tamaño de población es el 200, que en el segundo indicador es, apenas, un poco peor que el tamaño 500 y, aunque en el tercer indicador es el peor de los tres (sin existir, tampoco, grandes diferencias), se selecciona el tamaño 200 como el mejor.

Cabe destacar, también, que el tiempo de computación con el tamaño 500 se incrementa mucho frente a los otros dos. Como, además, la mejora que se obtiene al aumentar el tamaño de población no es notable, no compensa emplear el tamaño 500.

6.3.3. Experimento 3

Para terminar, se va a realizar el estudio sobre los cruces, comparando el cruce 1 (BLX) con el 3 (SBX). La tabla de variables es:

Número de variables		3
Valores máximos de las variables	Masa	6340
	Frecuencia	10000
	Factor de amortiguamiento crítico	0.5
Valores mínimos de las variables	Masa	0.001
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tipo de variable	Masa	0
	Frecuencia	0
	Factor de amortiguamiento crítico	0
Tamaño de población		200
Número de evaluaciones		300000
Número de ejecuciones		5
Probabilidad de cruce		90
Probabilidad de mutación		10
Cruce	1: BLX	1 ó 0
	2: PNX	0
	3: SBX	0 ó 1
Radio de nicho		-1

Tabla 6.13 Variables 5P-1TMD del experimento 3

Una vez obtenidos los conjuntos de soluciones no dominados, se realiza la representación gráfica de las fronteras de Pareto que se adjuntan a continuación.

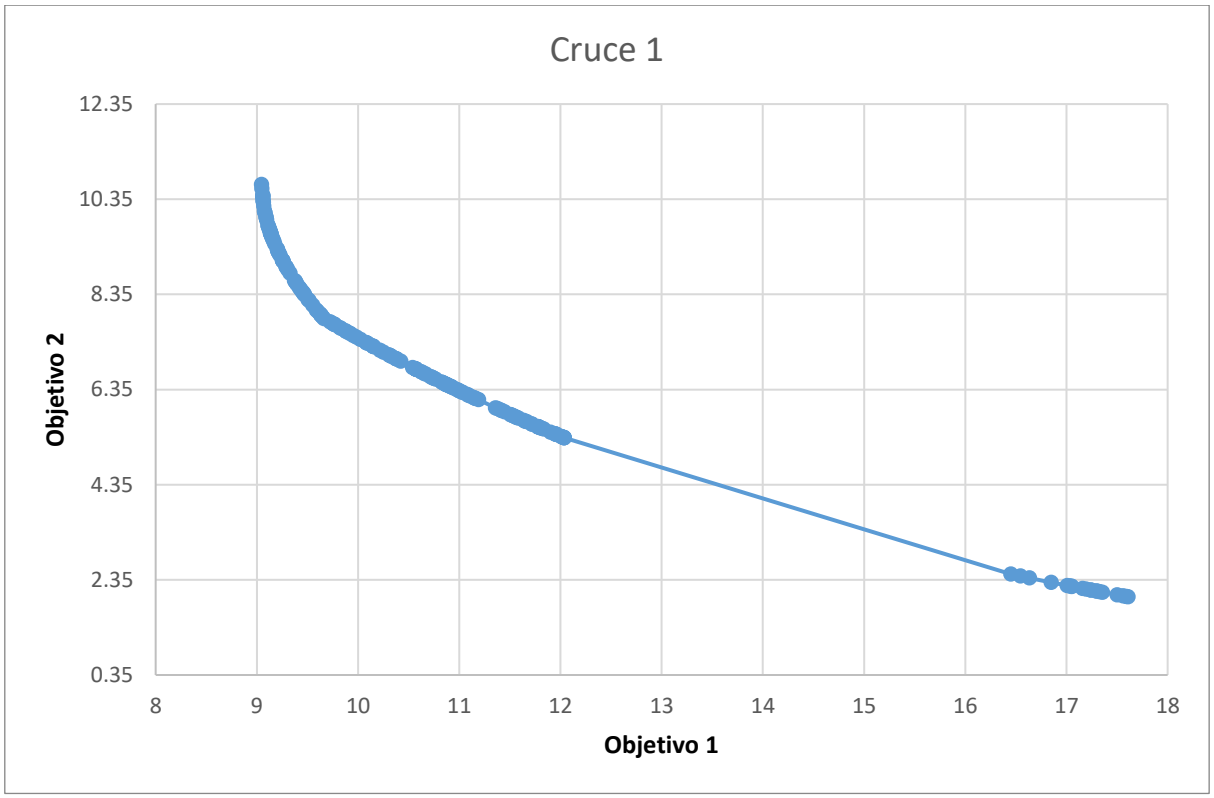

Figura 6.36 Frontera de Pareto del Cruce 1 5P-1TMD

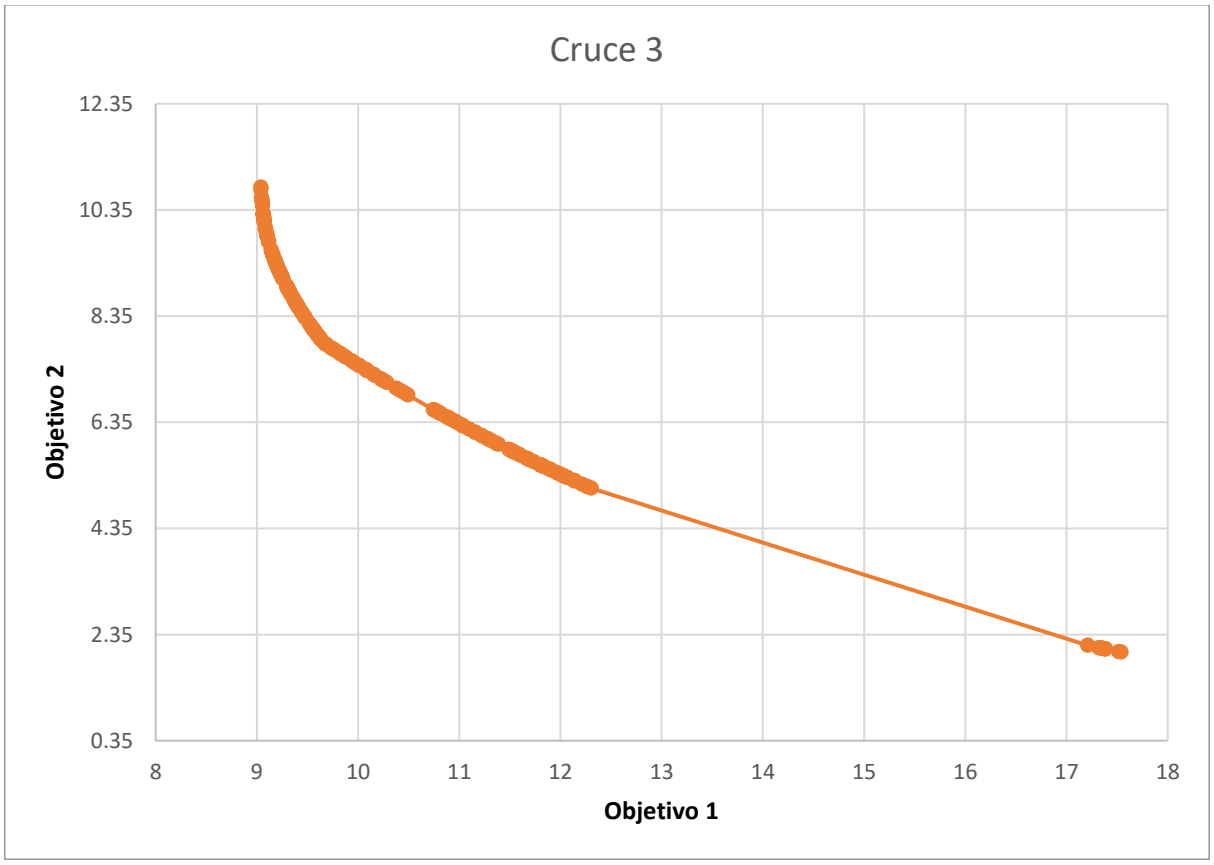

Figura 6.37 Frontera de Pareto del Cruce 3 5P-1TMD

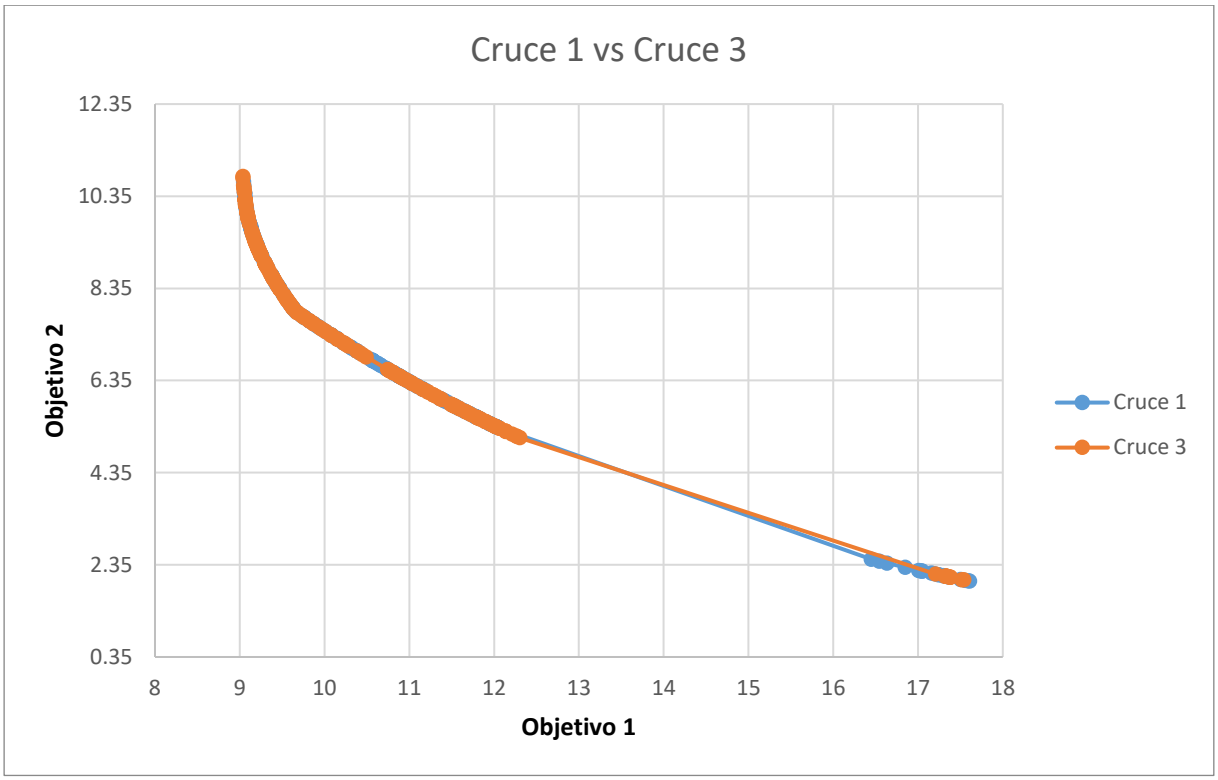

Figura 6.38 Comparación de los cruces 5P-1TMD

Aplicando las métricas de comparación de las fronteras de Pareto, los resultados obtenidos son:

Set Coverage Metric	Cruce 1 sobre Cruce 3	0.035
	Cruce 3 sobre Cruce 1	0.030

Spacing	Cruce 1	0.029286564458815
	Cruce 3	0.022314976058618

Hypervolume	Cruce 1	82.614586047210
	Cruce 3	81.940880016032

Maximum Spread	Cruce 1	12.18599244667900
	Cruce 3	12.20663083934010

En la comparación de los cruces ocurre algo similar a la de los tamaños, solo que, en este caso, todos los indicadores son muy cercanos entre sí, por lo que cualquiera de los dos cruces sería válido para un estudio posterior.

7. CONCLUSIONES

A continuación, se adjunta un cuadro resumen con los resultados obtenidos en los experimentos realizados para casa caso de estudio.

	EDIFICIO 2P-1TMD				
	Variable a comparar	Valores	Resultado obtenido	Número de ejecuciones	Tiempo de ejecución (horas)
Experimento 1	Radio	0	0	5	3
		-1			
Experimento 2	Tamaño de población	100	200		1,5
		200			3
		500			4,8
Experimento 3	Cruce	1 (BLX)	1 (BLX)		3
		3 (SBX)			
Experimento 4	Algoritmo	Original	Indiferente		3
		x100			
Experimento 5	Masa	0.16	0.16		3
		0.12			1,5
		0.08			1,5
		0.04			1,8

Tabla 7.1 Resumen de resultados para el estudio 2P-1TMD

	EDIFICIO 2P-2TMD				
	Variable a comparar	Valores	Resultado obtenido	Número de ejecuciones	Tiempo de ejecución (horas)
Experimento 1	Radio	0	-1	5	4,5
		-1			
Experimento 2	Tamaño de población	100	500		2,5
		200			4,5
		500			6
Experimento 3	Cruce	1 (BLX)	3 (SBX)		6
		3 (SBX)			
Experimento 4	Algoritmo	Original	Original (Similares)		6
		x100			

Tabla 7.2 Resumen de resultados para el estudio 2P-2TMD

	EDIFICIO 5P-1TMD				
	Variable a comparar	Valores	Resultado obtenido	Número de ejecuciones	Tiempo de ejecución (horas)
Experimento 1	Radio	0	-1	5	3
		-1			
Experimento 2	Tamaño de población	100	200		2
		200			3
		500			9,6
Experimento 3	Cruce	1 (BLX)	Indiferente		3
		3 (SBX)			

Tabla 7.3 Resumen de resultados para el estudio 5P-1TMD

En cada tabla viene indicado el experimento realizado junto con la variable sobre la que se actuaba. Además, se indica el resultado obtenido después de realizar la comparación de las fronteras de Pareto obtenidas.

Atendiendo a los resultados que se muestran, no se puede obtener una conclusión clara acerca de qué valores son mejores, puesto que, como puede verse, dependiendo del caso de estudio, los resultados son unos u otros. Entre el edificio 2P-1TMD y el 2P-2TMD, todas las variables obtenidas en los experimentos 1, 2 y 3 son distintos. Sin embargo, el criterio empleado para determinar qué variables seleccionar en los distintos experimentos ha sido el mismo en todos los casos de estudio (puede verse en el capítulo 6), de modo que la discrepancia que existe no puede achacarse a un cambio de criterio. Por otra parte, para el edificio 5P-1TMD, unos resultados son como el 2P-1TMD y otros como el 2P-2TMD.

En cuanto al experimento 5, solo se ha realizado en el edificio 2P-1TMD porque los resultados van a ser iguales para el resto de casos. Como se ha podido observar en el apartado correspondiente, a medida que se aumenta la masa del TMD, la frontera de Pareto se desplaza hacia el origen de coordenadas, lo cual implica la mejora de las dos funciones objetivo simultáneamente. Este efecto es el esperado.

Por otra parte, como dato de interés, se han añadido también los tiempos de ejecución. Dichos tiempos dependen fundamentalmente del tamaño de población, como puede observarse. El experimento 1, en todos los casos de estudio, está realizado con un tamaño de población fijo de 200 individuos. En el experimento 2 se estudia, precisamente, el tamaño de población, por lo que en la tabla pueden verse rápidamente las diferencias existentes en los tiempos de ejecución. Para los experimentos 3 y 4, dependiendo del caso, habrá un tamaño de población u otro (procedente del resultado del experimento 2), y los tiempos vendrán en función del mismo.

Se advierte también que en función del caso de estudio los tiempos se ven afectados. El edificio 2P-2TMD requiere mayor tiempo de computación que el 2P-1TMD, puesto que, en el caso del primero, se tienen 8 variables, mientras que, en el segundo, tan solo hay 3 variables. En cuanto al 5P-1TMD, los tiempos de ejecución son menores puesto que el experimento se ha realizado en un ordenador remoto cuyo acceso ha sido proporcionado por el Departamento de Ingeniería de Estructuras. Aun así, para el tamaño de población 500 el tiempo de ejecución se dispara a 2 días (9,6 x 5 = 48h). El experimento 5P-1TMD requiere tiempo de computación mucho más elevados.

Los tiempos de ejecución del experimento 5 también son más reducidos debido a que, para su realización, se ha empleado el equipo remoto de mayores prestaciones indicado anteriormente.

Como conclusión general, puede decirse que el Algoritmo Genético Multiobjetivo empleado para la obtención de las fronteras de Pareto en cada experimento permite obtener conjuntos de soluciones acordes a lo esperado en los experimentos 2P-1TMD y 2P-2TMD.

No obstante, para el experimento 5P-1TMD, las fronteras de Pareto obtenidas tienen un aspecto inesperado, puesto que, al obtener el conjunto de soluciones no dominado, se pierde diversidad en la parte central del frente de Pareto. Esto hace que, para la selección de una solución, haya que priorizar un objetivo sobre el otro, puesto que las soluciones están concentradas en los extremos de la curva.

Las modificaciones realizadas sobre los parámetros de entrada del algoritmo no generan, en ningún caso, diferencias muy grandes (las fronteras de Pareto son muy cercanas unas a otras) y la elección de un valor u otro se ha llevado a cabo en base a métodos de comparación que aportaban algo más de información (diversidad del conjunto de soluciones e "hipervolumen" del espacio de soluciones cubierto por cada conjunto fundamentalmente).

Respecto a esto último, se puede decir que el algoritmo SPEA2 se comporta de una forma bastante robusta puesto que, independientemente de los valores de los parámetros sobre los que se ha actuado (dentro de lo recomendado en la literatura), el algoritmo obtiene buenos resultados.

Como conclusión final, es importante destacar la importancia de obtener la Frontera de Pareto. En ella están recogidas todas las soluciones válidas a los problemas planteados. Sin embargo, en función de la posición del frente en la que nos encontremos, la solución seleccionada será mejor desde el punto de vista de un objetivo frente al otro.

Por ejemplo, si estamos a la izquierda y arriba, estamos priorizando el objetivo 2 (Estado Límite de Servicio, ELS), frente al objetivo 1 (Estado Límite Último, ELU). A medida que nos desplazamos hacia la derecha y hacia abajo en la frontera de Pareto, el proceso es inverso: se mejora el ELU y se empeora el ELS.

Para la selección de la solución "definitiva" es necesario tener información más avanzada, puesto que todas las soluciones son válidas, pero puede haber aplicaciones en las que uno de los objetivos prime sobre los demás.

8. BIBLIOGRAFÍA

Antón, L. (2012). *Resolución de un nuevo modelo biobjetivo para la localización de un centro semi-repulsivo mediante algoritmos evolutivos*. Murcia: Universidad de Murcia.

Bagchi, T. P. (1999). *Multiobjective Scheduling By Genetic Algorithms*. Kluwer Academic Publishers.

Beasley, D., Bull, D. R., Martin, R. R. (1993). *An Overview of Genetic Algorithms: Part 1, Fundamentals*.

Bermúdez, C., Minetti, G., Salto, C. (2016). *Búsqueda local iterada para resolver problemas de planificación*. Departamento de Informática, Facultad de Ingeniería. Universidad Nacional de La Pampa.

Cohon, J. L. (1985). Multicriteria programming: Brief review and application. J. S. Gero (Ed.), *Design Optimization*. New York: Academic Press.

Deb, K. (1989). Genetic Algorithms in Multi-Modal Function Optimization. Master's Thesis, Tuscaloosa, AL: University of Alabama.

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley. First Edition.

Deb, K., Agrawal, S., Pratap, A. and Meyarivan, T. (2000). A fast and elitist multiobjective genetic algorithm: NSGA-II. Technical Report 200001, Indian Institute of Technology, Kanpur: Kanpur Genetic Algorithms Laboratory (KanGAL).

Deb, K., Goldberg, D. E. (1989). AN investigation of niche and species formation in genetic function optimization. *Proceedings of the Third Interntional Conference on Genetic Algotithms*.

DeJong, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptative Sustems*. Ph. D. Thesis, AnnArbor, MI: University of Michigan. Dissertation Abstracts International.

Den Hartog, J. (1947). *Mechanical Vibrations*. McGraw-Hill Book Company.

Dorigo, M. (1992). Optimization, Learning and Natural Algorithms. *PhD thesis. Dipartamento Electtronica e Informazione, Politecnico di Milano, Italy*.

Feo, T., & Resende, M. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization March*, Volume 6, Issue 2.

Frahm, H. (1911). *Device for damping vibrations of bodies*.

Glover, F., Laguna, M. (1997). *Tabu Search.* Boston: Kluwer Academy.

Goldberg, D. E., Richardson, J. (1987). Genetic Algorithms with sharing for multimodal function optimization. *Proceedings of the Ninth Conference on Electronic Computations, ASCE*.

Guerra, A. (2017). *Estudio de distintos algoitmos multimodles para la sintonización óptima de TMDs múltiples en estructuras esbeltas.* Valladolid: Universidad de Valladolid.

Hwang, C.-L., Masud, A. S. M. (1979). *Multiple Objective Decision Making – Methods and Applications: A Stage-of-the-art Survey*. Berlin: Springer-Verlag.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks*, Vol. IV.

Kirkpatrick, S., Gelatt, J., & Vecchi, M. (1983). *Optimization by simulated annealing*. Science.

Magdaleno, A. (2017). *Estudio de nuevos indicadores en el dominiode la frecuencia y del tiempo para la sintonización óptima de TMDs múltiples en estructuras esbeltas*. Valladolid: Universidad de Valladolid.

Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. Boston: Kluwer.

Norvig, P., Russell, S. (2003). *Artificial Intelligence: A Modern Approach.* Prentice Hall. Second Edition.

Ormondroyd, J., & Den Hartog, J. (1928). The theory of the dynamic vibration absorber. *Transactions of the American Society of Mechanical Engineering.*

Pérez, E. (2010). *Guía para recién llegados a los algoritmos genéticos*. Valladolid: Universidad de Valladolid.

Rosenthal, R. E. (1985). Principles of multiobjective optimization. *Decision Sciences 16*(2).

Salcedo-Sanz, S., Del Ser, J., Landa-Torres, I., Gil-Lopez, S., & Portilla-Figueras, J. (2014). The Coral Reefs Optimization algorithm: a novel metaheuristic for efficiently solving optimization problems. *The Scientic World Journal*.

Santana, L. V., Coello, C. A. (2006). Una introducción a la computación y algunas de sus aplicaciones en Economía y Finanzas. *Revista de Métodos Cuantitativos para la Economía y la Empresa* (2), 3-26.

Schott, J. R. (1995). Fault Tolerant Design Using Single an Multi-Criteria Genetic Algorithms. Master's Thesis, Boston, MA: Department of Aeronautics and Astronautics, Massachusetts Institute of Technology.

Veldhuizen, D. V. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Ph. D. Thesis, Dayton, OH: Air Force Institute of Technology. Technical Report No. AFIT/DS/ENG/99-01.

Ziztler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Ph. D. Thesis, Zürich, Switzerland: Swiss Federal Institute of Technology (ETH) (Dissertation ETH No. 13398).

Ziztler, E. and Thiele, L. (1998) Multiobjective optimization using evolutionary algorithms – A comparative case study. In *Parallel Problem Solving from Nature V (PPSN-V)*.

Ziztler, E., Laumanns, M., Thiele, L. (2001). *SPEA2: Improving the Strenght Pareto Evolutionary Algorithm*. Computer Engineering and Networks Laboratoy (TIK), Department of Electrical Engineering. Zurich: Swiss Federal Institute of Technology (ETH).