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

Estudio de distintos algoritmos genéticos multimodales para la sintonización óptima de TMDs múltiples en estructuras esbeltas

Autor: D. Alejandro Guerra Diego

Tutor: D. Elena Pérez Vázquez

Valladolid, Junio de 2017

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

Estudio de distintos algoritmos genéticos multimodales para la sintonización óptima de TMDs múltiples en estructuras esbeltas

Autor: D. Alejandro Guerra Diego

Tutor: D. Elena Pérez Vázquez

Valladolid, Junio de 2017

RESUMEN

El presente Trabajo de Fin de Máster se enmarca en el estudio de procedimientos de optimización metaheurísticos, en concreto los Algoritmos Genéticos. El trabajo se centra en la aplicación de estos algoritmos a un problema multimodal real: la sintonización de TMDs. Los algoritmos genéticos están bien adaptados a la optimización multimodal y existen diferentes técnicas o procedimientos que abordan estos problemas. En este trabajo se desarrollan y estudian tres Algoritmos Genéticos Multimodales con el fin de determinar tanto la influencia sobre los resultados de sus parámetros, como cual de cada uno de ellos se comporta mejor frente al problema caso de estudio.

ABSTRACT

This Master Thesis is framed in the study of metaheuristic optimization procedures, specifically the Genetic Algorithms. The thesis focuses on the application of these algorithms to a real multimodal problem: tuning of TMDs. Genetic algorithms are well adapted to multimodal optimization and there are different techniques or procedures that address these problems. In this thesis three Multimodal Genetic Algorithms are developed and studied in order to determine the influence on the results of the parameters and which of each of them behave better.

A mis padres, Mariano y Pepi

AGRADECIMIENTOS

A mi madre, Pepi, dado que sin su esfuerzo y dedicación esto no sería posible. A mi padre, Don Mariano, de quien he aprendido a ver la vida desde un ángulo diferente. A mi hermana, Carmen, por su apoyo incondicional siempre. A mis abuelos, Encarna y Martín, por todo el cariño y sabiduría recibida.

A mis grandes amigos y amigas, por la fuente de inspiración que suponen para mí, por las grandes conversaciones y los periodos ociosos compartidos, ya que sin ellos esto no tendría sentido.

A numerosos profesores de calidad de la Universidad de Valladolid. A mi tutora Elena Pérez. A Álvaro Magdaleno y Antolín Lorenzana.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNIDICE DE TABLAS

1. INTRODUCCIÓN

Los Algoritmos Genéticos o AG son procedimientos basados en la evolución natural usados frente a problemas de búsqueda y optimización. En las últimas décadas, han experimentado un gran auge y diversificación, siendo utilizados para resolver multitud de problemas en diferentes campos, desde la economía hasta la ingeniería mecánica.

Los Algoritmos Genéticos Simples o AGS han sido utilizados para optimizar problemas unimodales de forma satisfactoria desde su aparición. Sin embargo, ha quedado patente en numerosos estudios que los AGS son vulnerables a los óptimos locales y en muchos casos no llegan a alcanzar el óptimo global del problema.

Además, gran cantidad de problemas de optimización, susceptibles de ser abordados mediante AG, tienen varios óptimos de calidad. Cuando se desea encontrar todos los óptimos de un problema concreto, los AGS no son válidos ya que el proceso evolutivo provoca la convergencia hacia un solo óptimo del problema.

Este hecho, ha motivado la aparición de los Algoritmos Genéticos Multimodales o AGMM, que mediante diversas técnicas, muchas de ellas inspiradas en la naturaleza, consiguen obtener la mayoría de óptimos del problema con buena calidad.

La optimización o sintonización de dispositivos inerciales pasivos o TMDs, de sus siglas en inglés Tuned Mass Damper, es debido a varios motivos, un problema de optimización multimodal.

Un TMD, consiste en una masa unida a una estructura mediante un elemento elástico y otro disipador de energía. Instalando uno o varios de estos dispositivos se consigue la reducción de vibraciones de una estructura.

Optimizar o sintonizar un TMD consiste en elegir de manera óptima la serie de parámetros, (masa, rigidez y amortiguamiento) que rigen su comportamiento. En el caso que se instalen varios TMDs o que la posición del TMD en la estructura se tome como variable, el problema pasa a ser inabordable de manera analítica.

Es por esto, que en los últimos años, numerosos autores han abordado el problema de sintonización de TMDs mediante metaheurísticas o procedimientos de optimización como los AG.

En este Trabajo de Fin de Máster, se trata de sintonizar los parámetros de dos TMDs, incluyendo su posición en el edificio, que serán instalados sobre una estructura de dos plantas. Esta sintonización se llevará a cabo mediante tres AGMM diferentes, que posteriormente serán comparados para determinar cual se comporta mejor frente a este tipo de problemas.

En el Capítulo 2, se expondrán las bases teóricas que rigen el comportamiento de los TMDs, tanto de forma aislada, como cuando son solidarios a una estructura. Se analizará la respuesta en frecuencia de la estructura y se definirá la función de coste a optimizar por los AGMM. En el Capítulo 3, se describirán los distintos métodos de sintonización de TMDs, tanto los analíticos como los llevados a cabo por medio de procedimientos optimización. También en este capítulo, se justifica porque es una buena idea abordar este tipo de problemas mediante métodos de optimización o procedimientos metaheurísticos como los AGMM. En el Capítulo 4, se introducirá las bases teóricas sobre las que se asientan los AG para después describir el funcionamiento de un AGS. En el Capítulo 5, se estudiarán los Algoritmos Genéticos Multimodales y se describirán los AGMM que forman parte del estudio. En el Capítulo 6, se expondrán los resultados experimentales obtenidos tras aplicar cada uno de los tres AGMM que forman parte del estudio al problema de sintonización de TMDs. Por último, en el Capítulo 7, se exponen las conclusiones y líneas futuras.

1.1. Motivación del trabajo

Como se ha comentado anteriormente, la sintonización de TMDs a escala real no puede abordarse de manera analítica. En los últimos años, son muchos los autores que tratan de llevar a cabo esta sintonización con diferentes metaheurísticas.

En este trabajo se pretende abordar el problema de sintonización de TMDs mediante AGMM ya que estos han sido usados de manera satisfactoria frente a otros tipos de problemas de ingeniería mecánica.

1.2. Objetivos y alcance

De lo anteriormente expuesto, se determinan los siguientes objetivos que definen el alcance de este Trabajo de Fin de Máster:

- Estudiar tres AGMM y aplicarles a un problema de optimización real: la sintonización de TMDs.
- Analizar las soluciones obtenidas por cada uno de los AGMM estudiados para determinar que selección de parámetros es más beneficiosa para cada algoritmo frente a este problema.
- Analizar las soluciones obtenidas por cada uno de los AGMM estudiados para determinar cual de los tres AGMM se comporta mejor frente a este problema.
- Comparar los tres AGMM estudiados frente a un AGS para comprobar la mejor calidad de las soluciones obtenidas mediante AGMM.

Como objetivos secundarios se establecen:

- Implementar cada uno de los tres AGMM en un lenguaje de alto nivel.
- Implementar un AGS en un lenguaje de alto nivel.

1.3. Medios de hardware y software utilizados

La implementación de los distintos algoritmos se ha llevado a cabo mediante el software Matlab en su versión 2015a.

Las optimizaciones se han llevado a cabo sobre un ordenador con Windows 10 Pro, con un procesador Intel Core I5-2500K a 3,3 GHz con 4 Gb de memoria RAM.

2. MARCO TEÓRICO

El dispositivo absorbedor pasivo de vibraciones conocido como TMD (de sus siglas en ingles *Tuned Mass Dumper*) fue introducido por el ingeniero naval Hermann Frahm en el año 1908 con el objetivo de desarrollar un dispositivo capaz de reducir la oscilación lateral de los barcos. Este trabajo acabó en dos patentes que se publicaron en el año 1911 (Frahm, 1911). A partir de este concepto, otros autores han desarrollado y evolucionado el dispositivo adaptándolo a otros ámbitos como la ingeniería de estructuras o la ingeniería aeronáutica.

Ya en 1928, un análisis sistemático de los TMDs, la elección correcta de sus parámetros y su influencia sobre las estructuras fue desarrollado por (Ormondroyd y Den Hartog, 1928) que fue publicada más tarde en el libro *Mechanical vibrations* (Den Hartog, 1934).

Un TMD consiste, como se detallará más adelante, en una masa unida a una estructura mediante un elemento elástico y otro disipador de energía que permite la reducción de vibraciones de una estructura debidas a diversos factores como sismos o viento.

En este capítulo, se detallarán el modelo matemático que rige el comportamiento de un TMD, se presentará el modelo de un edificio esbelto de n plantas, para más adelante detallar el modelo del conjunto de estructura + TMD. A continuación, se analizará la respuesta en frecuencia del sistema y los indicadores que pueden ser utilizados para optimizar un TMD en base a esta respuesta. Por último y una vez expuestas las bases teóricas, se describirá el modelo experimental sobre el que se realizarán las optimizaciones y se verán algunos ejemplos de TMDs reales instalados en la actualidad.

2.1. Modelo de un TMD

Un TMD se puede modelar como un sistema mecánico de un grado de libertad formado por una masa móvil (m_j) solidaria a la estructura (S) por medio de un elemento elástico (k_j) y un elemento disipador de energía (c_j) como se muestra en la Figura 2.1:

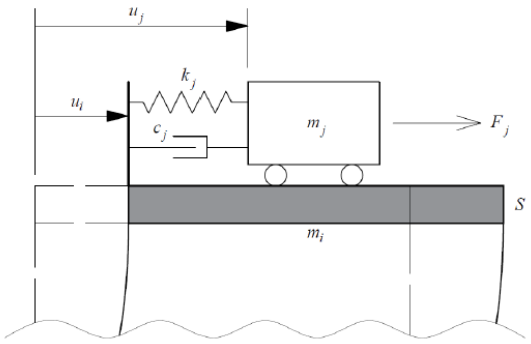

Figura 2.1: Modelo de un TMD (Magdaleno, 2017)

El TMD tiene su masa concentrada en un solo punto y se desplaza horizontalmente sobre la estructura, representado en la Figura 2.1 por el carrito. Frente a una fuerza, tanto la estructura como el TMD sufren un desplazamiento de magnitud u_i y u_j respectivamente.

Aplicando la Segunda Ley de Newton a la masa del TMD se puede obtener la ecuación del movimiento (Ecuación 2.1) que rige el comportamiento del TMD:

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

Donde F_j es una fuerza que puede ser aplicada sobre el TMD.

Del mismo modo, se puede obtener la reacción del TMD (Ecuación 2.2) sin más que aplicar la misma ley al resorte y al amortiguador (elementos sin masa):

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

Cuando se trabaja con TMDs es habitual hablar de frecuencia propia (ω_j) y factor de amortiguamiento crítico (ξ_j) en lugar de rigidez (k_j) y constante de amortiguamiento (c_j). La frecuencia propia de un sistema es aquella frecuencia ante la cual el sistema responde con amplitud máxima. El factor de amortiguamiento crítico es aquel que anula la respuesta del sistema en tiempo mínimo.

Estos parámetros están relacionados por las Ecuaciones 2.3 y 2.4:

$$\omega_j^2 = \frac{k_j}{m_j} \tag{2.3}$$

$$2\xi_j\omega_j = \frac{c_j}{m_j} \tag{2.4}$$

De esta manera, a partir de las propiedades modales (frecuencia propia y factor de amortiguamiento crítico) podemos obtener las propiedades físicas del TMD (rigidez y constante de amortiguamiento).

2.2. Modelo de una estructura de n plantas

En este TFM se trabajará con un edificio de dos plantas al que se le aplica una excitación en su base. Tanto la excitación como el movimiento del edificio se producen en un mismo plano, por lo que el movimiento que sufre el edificio es únicamente un movimiento traslacional. De esta manera, en este apartado se presenta el modelo de un edificio de n plantas que a la hora de realizar las optimizaciones será particularizado para un edificio de dos plantas.

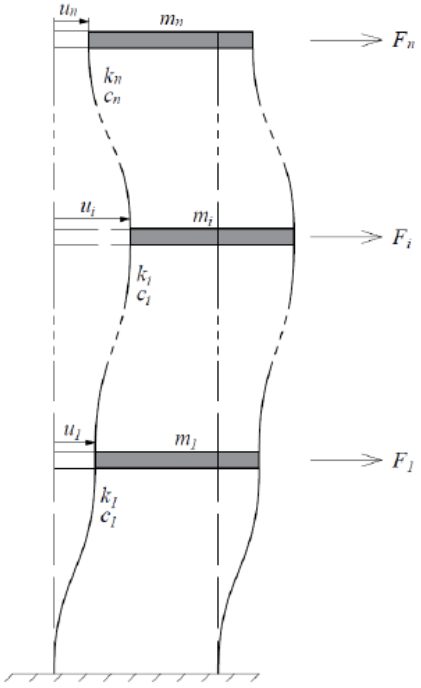

Figura 2.2: Modelo de un edificio de n plantas (Magdaleno, 2017)

En la Figura 2.2 se muestra un esquema del edificio, empotrado en el nivel inferior, donde cada piso se desplaza horizontalmente una magnitud u_i debido a una fuerza F_i. Además, la masa de cada piso se encuentra concentrada y tiene valor m_i. Entre los pisos existe una rigidez k_i y existe un amortiguamiento estructural de valor c_i.

Aplicando la Segunda Ley de Newton piso por piso se pueden obtener las ecuaciones del movimiento de cada uno de ellos, constituyendo un sistema de tantas ecuaciones como pisos tiene el edificio (Ecuación 2.5):

$$\begin{aligned}
&F_1 = m_1\ddot{u}_1 + (c_1 + c_2)\dot{u}_1 - c_2\dot{u}_2 + (k_1 + k_2)u_1 - k_2u_2 \\
&F_2 = m_2\ddot{u}_2 - c_2\dot{u}_1 + (c_2 + c_3)\dot{u}_2 - c_3\dot{u}_3 - k_2u_1 + (k_2 + k_3)u_2 - k_3u_3 \\
&\vdots \\
&F_i = m_i\ddot{u}_i - c_i\dot{u}_{i-1} + (c_i + c_{i+1})\dot{u}_i - c_{i+1}\dot{u}_{i+1} - k_iu_{i-1} + (k_i + k_{i+1})u_i - k_{i+1}u_{i+1} \\
&\vdots \\
&F_n = m_n\ddot{u}_n + c_n(\dot{u}_n + \dot{u}_{n-1}) - k_n(u_n - u_{n-1})
\end{aligned} \tag{2.5}$$

2.3. Modelo del conjunto estructura + TMD

Con el objetivo de evaluar la influencia del TMD sobre la respuesta de la estructura es necesario construir un modelo que englobe ambos elementos. Para ello, se partirá de una planta que lleva instalado un TMD y además sobre ella actúa una fuerza exterior F_i.

Aplicando la Segunda Ley de Newton al conjunto estructura + TMD para una planta determinada se obtiene la Ecuación 2.6:

$$\begin{aligned}
F_i - R_i = m_i\ddot{u}_i - c_i\dot{u}_{i-1} + (c_i + c_{i+1})\dot{u}_i - c_{i+1}\dot{u}_{i+1} - k_iu_{i-1} + (k_i + k_{i+1})u_i \\
- k_{i+1}u_{i+1}
\end{aligned} \tag{2.6}$$

Como puede verse, la ecuación obtenida es la combinación de la ecuación de la estructura para una planta genérica *i* (Ecuación 2.5) y la reacción generada por el TMD en sentido contrario a la fuerza aplicada.

Sustituyendo la expresión de la reacción R_i (Ecuación 2.2) obtenemos la Ecuación 2.7 que representa el comportamiento de una planta concreta del edificio con un TMD instalado.

$$\begin{aligned}
F_i = m_i\ddot{u}_i - c_i\dot{u}_{i-1} + \left(c_i + c_{i+1} + c_j\right)\dot{u}_i - c_{i+1}\dot{u}_{i+1} - k_iu_{i-1} + \left(k_i + k_{i+1} + k_j\right)u_i \\
- k_{i+1}u_{i+1}
\end{aligned} \tag{2.7}$$

La nueva ecuación obtenida no es más que la ecuación del movimiento de una planta del edificio más el efecto de la rigidez k_j y el amortiguamiento c_j del TMD.

A partir de las ecuaciones del movimiento para cada piso, teniendo en cuenta si tiene instalado un TMD (Ecuación 2.7) o no (Ecuación 2.5), es posible obtener un sistema de ecuaciones que puede expresarse de forma matricial en función de tres matrices: la matriz de Masa ($\mathbb{M}$), la matriz de Rigidez ($\mathbb{K}$) y la matriz de Amortiguamiento ($\mathbb{C}$).

$$\mathbb{M}\ddot{q} + \mathbb{C}\dot{q} + \mathbb{K}q = \mathbb{F} \tag{2.8}$$

Debido a que la optimización del conjunto se realizará con Matlab, es conveniente utilizar el modelo de espacio de estados, ya que este software incluye funciones capaces de simular un modelo en espacio de estados tanto en el dominio del tiempo como en el de la frecuencia.

Un modelo matemático representado en espacio de estados (Ecuación 2.8) hace uso de dos vectores, uno denominado *vector de estado* (*x*), que incluye las variables del sistema a representar, y otro *vector de entradas* (*u*). Estos vectores, están relacionados por dos matrices, la *matriz de estado* (*A*) y la *matriz de entradas* (*B*) formando las *ecuaciones de estado*. Además, el modelo posee un conjunto de *ecuaciones de salida* que permiten obtener diferentes salidas (*y*) como combinación lineal de las variables de estado y las entradas, utilizando la *matriz de salida* (*C*) y la matriz de *transmisión directa* (*D*).

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{2.9}$$

De esta manera, a partir de la representación matricial del sistema estructura + TMD (Ecuación 2.8) se obtendrá el modelo equivalente en espacio de estados del conjunto (Ecuación 2.9) listo para trabajar con Matlab.

2.4. Análisis de la respuesta en frecuencia

La respuesta en el dominio de la frecuencia ha sido utilizada por los autores que han abordado la optimización de TMDs desde su aparición. Para ello, se hace uso de la Función de Respuesta en Frecuencia o FRF de las siglas en ingles de Frequency Response Function. Esta función, es una curva característica del sistema que relaciona la frecuencia de excitación de la estructura con la relación de amplitudes respuesta/excitación en algún punto de la estructura. De forma teórica, se define la FRF como la relación entre la salida y la entrada en el dominio de la frecuencia (Ecuación 2.10):

$$H^{ib}(\omega) = \frac{X_i(\omega)}{A_b(\omega)} \tag{2.10}$$

Los máximos de esta curva, se encuentran en las frecuencias propias del sistema, donde la amplitud de salida es máxima debido al fenómeno de resonancia.

A modo de ejemplo, en la Figura 2.3 se muestra la FRF de un edificio de dos plantas.

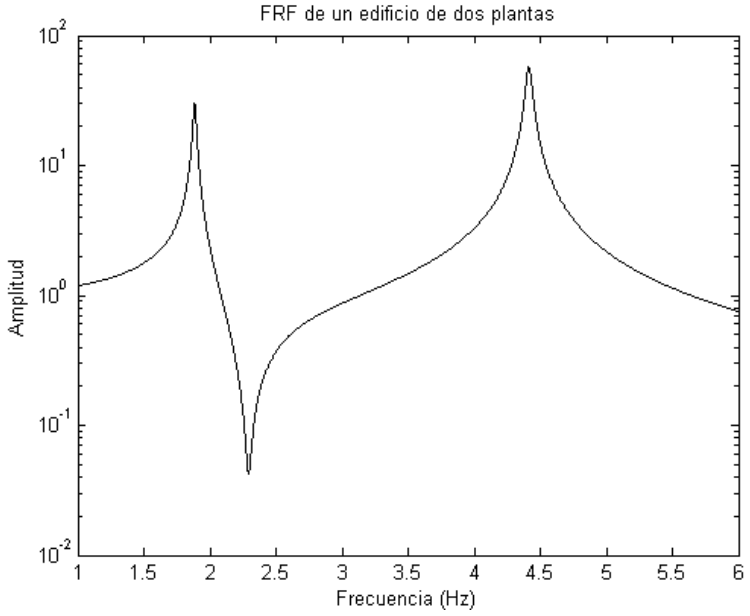

Figura 2.3: Ejemplo de FRF para un edificio de dos plantas

A partir de esta respuesta, sintonizar un TMD, es realizar una minimización de esta curva ya sea en desplazamientos (orientado a estructura) o en aceleraciones (orientado al confort de las personas).

Como puede verse en la Figura 2.4, la presencia de un TMD bien sintonizado entorno a una frecuencia propia del sistema, produce el desdoblamiento de la cresta de esa zona y además, si el ajuste de la frecuencia y amortiguamiento del TMD es óptimo, el valor de las crestas será mínimo y el mismo.

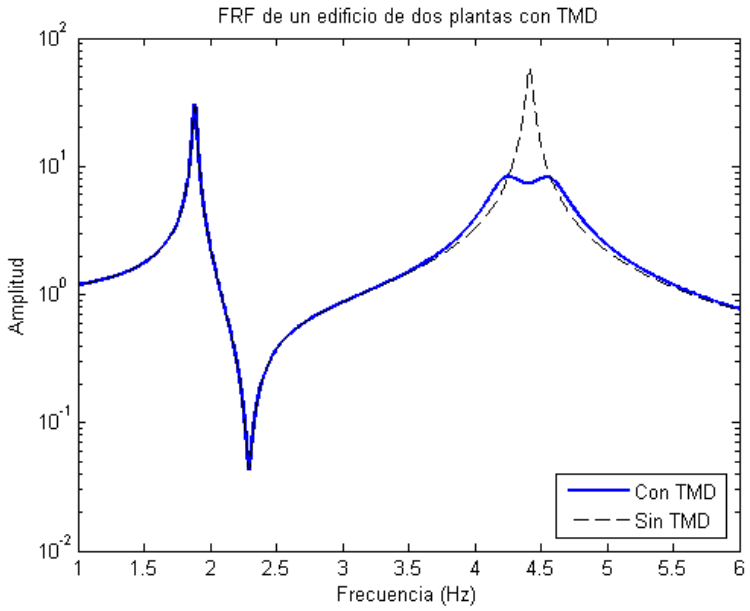

Figura 2.4: Ejemplo de efecto de un TMD sobre la FRF de un edificio de dos plantas

Del análisis realizado por (Magdaleno, 2017), existen distintos indicadores de respuesta en frecuencia, que pueden ser utilizados para confeccionar distintas funciones de coste y luego sintonizar un TMDs. Estos son:

1. *Máximo local de una FRF*: trata de minimizar el máximo de las crestas producidas por el desdoblamiento debido a la instalación de un TMD en un rango de frecuencias. Este rango, normalmente se encuentra centrado en una frecuencia propia del sistema y es de tamaño lo suficientemente grande para englobar las dos crestas. Para ello, se confeccionará una función de coste que devuelva el máximo de la FRF en un rango de frecuencias dado.

Matemáticamente, la expresión correspondiente a minimizar el máximo local de una FRF se muestra en la Ecuación 2.11, donde *x* es un vector que contiene las propiedades del TMD (masa, rigidez, amortiguamiento y posición) y ω_1, ω_2 marcan los límites del rango de frecuencias a optimizar.

$$\min_x(\max(H(\omega)))\ /\omega \in [\omega_1, \omega_2] \tag{2.11}$$

Este indicador ha sido estudiado ampliamente por Den Hartog y otros autores llegando a proponer expresiones para la sintonización óptima de un TMD sobre un sistema de un grado de libertad como ser verá en el Capítulo 3.

2. *Máximo global de una FRF*: en el caso de que las frecuencias de interés a minimizar se encuentren relativamente cerca unas de otras, puede ser interesante minimizar la respuesta global de la estructura. Para ello, se confeccionará una función de coste que devuelva el máximo de la FRF, esté donde esté.

La expresión matemática correspondiente a esta minimización se muestra en la Ecuación 2.12 y es similar a la anterior, sin más que cambiar los límites de las frecuencias a optimizar, Ω_1 y Ω_2 coincidirán con los límites de cálculo de la FRF.

$$\min_x(\max(H(\omega)))\ /\omega \in [\Omega_1, \Omega_2] \tag{2.12}$$

3. *Máximo local de todas las FRF*: este caso es similar a optimizar el máximo local de una FRF sin más que considerar las FRFs de todos los pisos de la estructura. De esta manera, se optimizará la respuesta en todos los puntos de la estructura donde se calculen las FRFs en torno a una frecuencia propia del sistema.

De forma matemática puede expresarse mediante la Ecuación 2.13, donde $H_1(\omega), \dots, H_n(\omega)$ corresponden a todas las FRFs calculadas y ω_1, ω_2 al rango de frecuencias a optimizar en torno a una frecuencia propia del sistema.

$$\min_x(\max(H_1(\omega), H_2(\omega), \dots, H_n(\omega)))\ /\omega \in [\omega_1, \omega_2] \tag{2.13}$$

4. *Máximo global de todas las FRF*: por último, puede ser deseable incluir en el indicador todas las crestas de todas las FRFs en la función de coste a minimizar. De esta manera, se minimizará la mayor cresta de todas las FRFs en todo el rango de frecuencias.

Matemáticamente, puede expresarse mediante la Ecuación 2.14 donde $H_1(\omega), \dots, H_n(\omega)$ corresponden a todas las FRFs calculadas entre los límites Ω_1 y Ω_2.

$$\min_x(\max(H_1(\omega), H_2(\omega), \dots, H_n(\omega)))\ /\omega \in [\Omega_1, \Omega_2] \tag{2.14}$$

2.5. Modelo experimental y criterio de optimización

Las optimizaciones llevadas a cabo en este trabajo se efectuarán sobre una maqueta de un edificio de dos plantas fruto de trabajos previos. Esta maqueta cuenta con dos tipos de elementos, pilares y forjados, como puede verse en la Figura 2.5.

Figura 2.5: Maqueta del edifico de dos plantas

Los pilares, están formados por placas de aluminio y aportan flexibilidad lateral unidireccional al sistema. Además, al tener un pilar a cada lado se reduce la posibilidad de torsión quedando el movimiento confinado en un solo plano.

Los forjados, están formados por placas de metacrilato y aportan rigidez vertical al sistema. Estas placas, cuentan con un espesor suficiente para que todos los desplazamientos verticales que puedan ocurrir sobre ellas sean varios ordenes de magnitud más bajos que los desplazamientos horizontales.

Como se expuso en la descripción del modelo de un edifico de n plantas, los parámetros que rigen su comportamiento son: la masa de cada planta (m_i), la rigidez entre plantas (k_i) y el amortiguamiento estructural (c_i). Estos parámetros han sido previamente identificados por (Magdaleno, 2017) en otro trabajo y se resumen en la Tabla 2.1.

$m_1 = 2.1382\ kg$	$k_1 = 1111.8\ N/m$	$\xi_1 = 0.0066\ \%$
$m_2 = 1.8756\ kg$	$k_2 = 389.05\ N/m$	$\xi_2 = 0.0052\ \%$

Tabla 2.1: Propiedades de la maqueta de 2 plantas

Una vez identificado el edificio sobre el que se realizarán las optimizaciones, queda definir que indicador se quiere optimizar para confeccionar la función de coste con la que trabajarán los algoritmos. En este trabajo se ha optado por minimizar el máximo global de todas las FRFs en aceleración, es decir el máximo global de las FRFs en aceleración de la primera y segunda planta. Este criterio queda definido formalmente mediante la Ecuación 2.15.

$$\min_x(\max(H_1(\omega), H_2(\omega)))\ /\omega \in [\Omega_1, \Omega_2] \tag{2.15}$$

Donde, $H_1(\omega)$ y $H_2(\omega)$ representan las FRFs de las plantas 1 y 2 respectivamente y Ω_1 y Ω_2 son los límites de cálculo de las FRFs.

2.6. Ejemplos de TMDs instalados en la actualidad

El uso de TMDs para mejorar la respuesta de los edificios ante solicitaciones de viento o sismos es común en la actualidad, siendo los edificios esbeltos donde estos dispositivos tienen su mayor campo de aplicación debido a la tendencia de estos edificios a sufrir desplazamientos y aceleraciones que comprometen la estructura o el bienestar de los usuarios.

A continuación, se muestran dos edificios que hacen uso de uno o varios TMDs para mejorar su respuesta con una breve descripción tanto del edificio como del TMD instalado.

Citigroup Center (EEUU): El Citigroup Center (Figura 2.6) es un rascacielos de oficinas situado en Manhattan construido en 1977 para albergar la sede del Citibank. El edificio cuenta con 59 plantas y tiene 279 metros de altura.

Cuenta con un TMD destinado a aumentar el confort de los usuarios durante los temporales de viento situado en la parte superior de la torre. Consiste en un bloque de hormigón de sección cuadrada apoyado sobre una serie de resortes y amortiguadores hidráulicos. El TMD posee un sistema de activado automático que entra en funcionamiento cuando la aceleración horizontal excede un valor concreto.

Taipei 101 (Taiwán): El Taipei 101 (Figura 2.7) es un rascacielos ubicado en Taipéi (Taiwan) inaugurado en 2004 que cuenta con 106 plantas, 5 subterraneas y 101 sobre el nivel del suelo, de ahí su denominación. Es el octavo rascacielos más alto del mundo, elevándose 508 metros desde el nivel del suelo hasta la aguja que lo corona.

Debido a que los sismos y temporales de viento son comunes en la zona, el edificio lleva instalado un TMD que consiste en una esfera de acero de 728 toneladas suspendida entre las plantas 92 y 87 mediante cables de acero y 8 amortiguadores viscosos (Figuras 2.8 y 2.9).

Figura 2.6: Citigroup Center

Figura 2.7: Taipei 101

Además, el Taipei 101 cuenta con otros dos TMDs adicionales instalados en la punta de la aguja que brindan protección contra fuertes vientos.

Figura 2.8: Ubicación del TMD Taipei 101

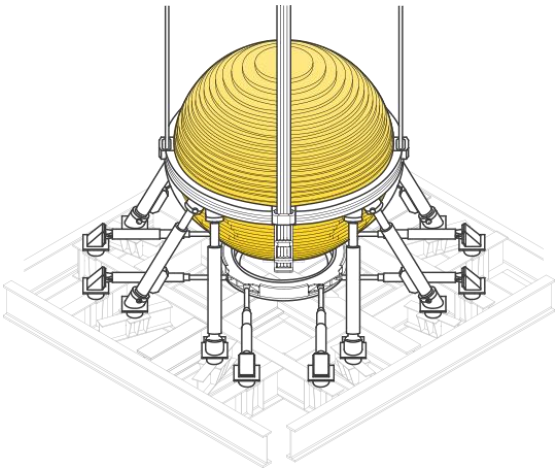

Figura 2.9: Detalle TMD Taipei 101

3. MÉTODOS DE SINTONIZACIÓN DE TMDs

Para aplicar de forma efectiva un TMD a una estructura concreta, sus parámetros deben ser optimizados o sintonizados. Los métodos de sintonización aplicados hasta ahora son diversos tanto en su procedencia como en sus resultados.

Uno de los trabajos más importantes realizados y aun presente en la actualidad, es el llevado a cabo por (Warburton, 1982) basado en los trabajos de (Den Hartog y Ormondroyd, 1928). En estos trabajos se obtienen soluciones analíticas para la sintonización óptima de TMDs frente a varios tipos de excitación. Sin embargo, estos trabajos obtienen resultados para sistemas sencillos de un solo grado de libertad y sin amortiguamiento, el único problema que posee solución analítica. Debido a esto, en las últimas décadas, múltiples autores han tratado de completar este trabajo mediante aproximaciones y correlaciones empíricas para estructuras más complejas.

Otros autores, empezaron a trabajar en la aplicación de TMDs a estructuras más complejas generalizando las expresiones usadas para la sintonización de sistemas de un grado de libertad para sistemas de múltiples grados de libertad (Sadek et al., 1997). Esta generalización afecta a los resultados y por tanto al comportamiento del TMD.

El uso de las expresiones analíticas comentadas anteriormente, se ve restringido a problemas sencillos, pero además, estas expresiones son válidas únicamente cuando se instala un solo TMD. Cuando se requiere la sintonización de varios TMDs cuyo efecto modifica el comportamiento de los demás TMDs presentes en la instalación, las formulas desarrolladas no son válidas. Este hecho, motiva la utilización de otros métodos de optimización que permitan la sintonización conjunta de varios TMDs.

Recientemente, con el ánimo de superar las limitaciones de las expresiones analíticas utilizadas hasta la fecha, se han usado distintos algoritmos de optimización que permiten trabajar conjuntamente sobre varios TMDs. Estos algoritmos, son generalmente metaheurísticas, la mayoría inspiradas en la naturaleza que simulan distintos procesos biológicos.

La evolución natural ha inspirado los Algoritmos Genéticos (GA) que han sido utilizados en diferentes ocasiones frente a varios problemas de ingeniería mecánica, incluyendo la optimización de TMDs (Hadi y Arfiadi, 1998) y otros problemas de optimización de estructuras (Rajeev y Krishnamoorthy, 1992). También, la inteligencia colectiva o inteligencia de enjambre ha inspirado otros algoritmos usados con frecuencia en optimización. Por ejemplo, los algoritmos de optimización basados en Enjambres de Partículas (PSO) (Kaveh y Talatahari, 2009) y en el comportamiento de las Colonias de Hormigas (ACO) (Farshidianfar y Soheili, 2013), han sido utilizados en ingeniería mecánica para sintonizar TMDs. Recientemente, se ha modelado el comportamiento de los Arrecifes de Coral (CRO) y aplicado a la optimización de TMDs (Salcedo-Sanz et al., 2014).

En menor medida, las metaheurísticas basadas en trayectorias han sido utilizadas en ingeniería mecánica y optimización de estructuras. La Búsqueda Tabú (TS) (Bennage y Dhingra, 1995) y el Recocido Simulado (SA) (Shim y Manoochehri, 1997) se han utilizado para optimizar las uniones de estructuras en celosía.

Además, han aparecido nuevas metaheurísticas que están siendo aplicadas a distintos problemas de ingeniería mecánica como el algoritmo Big-Bang Big-Crunch (Kaveh y Mahdavi, 2013), la Optimización por Colisión de Cuerpos (Kaveh y Mahdavi, 2014), o la Búsqueda de Sistemas de Carga (Kaveh y Talatahari, 2014).

Los últimos trabajos realizados emplean algoritmos de optimización multiobjetivo que incluyen parámetros como las carreras de los TMDs o el coste de estos. (Greco et al., 2016).

En este capítulo se repasarán algunos de los métodos de sintonización de TMDs. Se comenzará con el método analítico más sencillo para un grado de libertad y su generalización a varios grados de libertad. A continuación, se detallarán algunos métodos de optimización metaheurísticos utilizados frente a problemas de ingeniería mecánica. Por último, se expondrá una recopilación de trabajos de sintonización de TMDs realizados con algoritmos genéticos.

3.1. Sintonización analítica del TMD

Como se ha comentado anteriormente, muchos autores han tratado de obtener expresiones analíticas y experimentales que permitan el diseño óptimo de TMDs. Uno de los trabajos más populares vigente en la actualidad es el desarrollado por (Den Hartog y Ormondroyd, 1928) y ampliado posteriormente por (Warburton, 1982). En él se proporcionan expresiones muy sencillas para optimizar en frecuencia y amortiguamiento un TMD para un sistema de un grado de libertad a partir, del tipo de excitación y de la relación de masas μ entre la masa del TMD y la masa del sistema.

Estas expresiones analíticas son válidas para sistemas de varios grados de libertad que han sido reducidos a un sistema de un grado de libertad que responde igual a como lo hace la estructura original en alguno de sus grados de libertad frente a una excitación armónica resonante.

Para sintonizar un TMD instalado en un sistema de varios grados de libertad, se obtienen los parámetros equivalentes (masa M, rigidez K y constante de amortiguamiento viscoso C) a un sistema de un solo grado de libertad haciendo coincidir la frecuencia propia ω_0 y el factor de amortiguamiento crítico ξ_0 del modelo de un grado de libertad con la del modo al que resuena el sistema de varios grados de libertad.

Una vez obtenidos los parámetros equivalentes, en base a la masa del TMD y la masa del modelo equivalente será posible sintonizar el TMD para la frecuencia propia establecida mediante la aplicación de los resultados de Den Hartog, que para un sistema excitado en su base y minimizando la respuesta en aceleración, se reducen a las Ecuaciones 2.1 y 2.2:

$$f_{opt} = \frac{1}{1+\mu}\left(\sqrt{\frac{2-\mu}{2}}\right) \tag{2.1}$$

$$\xi_{opt} = \sqrt{\frac{3\mu}{8(1+\mu)}}\left(\sqrt{\frac{2}{2-\mu}}\right) \tag{2.2}$$

Donde, como se apuntó anteriormente, la relación entre masas del TMD y sistema μ corresponde a la Ecuación 2.3:

$$\mu = {m_{TMD}}/{m_{sistema}} \tag{2.3}$$

De las fórmulas anteriormente expuestas, se obtiene la relación enre la frecuencia propia del TMD y la del sistema original (f_{opt}) en tanto por uno y el amortiguamiento crítico también en tanto por uno.

3.2. Métodos metaheurísticos de optimización

El término metaheurística fue introducido por (Glover, 1977) para designar todos los métodos que integran procedimientos y estrategias de alto nivel para la mejora local de soluciones, evitar el estancamiento en óptimos locales, y de esta manera, realizar una exploración eficiente de todo el espacio de búsqueda.

Los métodos metaheurísticos proporcionan unos pasos que el usuario debe seguir para resolver el problema. Además, estos métodos disponen de una serie de parámetros para ajustar su aplicación a un problema concreto, así como espacios donde el usuario puede insertar sus propuestas.

Las ventajas e inconvenientes que presentan los procedimientos metaheurísticos aparecen recogidos en la siguiente tabla:

Ventajas	Inconvenientes
Algoritmos de propósito general	Algoritmos aproximados, no exactos
Rapidez de ejecución de los algoritmos con buenos resultados	Algoritmos altamente probabilísticos (no determinísticos)
Fácilmente implementables, flexibilidad ante cambios en el algoritmo.	Presentan poca base teórica

Tabla 2.2: Ventajas e inconvenientes de los procedimientos metaheurísticos

3.2.1. Definición de metaheurística

Una metaheurística (Aardal et al., 2007) es un conjunto de conceptos algorítmicos que pueden ser usados para definir métodos heurísticos aplicables a un conjunto de problemas diferentes. En otras palabras, una metaheurística puede ser vista como un método heurístico general diseñado para guiar un problema heurístico específico (algoritmo búsqueda local o método constructivo) hacia regiones del espacio de búsqueda que contienen soluciones de alta calidad.

Matemáticamente, un problema de optimización se formaliza como un par (S,f) donde S representa el espacio de soluciones (o de búsqueda) del problema, mientras que f es una función denominada función objetivo, definida de la siguiente manera:

$$f: S \rightarrow R \tag{2.4}$$

De esta manera, resolver un problema de optimización, consiste en encontrar una solución i* que satisfaga la Ecuación 2.5:

$$f(i^*) \leq f(i), \quad \forall\, i \in S \tag{2.5}$$

Que un problema sea de minimización o maximización no restringe la generalidad de la definición, ya que se puede establecer una analogía entre los dos tipos de problemas:

$$máx\{f(i) | i \in S\} \equiv mín\{-f(i) | i \in S\} \tag{2.6}$$

De esta manera un método metaheurístico, trata de explorar el espacio de búsqueda para obtener una solución *i*, identificando las regiones del espacio que contienen soluciones de alta calidad y tratando de consumir el menor tiempo en regiones del espacio menos prometedoras.

3.2.2. Clasificación de las metaheurísticas

Existen diversas clasificaciones dependiendo de las características que usemos para clasificar las metaheurísticas. Por ejemplo, podemos tener metaheurísticas basadas en la naturaleza o no basadas en ella, con memoria o sin ella, con una o varias estructuras de vecindario, etc.

Una clasificación más formal es la que se obtiene a partir del proceso de búsqueda. De esta manera podemos clasificar las metaheurísticas en dos grupos: las metaheurísticas basadas en trayectorias y las metaheurísticas basadas en poblaciones. En las metaheurísticas basadas en trayectorias, la búsqueda de la solución se realiza manipulando un solo elemento del espacio de búsqueda en cada paso, mientras que en las metaheurísticas basadas en poblaciones se manipula un conjunto de soluciones potenciales en cada paso.

Dentro de las metaheurísticas basadas en poblaciones, tienen especial importancia las inspiradas en la naturaleza o bio-inspiradas. Fundamentalmente, se basan en dos fenómenos observados en la naturaleza: la inteligencia colectiva y la evolución de las especies.

De acuerdo a lo expuesto anteriormente, en el siguiente esquema se muestran las principales metaheurísticas que serán detalladas en este capítulo:

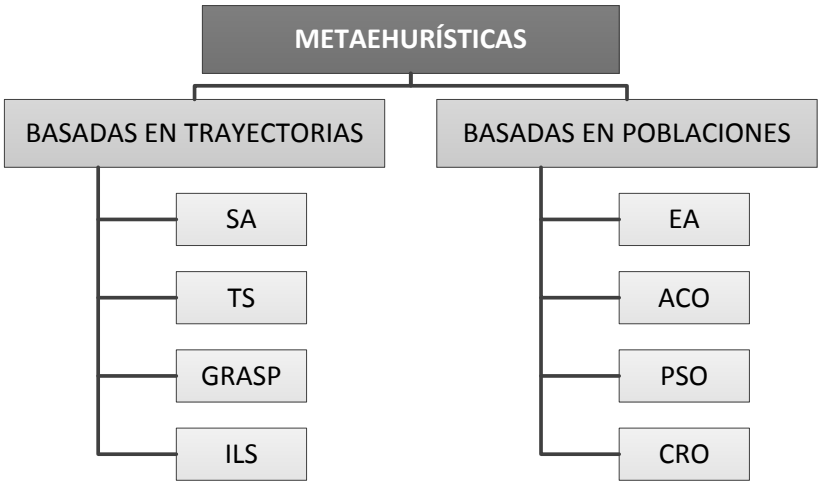

Tabla 2.3: Clasificación de las metaheurísticas

Donde las siglas:

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 2.4: Siglas de las metaheurísticas analizadas

A continuación, se analizará brevemente el funcionamiento de las metaheurísticas expuestas en el cuadro anterior. Para cada una, se da una breve descripción y un diagrama de flujo básico que representa el proceso que lleva a cabo el algoritmo.

3.2.2.1.Recocido Simulado

El Recocido Simulado (Simulated Annealing o SA) es un método de optimización inspirado en el proceso de recocido de los metales. Este proceso, implica el calentamiento del metal a altas temperaturas para luego dejarlo enfriar gradualmente hasta que alcance un estado de mínima energía. En primer lugar, el metal es calentado hasta una temperatura determinada elevando la energía de los átomos. Esta energía permite la movilidad de los átomos que se acomodan en estados de mínima energía. A continuación, en el proceso de enfriamiento, se permite a los átomos recristalizar en mejores configuraciones con menor energía que la inicial.

Este proceso, fue modelado por primera vez por (Metropolis et al., 1953), sin embargo, (Kirkpatrick et al., 1983) fueron los pioneros en aplicarlo a un problema de optimización para encontrar soluciones en el diseño de circuitos impresos.

El algoritmo consiste en un método iterativo que se inicia con un cierto estado s. Mediante un proceso, se genera un estado vecino s'. Si la energía (fitness) de s' es menor que la del estado s, se cambia el estado s por s'. Si la energía de s' es mayor que la de s, se puede empeorar el estado eligiendo s' con una probabilidad que depende de la diferencia entre las energías de los estados s y s' y de la temperatura del sistema. Esta posibilidad de elegir un estado peor del actual permite al algoritmo salir de óptimos locales y así alcanzar el óptimo global.

El diagrama de flujo del algoritmo descrito anteriormente es el siguiente:

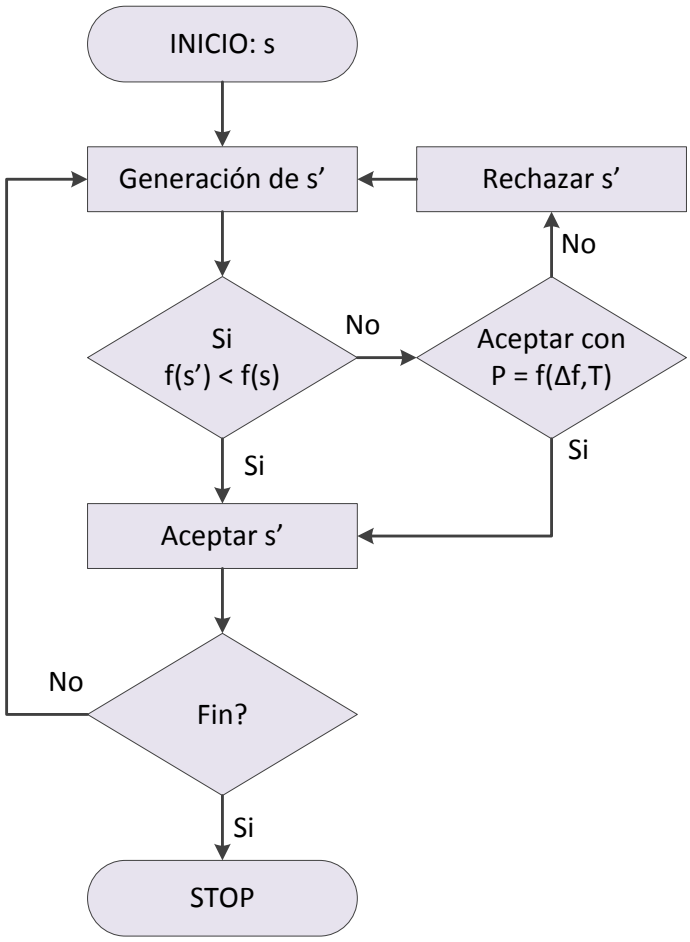

Figura 2.10: Diagrama de flujo del algoritmo Recocido Simulado

3.2.2.2.Búsqueda Tabú

La Búsqueda Tabú (Tabu Search o TS) es un método de optimización propuesto por (Glover, 1986) que explora la vecindad encontrando óptimos locales.

El procedimiento comienza con una solución inicial, y busca en el vecindario de esta solución una que presente mejor rendimiento. Cuando esta solución aparece, la búsqueda se mueve a la vecindad de la nueva solución y se repite el proceso de forma iterativa hasta que alguna condición de parada se satisfaga.

Una de las claves para que este algoritmo funcione correctamente es implementar mecanismos que impidan que la búsqueda se quede atrapada en un óptimo local. De esta manera se introducen en el algoritmo algunos movimientos prohibidos que no podrán aplicarse en un momento dado.

La búsqueda tabú (Figura 2.2) considera prohibidos todos los movimientos hacia configuraciones que contengan atributos seleccionados en el pasado reciente, de esta manera todas las configuraciones que posean alguno de los atributos prohibidos (tabú) son excluidas de la formación de la nueva vecindad. Así el algoritmo evita volver a configuraciones ya visitadas y ampliar la búsqueda del algoritmo. De esta manera, la búsqueda tabú hace uso de un historial de búsqueda o lista tabú para seleccionar el mejor movimiento en cada iteración.

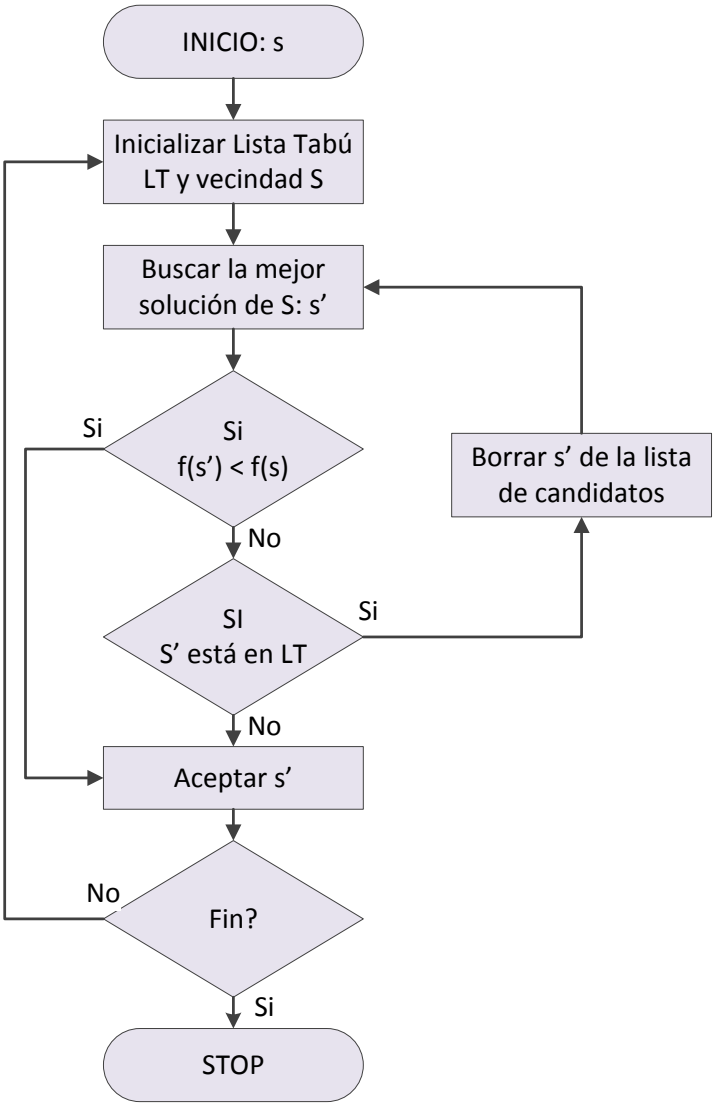

Figura 2.11: Diagrama de flujo del algoritmo Búsqueda Tabú

3.2.2.3.Algoritmos voraces o GRASP

El termino GRASP procede de las siglas en ingles “Greedy Randomized Adaptative Search Procedure” o Procedimientos de Búsqueda basados en funciones Greedy Aleatorizadas Adaptativas. Esta metodología fue propuesta por (Feo y Resende, 1995) para resolver el problema del conjunto de cobertura.

Se trata de un método de optimización basado en un procedimiento iterativo donde cada paso consiste en dos fases: una de construcción y una de mejora.

En la fase de construcción, se construye iterativamente una solución posible añadiendo un elemento en cada paso. En cada iteración, la elección del próximo elemento para ser añadido a la solución parcial viene determinado por una función greedy. Dicha función mide el beneficio de añadir cada uno de los elementos.

Para evitar el estancamiento y explorar todo el espacio de búsqueda, no se selecciona el mejor candidato según la función greedy, si no que se construye una lista con los mejores candidatos y se selecciona uno aleatoriamente.

En la fase de mejora, se aplica un algoritmo de búsqueda local con el objetivo de mejorar la solución construida en la primera fase. (Ver Figura 2.3)

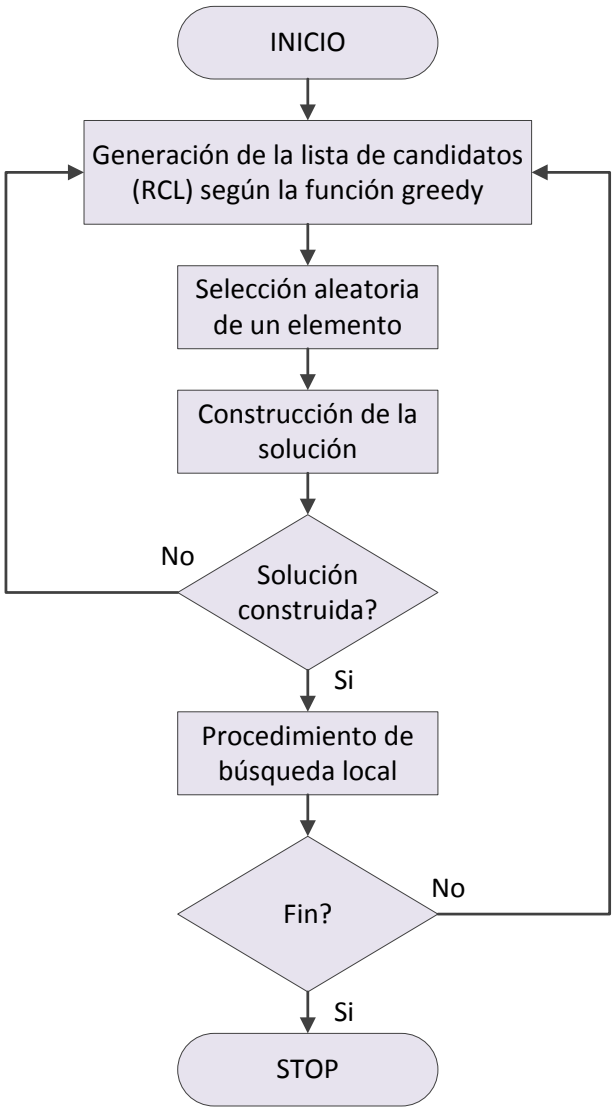

Figura 2.12: Diagrama de flujo del algoritmo GRASP

3.2.2.4.Búsqueda local iterada

La búsqueda local iterada (Iterated Local Search o ILS) es una modificación de la búsqueda local empleada para resolver problemas de optimización. Los algoritmos de búsqueda local suelen quedarse atascados en óptimos locales y no llegar al óptimo global del problema. Para evitar esto, la búsqueda local iterada implementa una modificación que consiste en iterar distintas búsquedas locales que parten de diferentes configuraciones iniciales.

Las configuraciones iniciales de partida, son generadas a partir de los óptimos locales obtenidos anteriormente. De esta manera, el algoritmo tiene en cuenta la historia anterior y es capaz de encontrar cada vez mejores soluciones.

De esta manera, se transforma una solución cualquiera de partida s en otra s* que es un óptimo local. Para conseguir salir del óptimo local y alcanzar el óptimo global, se provoca una perturbación a la solución s*, lo suficientemente grande como para escapar del óptimo local pero no tan intensa como para aleatorizar la búsqueda. Con esta perturbación se obtiene una solución s' a la cual se vuelve a aplicar el algoritmo de búsqueda para alcanzar un nuevo óptimo local s*'. El nuevo óptimo local s*' es aceptado en lugar de s' en función de algún criterio dependiendo de cuál sea el objetivo del algoritmo. (Ver Figura 2.4)

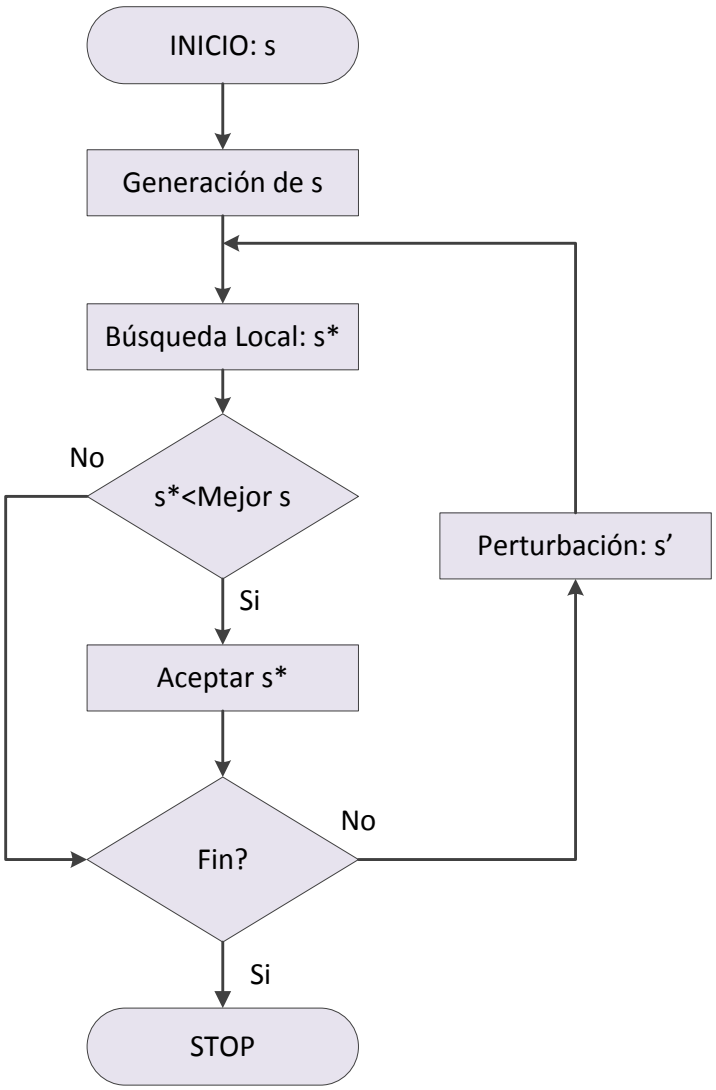

Figura 2.13: Diagrama de flujo del algoritmo de Búsqueda Local Iterada

3.2.2.5.Algoritmos evolutivos

Los Algoritmos Evolutivos (Evolutionary Algorithms o EA) son métodos de optimización basados en evolución natural y constituyen un enfoque alternativo para resolver problemas complejos de optimización mediante modelos computacionales de procesos evolutivos.

Estos algoritmos trabajan con una población de individuos que representan posibles soluciones al problema. Esta población es sometida a ciertas transformaciones y después a un proceso de selección que favorece a los mejores. De cada iteración surge una nueva generación de individuos que vuelve a ser transformada y seleccionada. Después de un cierto número de generaciones se espera que alguno de los individuos de la generación final esté cerca de la solución óptima. (Ver Figura 2.5)

De esta manera, se combina la búsqueda aleatoria, dada por las transformaciones de los individuos con una búsqueda dirigida, dada por los mecanismos de selección.

Dentro de los algoritmos evolutivos, encontramos los algoritmos genéticos, creados por (Holland, 1975) y desarrollados posteriormente por (Goldberg, 1989).

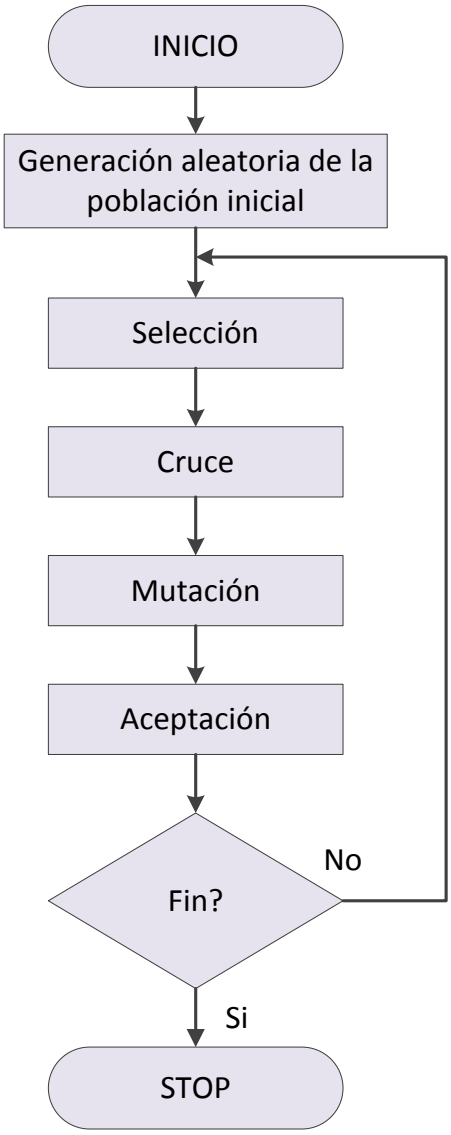

Figura 2.14: Diagrama de flujo del Algoritmo Evolutivo

3.2.2.6.Algoritmos basados en colonias de hormigas

Los Algoritmos Basados en Colonias de Hormigas (Ant Colony Optimization o ACO), propuestos por (Dorigo, 1992), están basados en el comportamiento de las hormigas cuando tratan de organizarse en comunidades y llevar comida hacia el hormiguero.

En la naturaleza, inicialmente, las hormigas vagan aleatoriamente. Una vez encontrada la comida regresan al hormiguero dejando un rastro de feromonas. Si otras hormigas encuentran este rastro, puede que dejen de caminar aleatoriamente y sigan el rastro de feromonas. Si encuentran comida, al regresar al hormiguero, las hormigas reforzaran el rastro original de feromonas.

No obstante, con el paso del tiempo, las feromonas se evaporan reduciendo la fuerza de atracción del rastro. Cuanto más tiempo tarde una hormiga en regresar al hormiguero por un rastro determinado, más tiempo tienen las feromonas para evaporarse. De esta manera, se favorecen los caminos cortos u óptimos. En estos algoritmos, la evaporación de las feromonas evita la convergencia a óptimos locales.

La idea del algoritmo es imitar el comportamiento de las hormigas con hormigas simuladas caminando a través de un espacio que representa el problema a resolver. (Ver Figura 2.6)

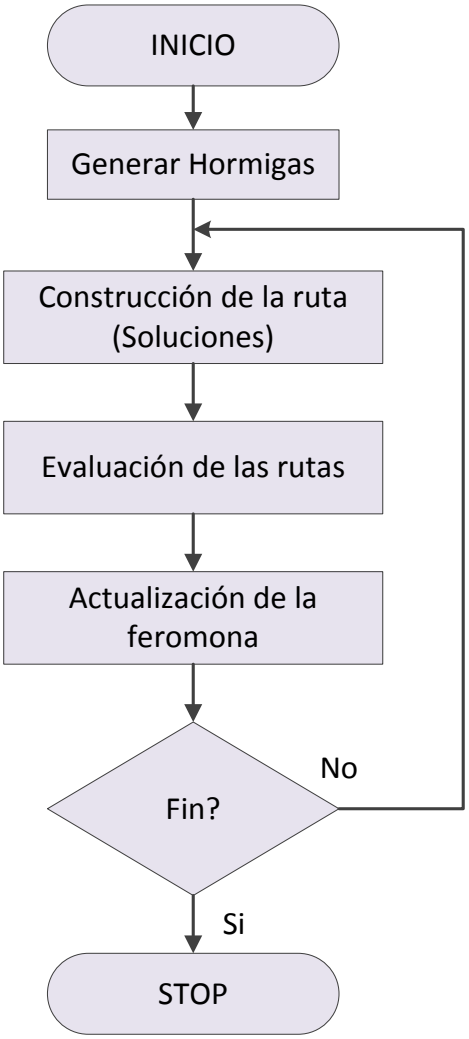

Figura 2.15: Diagrama de flujo del algoritmo basado en Colonia de Hormigas

3.2.2.7.Algoritmos basados en enjambres de partículas

Los Algoritmos Basados en Enjambres de Partículas (Particle Swarn Optimization o PSO) están basados en el comportamiento de las partículas en la naturaleza. Fueron desarrollados originalmente por (Kennedy y Eberhart, 1995) y utilizados para modelar conductas sociales, como el movimiento de un enjambre de abejas o un banco de peces. Posteriormente, el algoritmo se simplificó y fue a aplicado a problemas de optimización.

En un enjambre de abejas, cada abeja vuela de manera aleatoria recordando que región del espacio tiene más flores. Además, el enjambre sabe de manera colectiva cual es la región donde se han encontrado más flores. De esta manera, cada abeja vuela individualmente hacia algún lugar intermedio del espacio entre esas dos regiones.

El algoritmo (Figura 2.7), permite optimizar un problema partiendo de una población de soluciones posibles, denominadas partículas, que se mueven por el espacio de búsqueda según reglas matemáticas que tienen en cuenta la posición y la velocidad de las partículas.

El movimiento de la partícula, se ve influido por la mejor solución local hallada por la partícula, así como por las mejores soluciones halladas por las demás partículas. A medida que se obtienen mejores soluciones, estas orientan el movimiento de las demás partículas.

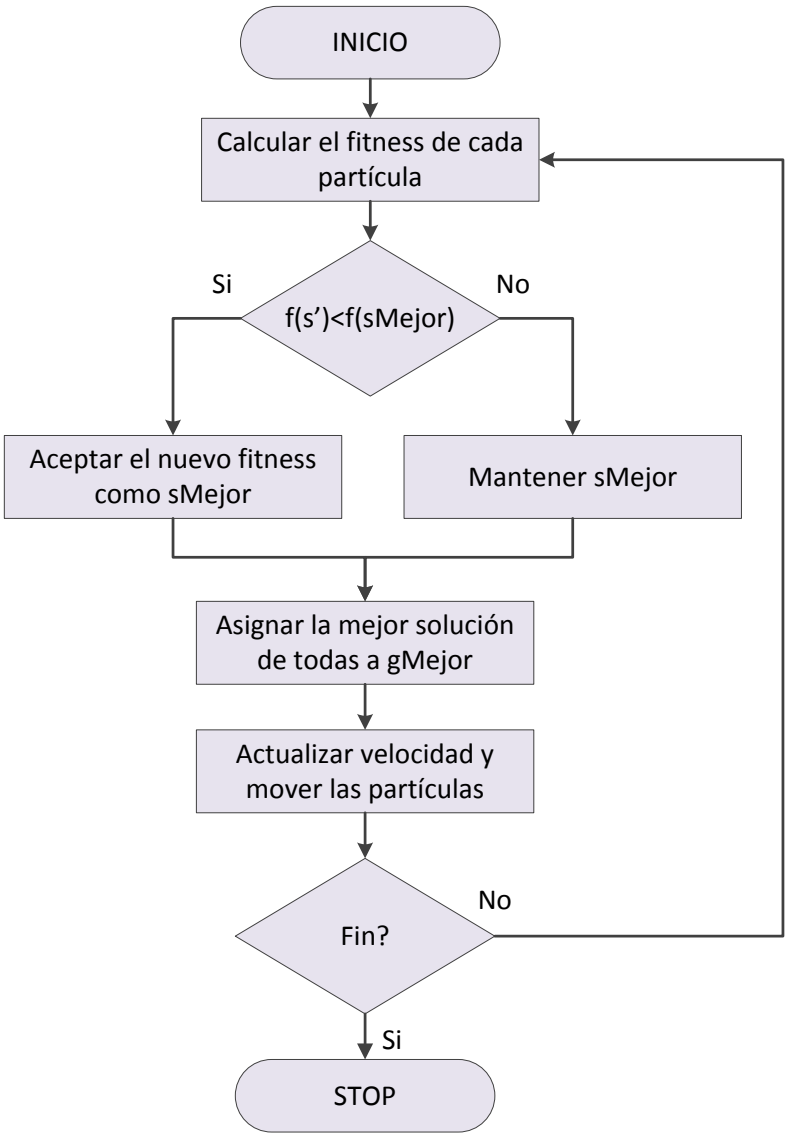

Figura 2.16: Diagrama de flujo del algoritmo basado en Enjambres de Partículas

3.2.2.8.Algoritmos basados en el crecimiento de los arrecifes de coral

Los Algoritmos Basados en el Crecimiento de los Arrecifes de Coral (Coral Reefs Optimization o CRO) son una técnica de optimización recientemente propuesta (Salcedo-Sanz, 2014) que simula la reproducción y el proceso de formación de los arrecifes de coral.

El algoritmo está basado en un arrecife de coral artificial que consiste en una matriz de nxm elementos. Cada elemento es capaz de alojar un coral o solución posible al problema. Inicialmente, se asignan aleatoriamente algunos elementos de la matriz para que sean ocupados por corales y otros se dejan vacíos simulando los agujeros donde los corales pueden asentarse y crecer en el futuro.

Cada coral, tiene asociada una propiedad llamada salud que corresponde al resultado de la función objetivo del problema o fitness. De esta manera, el algoritmo, está basado en el hecho de que el arrecife se desarrolla a partir de los corales más sanos o que presentan mejores soluciones al problema, mientras que los corales con peor salud perecen.

A la hora de modelar el comportamiento de los corales, el algoritmo, hace uso de cuatro funciones: reproducción sexual, donde una pareja de padres forma una larva; reproducción asexual, donde a partir de un coral, se forma una larva aplicándole una mutación aleatoria; asentamiento de larvas, donde las larvas tratan de asentarse en el coral, si la posición está vacía pasan a ocuparla, sin embargo si la posición está ocupada solo se asientan si su salud es mayor que la del coral que ocupa la posición; y depredación, donde algunos corales mueren dejando espacio en el arrecife. (Ver Figura 2.8)

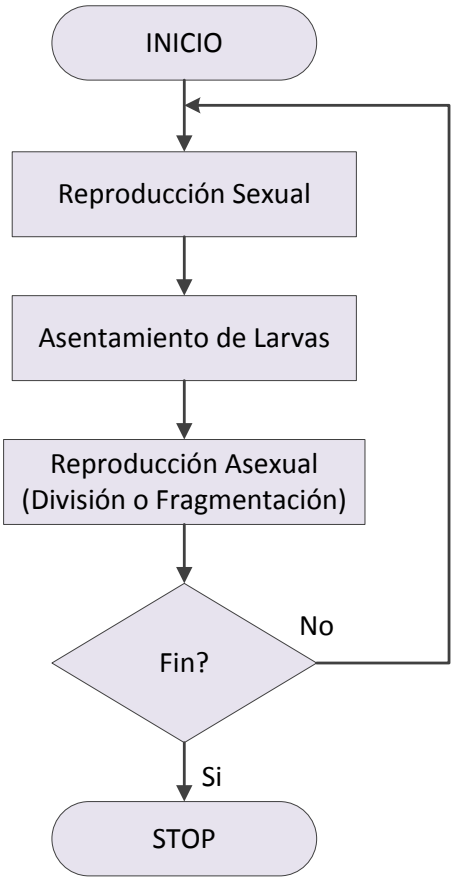

Figura 2.17: Diagrama de flujo del algoritmo basado en los Arrecifes de Coral

3.2.3. Los algoritmos genéticos en la sintonización de TMDs

Como se ha apuntado anteriormente, el enfoque clásico de sintonización de TMDs no tiene en cuenta ciertas variables que son de interés en la actualidad, como la localización optima de los TMDs, la carrera que pueda sufrir el TMD o incluso su coste.

Es por esto que, en los últimos años, múltiples autores han abordado el problema con un enfoque diferente al tradicional utilizando algoritmos de optimización metaheurísticos. Dentro de estos, los algoritmos genéticos se han usado ampliamente para la sintonización de TMDs. El uso de estos algoritmos aporta gran versatilidad y puede ser usado para optimizar diferentes criterios de diseño.

Dentro de los algoritmos genéticos, existen múltiples enfoques que han sido usados para la sintonización de TMDs. Los primeros algoritmos utilizados, fueron algoritmos genéticos simples con codificación binaria de las soluciones que buscaban el óptimo global del problema (Hadi y Arfiadi, 1998) (Singh et al., 2002). Debido a que la mayoría de las propiedades de los TMDs son números reales, el uso de algoritmos genéticos con codificación real es más apropiado y es la tendencia dominante en la actualidad (Arfiadi y Hadi, 2011). Recientemente, se están aplicando algoritmos genéticos multi-objetivo con el propósito de optimizar varias variables como el coste o la carrera del TMD (Pourzeynali et al., 2013).

En este trabajo, se analizará el uso de los algoritmos genéticos multimodales para encontrar varias soluciones a un problema de sintonización de TMDs.

4. BASES DE LOS ALGORITMOS GENÉTICOS

Como se apuntó en el capítulo anterior, los Algoritmos Genéticos o AG son una clase de algoritmos evolutivos usados para resolver problemas de búsqueda y optimización de forma secuencial o iterativa.

Los AG están basados en la teoría de la evolución por selección natural enunciada por (Darwin, 1959). Según esta teoría, solamente los individuos más aptos prosperan y tienen la oportunidad de transmitir algunas de sus características a sus descendientes. Además, existen modificaciones espontaneas en los individuos que pueden hacer que se adapten mejor a su entorno o no. Después de varias generaciones, la aptitud de los individuos que forman la población es mayor que la de las generaciones predecesoras.

Los primeros AG surgieron a principios de los años 60, inspirados en modelos simples de evolución biológica. John Henry Holland trabajó en estas ideas que recopiló en su libro "Adaptation in Natural and Artificial Systems" (Holland, 1975).

En las décadas de los 60 y 70, las ideas de Holland fueron implementadas en modelos en ordenador, pero muchas de las propiedades observadas no pudieron ser comprobadas experimentalmente debido a la falta de recursos computacionales.

Más adelante, en 1989, David Goldberg publicó el primer libro de texto sobre AG titulado "Genetic algorithms in search, optimization and machine learning" (Goldberg, 1989) que sirvió de base para los futuros desarrollos en el campo de los AG producidos en la década de los 90.

A partir de la década de los 90, los AG han sufrido un gran crecimiento y diversificación reflejado en la proliferación de numerosos congresos, libros y publicaciones periódicas. Actualmente, los AG son usados en un gran número de campos como la ingeniería, la economía o la simulación de ecosistemas biológicos.

Este capítulo tiene como objetivo introducir la teoría sobre la que se asientan los AG. En primer lugar, se definirá la terminología utilizada al trabajar con AG para más adelante desarrollar el funcionamiento de un algoritmo genético básico. A continuación, se tratará la representación del problema y la codificación de las soluciones. Más adelante, se profundizará en cada uno de los procesos que forman un AG para terminar el capítulo con algunas ventajas e inconvenientes de estos algoritmos.

4.1. Terminología

A continuación, se definirán una serie de términos utilizados comúnmente cuando se trabaja con AG. Estos términos tienen su origen en la biología y en nuestro caso representan una parte del problema a resolver.

Individuo: una de las soluciones candidatas al problema.

Genes: elementos que forman una solución del problema.

Población: conjunto de potenciales soluciones sobre las que trabaja el algoritmo genético.

Generación: conjunto de individuos presentes en cada iteración del algoritmo.

Padres: individuos de los que desciende la nueva generación.

Hijos: individuos obtenidos por mecanismos evolutivos a partir de los padres.

Aptitud (fitness): calidad de respuesta al problema que pretende se resolver.

4.2. Funcionamiento de un algoritmo genético básico

Un AG parte de una población inicial perteneciente al espacio de soluciones que será transformada paso a paso por medio de una serie de operadores genéticos. Cualquier AG cuenta con cinco fases: generación de la población inicial, evaluación, selección, cruce y mutación y sustitución. Todas excepto la generación de la población inicial forman el bucle del algoritmo como se muestra en la Figura 4.1.

De esta manera, la primera tarea es generar la población inicial, puede generarse de manera aleatoria o por medio de otros métodos.

Una vez generada la población inicial, se evaluarán todos los individuos presentes en la población para obtener su aptitud frente al problema. Cuando todos los individuos presentes en la población han sido evaluados, se seleccionan una serie de individuos que constituirán el conjunto de individuos Padres. Esta selección puede hacerse en función de la aptitud de los individuos o en función de otros criterios.

A partir del conjunto de individuos Padres, por medio de la reproducción y la mutación se obtiene el conjunto de individuos Hijos. En este punto, se cuenta con la población de la generación actual, un conjunto de individuos Padres y un conjunto de individuos Hijos.

El último paso consiste en obtener la nueva generación mediante la sustitución de los individuos de la generación actual por algunos de los individuos presentes en el conjunto Hijos. Existen distintos criterios de sustitución, uno de los más comunes es el criterio elitista, que conserva los individuos con mejor adaptación al problema.

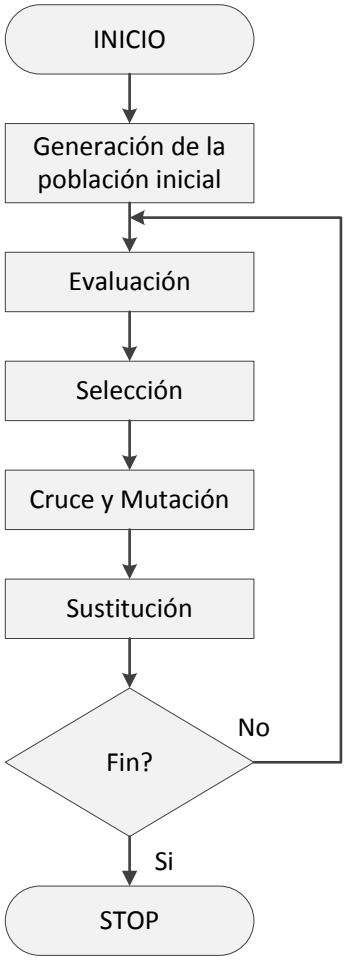

Figura 4.1: Diagrama de flujo de un algoritmo genético básico

4.3. Representación del problema

El primer proceso que se debe llevar a cabo a la hora de desarrollar un algoritmo genético es representar el problema a resolver, esto implica definir la función objetivo, las restricciones de las variables y las zonas del espacio de búsqueda que serán exploradas.

Generalmente, en los problemas de optimización existe una función de que determina el valor o calidad de la solución. Esta función se emplea como función objetivo del algoritmo (fitness function) siempre que los valores de calidad que asigne a las soluciones sean positivos.

Las funciones de calidad pueden ser definidas para problemas de maximización o minimización, existiendo métodos para transformar una función de maximización en otra de minimización y viceversa

4.4. Codificación de las soluciones

A la hora de trabajar con AG es necesario codificar de alguna forma los puntos del espacio de búsqueda donde se esté trabajando. De esta manera, cada posible solución del problema tendrá una cadena codificada. Existen diferentes tipos de codificaciones: binaria, real y permutacional. Elegir un tipo de codificación u otra depende del problema a resolver.

4.4.1. Codificación binaria

La codificación binaria es una de las más extendidas dado que gran parte de la teoría desarrollada sobre AG emplea esta codificación. Además, permite la aplicación de operadores genéticos de manera sencilla y es fácilmente implementable. En este caso la solución codificada está formada por una cadena de unos y ceros.

Sin embargo, la codificación binaria no se adapta adecuadamente a todos los problemas ya que la complejidad derivada de utilizar esta codificación disminuye el rendimiento del algoritmo. Este hecho motiva la aparición de otras codificaciones que se adaptan mejor al problema y facilitan el trabajo del desarrollador.

4.4.2. Codificación real

La codificación real emplea cadenas de números reales para representar las soluciones al problema. Generalmente, se emplea la codificación directa donde el valor de la cadena coincide con el valor de la solución. Esta codificación facilita la implementación en muchos casos, por ejemplo, cuando las soluciones están formadas por números reales, cuya codificación con números binarios sería muy compleja.

4.4.3. Codificación permutacional

La codificación permutacional se emplea en problemas donde el objetivo es ordenar algo. Así, una posible solución al problema se obtiene a través de la permutación de los n números presentes en la cadena, teniendo el espacio de búsqueda una dimensión de $n!$.

Existe una variante denominada *codificación permutacional con repetición* donde la cadena que representa la solución se repite tantas veces como sea necesario.

Estas codificaciones son empleadas, por ejemplo, para resolver problemas de programación de operaciones o *scheduling* donde cada número representa los trabajos u operaciones que serán llevados a cabo sobre los centros de trabajo.

4.5. Generación de la población inicial

La primera tarea del algoritmo es generar la población de individuos con los que trabajará en la primera iteración. Una de las maneras más eficaces es la generación aleatoria, que obtiene una población uniformemente distribuida por todo el espacio de búsqueda, aumentando la capacidad de exploración. Como contrapartida, la calidad media de las soluciones será baja, necesitando un mayor número de iteraciones para converger hacia una solución aceptable.

En algunos casos, es conveniente emplear otras formas de inicialización que utilizan heurísticas más sencillas para obtener soluciones de una región concreta del espacio de búsqueda facilitando la convergencia del algoritmo.

La diversidad de la población inicial afecta de manera significativa al rendimiento del algoritmo. La capacidad de exploración depende del tamaño de la población, cuanto mayor sea el tamaño mayor será la capacidad de exploración. Sin embargo, a mayor tamaño de la población menor velocidad de búsqueda, necesitando un mayor número de iteraciones para converger.

4.6. Selección

La selección es el proceso mediante el que se determina que individuos sobreviven y se reproducen y cuales son eliminados debido a una baja adaptación al entorno. Existen numerosos métodos de selección y es un proceso ampliamente estudiado.

Los métodos basados en la calidad o aptitud de las soluciones seleccionan los mejores individuos para formar el conjunto de Padres que más adelante se reproducirán. Del mismo modo, los peores individuos no serán seleccionados y desaparecerán.

El grado con el que las mejores soluciones son favorecidas se denomina presión selectiva. La rapidez de convergencia de un algoritmo genético está determinada, entre otros factores, por la presión selectiva. Cuando la presión selectiva es alta, el algoritmo converge rápidamente. Sin embargo, la prematura convergencia puede llevar al algoritmo hacia un óptimo local. Si la presión selectiva es baja, la rapidez de convergencia del algoritmo disminuirá y se empleará más tiempo en encontrar el óptimo.

Existen tres tipos de métodos de selección: basados en la calidad, basados en el orden y basados en competiciones o torneos. A continuación se expondrán los más relevantes.

4.6.1. Métodos proporcionales a la calidad

En estos métodos, la selección de los padres se lleva a cabo a partir de la calidad o aptitud de los individuos frente a la del resto de la población. De esta manera, los individuos mejor adaptados, que tienen una calidad superior al resto, pasarán a la población de padres llenándola con muchas copias. Como consecuencia, la diversidad se ve sacrificada haciendo que el algoritmo converja de forma prematura, posiblemente hacia un óptimo local. Dentro de esta categoría encontramos la selección proporcional o de ruleta y la selección estocástica por restos.

Selección proporcional: también conocida como selección por rueda de ruleta. Es un método desarrollado por Holland en 1975 y ampliamente estudiado por otros autores. La probabilidad de selección de un individuo depende la calidad del individuo a través de la Ecuación 4.1:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{4.1}$$

Dónde f_i representa la calidad o aptitud del individuo y n el tamaño de la población.

A partir de las probabilidades de selección de todos los individuos se confecciona una ruleta dividida en *n* regiones proporcionales a la probabilidad de selección. La selección se realiza rotando la ruleta un número de veces y la zona donde apunte el marcador determinará el individuo seleccionado.

Selección estocástica por restos: Es un método que asigna a cada individuo un número de copias en la población de padres dependiendo de su probabilidad según la Ecuación 4.2 redondeando al menor entero:

$$Nº\ Copias = p_i \cdot n \tag{4.2}$$

Con los restos, se utiliza la selección proporcional para seleccionar los individuos que ocuparán las posiciones vacantes.

4.6.2. Métodos basados en el orden

Los métodos basados en el orden seleccionan los individuos a partir de una lista ordenada en función de la calidad de los individuos. La ventaja de estos métodos es que elimina la supremacía de los individuos más aptos evitando la convergencia prematura hacia estos. Entre estos métodos se encuentran la selección por ordenación lineal y la no lineal.

Selección por ordenación lineal: Fue propuesta por (Baker, 1985) y selecciona los individuos a partir de una lista ordenada según el orden decreciente de calidades. Este método cuenta con un parámetro *r* que permite controlar la presión selectiva, variando desde presión selectiva nula hasta presión selectiva máxima. Cuando la presión selectiva es máxima el mejor individuo cuenta con dos copias en la población mientras que el peor no cuenta con ninguna.

Selección por ordenación no lineal: Fue desarrollado por (Michalewicz, 1995) y la diferencia entre la probabilidad de selección entre individuos no es lineal. Al igual que en el caso anterior, cuenta con un parámetro capaz de controlar la presión selectiva. Si aumenta este parámetro, la presión selectiva aumentará dando mayor importancia a los individuos mejor dotados.

4.6.3. Métodos basados en competiciones o torneos

Estos métodos fueron desarrollados por (Brindle, 1981) y se selecciona cada padre a través de la competición entre un número determinado de individuos *z* seleccionados de forma aleatoria. Este proceso se repetirá tantas veces como individuos se quieran obtener en la población padre. Para aumentar la presión selectiva debemos aumentar el valor de *z*, aumentando los individuos que compiten por formar parte de la población padre.

Los métodos basados en torneos son habitualmente utilizados en algoritmos genéticos multimodales o multiobjetivo donde se quiere controlar la cantidad de individuos que pasarán a la siguiente población dependiendo de en qué región del espacio de búsqueda se encuentren.

Frente a problemas multimodales se utilizan la selección por torneo de Boltzmann y la selección por torneo con actualización continua. Frente a problemas multiobjetivo se utilizan torneos que usan la superficie de Pareto.

4.6.4. Tipo de selección a utilizar

A la hora de elegir el tipo de selección a utilizar en el algoritmo, debe examinarse el problema a resolver y las características de los métodos de selección candidatos, como su comportamiento frente a una modificación de la función objetivo o la escala. La escala de calidad de la población se reduce a medida que converge el algoritmo, modificándose la presión selectiva.

Los métodos proporcionales no se ven influidos por la escala, pero si son sensibles a modificaciones en la función objetivo. Los métodos basados en el orden y en competición no son sensibles ni a la escala ni a un cambio en la función objetivo.

En cuanto a la presión selectiva, (Goldberg y Deb, 1991) recomiendan usar en las primeras generaciones una presión selectiva baja para evitar la prematura convergencia hacia un óptimo local. Sin embargo, a medida que avanza el algoritmo se deberá usar una presión selectiva cada vez mayor.

Además de los métodos comentados anteriormente, existen otras estrategias para mejorar el rendimiento del algoritmo. La más utilizada es el *método elitista* que consiste en pasar al mejor individuo directamente a la siguiente población.

4.7. Reproducción

La reproducción en los algoritmos genéticos consta de dos fases, en primer lugar, se realiza la operación de cruce donde se intercambia material genético entre los padres y en segundo lugar se realiza la mutación sobre los hijos obtenidos. De esta manera los operadores de reproducción son dos: el operador cruce y el operador mutación.

4.7.1. Operadores cruce

Los operadores cruce permiten el intercambio de material genético entre una generación y su sucesora. El cruce se lleva a cabo entre dos individuos en función de una probabilidad, parámetro del algoritmo, que llamaremos probabilidad de cruce, p_c. El valor asignado a la probabilidad de cruce depende del problema y del propio algoritmo genético, aunque un valor común es 60%, propuesto por (Goldberg, 1989).

El objetivo del cruce es combinar el material genético de los padres de la generación actual, que debido al proceso de selección tienen la capacidad de transmitir mejores genes que otros individuos de su generación. Es decir, parte de los mejores individuos para crear individuos aún mejores.

Si los padres seleccionados no fuesen los mejores, el resultado final sería una mutación debida al cambio de la mayor parte del material genético entre cada uno de los padres, por otro que no transmite características beneficiosas.

Existen gran cantidad de operadores cruce y dependiendo del tipo de codificación de las soluciones y el tipo de algoritmo deberá elegirse uno u otro. A continuación, se presentan algunos de los operadores cruce existentes.

Operador con un punto de cruce: es el primer operador cruce empleado a partir del que se han desarrollado los demás. Se genera un punto de corte de manera aleatoria a partir del cual se realiza el intercambio de material genético entre los padres. Este operador es comúnmente usado para codificación binaria.

Padre 1	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**
Padre 2	1	1	0	1	0	0	1	1	0	0	1	1	1	0	0	1	0
Hijo 1	1	1	0	1	0	0	1	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**
Hijo 2	**0**	**1**	**1**	**0**	**1**	**0**	**0**	1	0	0	1	1	1	0	0	1	0

Figura 4.2: Operador de cruce por un punto

Operador con varios puntos de cruce: es similar al anterior. Se generan varios puntos de cruce de manera aleatoria a partir de los cuales se realiza el intercambio de material.

Operador de cruce uniforme: cada gen tiene una probabilidad del 50% de ser intercambiado por su correspondiente gen del otro individuo. Para llevarse a cabo se genera una máscara de ceros y unos que indicará que puntos se realiza el cruce. En la Figura 4.3 se muestra este tipo de cruce con su máscara.

Padre 1	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**
Padre 2	1	1	0	1	0	0	1	1	0	0	1	1	1	0	0	1	0
Máscara	0	1	1	0	1	1	0	0	0	1	1	1	0	1	1	0	1
Hijo 1	1	**1**	**1**	1	**1**	**0**	1	1	0	**0**	**1**	**0**	1	**0**	**1**	1	**0**
Hijo 2	**0**	1	0	**0**	0	0	**0**	**1**	**1**	0	1	1	**0**	0	0	**1**	0

Figura 4.3: Operador de cruce uniforme

Operador de cruce ordenado (Order Crossover): es uno de los operadores más estudiados, desarrollado inicialmente para la codificación permutacional, se determina un segmento que no será alterado, siendo las posiciones fuera de ese segmento las que serán cruzadas entre los dos padres. Para el Hijo 1 se tomará una cadena auxiliar correspondiente al Padre 2, comenzando a rellenar las posiciones con los genes que están a partir del segundo corte.

Padre 1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Padre 2	9	3	7	8	2	6	5	1	4
Paso 1	x	x	x	**4**	**5**	**6**	**7**	x	x
Paso 2	3	8	2	**4**	**5**	**6**	**7**	1	9

Figura 4.4: Cruce ordenado (OX)

Operadores de cruce basados en entornos: estos operadores, utilizados con codificación real, generan los genes de los hijos a partir de valores tomados del entorno asociado a los genes de los padres, normalmente a partir de un intervalo. Dentro de estos operadores encontramos el operador BLX (Blend Crossover), el operador SBX (Simulated Binary Crossover) y el operador PNX (Normal Parents-centric Crossover).

El *operador BLX* genera dos hijos eligiendo los nuevos genes dentro del intervalo marcado por la Ecuación 4.3:

$$[C_{min} - I\alpha\ , C_{max} + I\alpha] \tag{4.3}$$

Donde C_{max} y C_{min} son los valores máximo y mínimo de los genes de los padres 1 y 2 e I es la diferencia entre estos valores, como muestra las Ecuaciones 4.4 y 4.5:

$$\begin{aligned} C_{max} &= \max\{c_i^1, c_i^2\} \\ C_{min} &= \min\{c_i^1, c_i^2\} \end{aligned} \tag{4.4}$$

$$I = C_{max} - C_{min} \tag{4.5}$$

De forma gráfica:

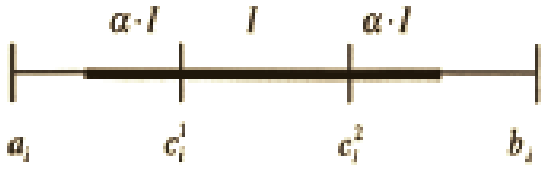

Figura 4.5: Operador BLX

El *operador SBX* genera dos hijos eligiendo los nuevos genes a partir de la Ecuación 4.6:

$$h_i^1 = \frac{1}{2}[(1 - \beta_k) \cdot c_i^1 + (1 + \beta_k) \cdot c_i^2] \tag{4.6}$$

Donde β_k es un número generado a partir de la distribución dada en la Ecuación 4.7:

$$\begin{aligned} \beta(u) &= (2u)^{\frac{1}{\eta+1}} && si\ u(0,1) \leq 0.5 \\ \beta(u) &= (2(1-u))^{\frac{1}{\eta+1}} && si\ u(0,1) > 0.5 \end{aligned} \tag{4.7}$$

De forma gráfica el efecto de este operador se muestra en la Figura 4.6:

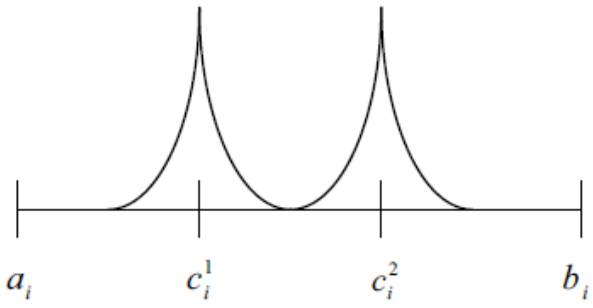

Figura 4.6: Operador SBX

El *operador PNX* genera los genes de los hijos a partir de una distribución normal que tiene como parámetros:

$$h_i^1 = N[c_i^1, \frac{|c_i^1 - c_i^2|}{\eta}] \tag{4.8}$$

De forma gráfica el efecto de este operador se muestra en la Figura 4.7:

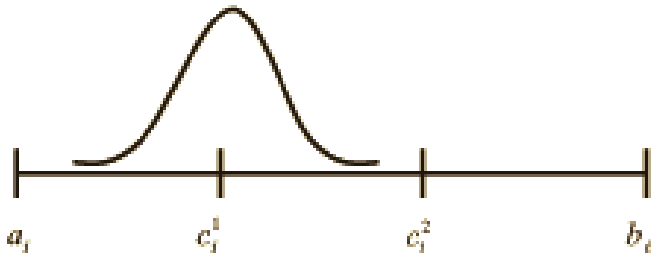

Figura 4.7: Operador PNX

4.7.2. Operadores mutación

El objetivo del operador mutación es explorar áreas del espacio de búsqueda no visitadas incluyendo diversidad en la nueva generación. De manera formal, el operador mutación permite que todos los individuos del espacio de búsqueda tengan una probabilidad de ser explorados mayor que cero. También permite recuperar características beneficiosas perdidas mediante la selección y el cruce que de otra manera no podrían ser recuperadas.

Existen multitud de operadores mutación. A continuación, se expondrán algunos de los operadores mutación más comunes dependiendo de la codificación utilizada.

Mutación estándar: este operador es utilizado con codificación binaria y consiste en cambiar los ceros por unos y viceversa según la probabilidad de mutación p_m. Esta probabilidad es muy baja para no desviar la búsqueda incluyendo demasiada diversidad. Un valor correcto, según Holland, estaría comprendido entre el 0.1% y el 1%.

Mutación no uniforme: este operador es utilizado con codificación real y consiste en generar el nuevo gen v_ka partir de la Ecuación 4.9, dependiendo si el número aleatorio generado es 0 o 1:

$$v_k = \begin{cases} v_k + \Delta(t, UB - v_k) & si\ num.aleatorio = 0 \\ v_k + \Delta(t, v_k - LB) & si\ num.aleatorio = 1 \end{cases} \tag{4.9}$$

Donde UB y LB son los límites superior e inferior de la variable y el valor de $\Delta(t, y)$ viene dado por la Ecuación 4.10:

$$\Delta(t, y) = y \cdot (1 - r^{(1-\frac{t}{T})^b} \begin{cases} r \equiv num.aleatorio\ entre\ 0\ y\ 1 \\ T \equiv num.total\ de\ generaciones \\ b \equiv grado\ de\ dependencia\ (b = 5) \end{cases} \tag{4.10}$$

Mutación basada en el orden: este operador es utilizado con codificación permutacional y consiste en el intercambio de dos posiciones de la cadena que forma el individuo seleccionadas de manera aleatoria.

4.8. Sustitución

Una vez obtenidos los individuos descendientes de una generación anterior se debe construir la siguiente generación incluyendo los nuevos individuos en la nueva generación. A grandes rasgos, existen dos tipos de técnicas de sustitución: la sustitución para algoritmos genéticos generacionales y la sustitución para algoritmos genéticos de creación continua.

En los *algoritmos genéticos generacionales* se parte de una población inicial de N individuos a partir de los cuales se obtienen N padres que formaran N hijos mediante los operadores cruce y mutación. Estos N hijos sustituirán a la generación anterior cerrando el bucle.

El elitismo es utilizado comúnmente en los algoritmos genéticos generacionales, de esta manera, de la población de hijos que formará la siguiente generación se elimina el peor individuo y se incluye el mejor de la generación anterior. Así siempre se conserva una copia del mejor individuo.

En los *algoritmos genéticos de creación continua* se seleccionan individuos dos a dos para obtener dos hijos que serán incorporados en la población mediante el proceso de sustitución. Este proceso puede ser totalmente aleatorio, a partir de la calidad del individuo o mediante otras técnicas de clasificación.

La principal diferencia entre algoritmos genéticos generacionales y de creación continua es que cuando aparece un individuo con buenas cualidades, este está disponible para ser procesado inmediatamente.

4.9. Ventajas y desventajas de los algoritmos genéticos

Una de las principales ventajas de los Algoritmos Genéticos es que son algoritmos paralelos, es decir, operan con varias soluciones simultáneamente en cada iteración. Mientras las técnicas basadas en trayectorias solo exploran el espacio de soluciones en una dirección, los Algoritmos Genéticos exploran el espacio de soluciones en varias direcciones simultáneamente. De esta manera donde una técnica basada en trayectorias se quedaría atascada y debería comenzar de nuevo, el algoritmo genético desecha la solución y continúa con otros individuos de la población.

Debido a la diversidad de la población, es decir, la diversidad de soluciones con las que trabaja el algoritmo, los algoritmos genéticos resultan menos vulnerables a los óptimos locales que las técnicas basadas en trayectorias. Muchos algoritmos quedan atrapados en óptimos locales dado que en las cercanías de estos no existe ninguna solución mejor y concluyen con que han alcanzado la mejor de las soluciones.

Por último, a la hora de implementar un algoritmo genético sobre algún problema en concreto, el programador no necesita conocimiento específico del problema dado que los cambios que se realizan en las soluciones son aleatorios para luego utilizar las funciones de aptitud sobre ellas.

Como desventajas, los diseñadores deben codificar correctamente el problema. Esta codificación debe permitir la aplicación de los operadores genéticos de manera que no se produzcan continuamente resultados carentes de sentido.

Además, si los mecanismos evolutivos que rigen el proceso no son los adecuados el algoritmo puede converger prematuramente debido a que uno de los individuos es más apto que todos sus competidores.

Este hecho, ha motivado la aparición de algoritmos genéticos multimodales que conserva la diversidad en todo momento explorando todo el espacio de búsqueda. Estos algoritmos, son capaces de encontrar tanto el óptimo global del problema como otros óptimos locales que en muchos casos pueden ser interesantes para los diseñadores.

5. ALGORITMOS GENÉTICOS MULTIMODALES

Numerosos problemas existentes en la actualidad tienen varias soluciones interesantes, presentando múltiples óptimos locales o globales. Cuando nos enfrentamos a este tipo de problemas, se desean obtener varias de estas soluciones. En un algoritmo genético simple, el proceso evolutivo suele provocar la convergencia hacia una zona concreta del espacio de búsqueda, abandonando otras zonas prometedoras donde podrían encontrarse soluciones igualmente válidas. Debido a esto, han surgido los algoritmos genéticos multimodales, los cuales tratan de conservar la diversidad para permitir la búsqueda simultánea en diferentes áreas del espacio de búsqueda.

En este capítulo, se estudiarán tres algoritmos genéticos multimodales que son el objeto de análisis de este TFM. En primer lugar, se definirá formalmente que es un problema multimodal y porqué la sintonización de TMDs a escala real es uno de estos problemas. Más adelante se hará un breve repaso por varios de los algoritmos genéticos multimodales existentes en la actualidad y, por último, se detallarán los tres algoritmos que forman parte del estudio: Clearing, RCS y SCGA.

5.1. Problemas de optimización multimodales

Estrictamente, un problema de optimización multimodal es aquel que tiene varios óptimos, todos ellos de la misma calidad. Sin embargo, el termino multimodal se aplica a problemas que tienen varias soluciones posibles, aunque tengan diferente calidad. La mayoría de los problemas de optimización suelen tener varios óptimos, uno de ellos mejor que el resto, denominado optimo global, siendo el resto óptimos locales. En muchos casos, varios de los óptimos son igual de válidos, bien por tener la misma calidad, o bien porque se establezca otro criterio cualitativo que haga que no se pueda decidir entre uno u otro.

De manera general, un problema multimodal es aquel en cuyo espacio de búsqueda pueden encontrarse múltiples óptimos (locales o globales), como puede verse en la Figura 5.1. Esta figura representa la función de Rastrigin, un ejemplo típico de función multimodal no lineal para espacios de búsqueda bidimensionales. En situaciones de este tipo interesa que los algoritmos de optimización encuentren el mayor número posible de dichos óptimos. Esto es especialmente importante cuando los óptimos locales son más amplios que el óptimo global, con lo que se dificulta la localización de este último y la prematura convergencia hacia el óptimo local.

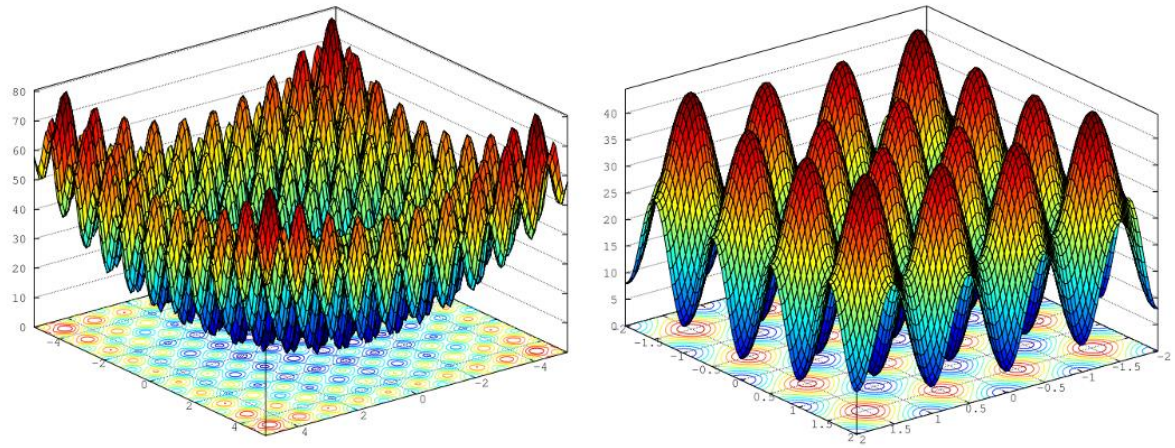

Figura 5.1: Función de Rastrigin

La optimización de TMDs es debido a varios motivos un problema de optimización multimodal. Cuando se quieren instalar varios TMDs en un edificio de varias plantas, existen múltiples configuraciones posibles que arrojan resultados similares, es decir existen varios óptimos globales. Además, en un gran número de casos, por cuestiones prácticas, es más conveniente implementar una configuración correspondiente a algún optimo local que implementar la configuración correspondiente al óptimo global.

5.2. Los algoritmos genéticos frente a problemas multimodales

Tradicionalmente, los algoritmos genéticos se han considerado una herramienta válida para encontrar el óptimo de funciones unimodales, ya que debido al propio proceso evolutivo llevado a cabo por el algoritmo, este converge hacia una única solución dentro del espacio de búsqueda. Sin embargo, cuando los algoritmos genéticos se enfrentan a problemas multimodales, el proceso evolutivo suele provocar la convergencia hacia una zona concreta del espacio de búsqueda abandonando la exploración del resto del espacio. Este proceso puede verse gráficamente en la serie de Figuras 5.2, 5.3 y 5.4.

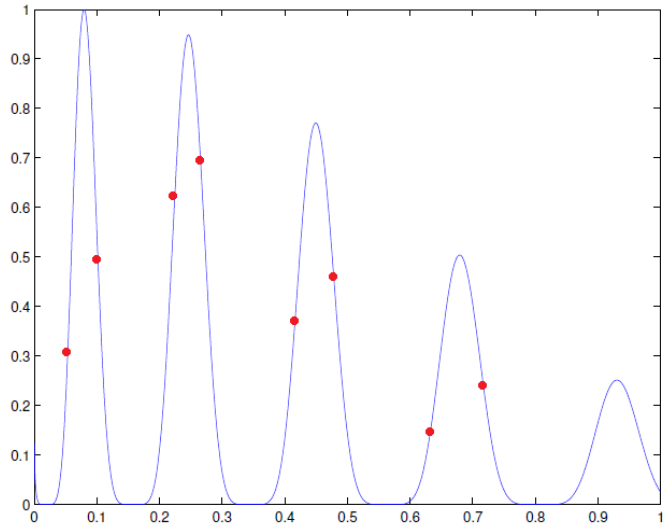

Figura 5.2: Población inicial uniformemente distribuida

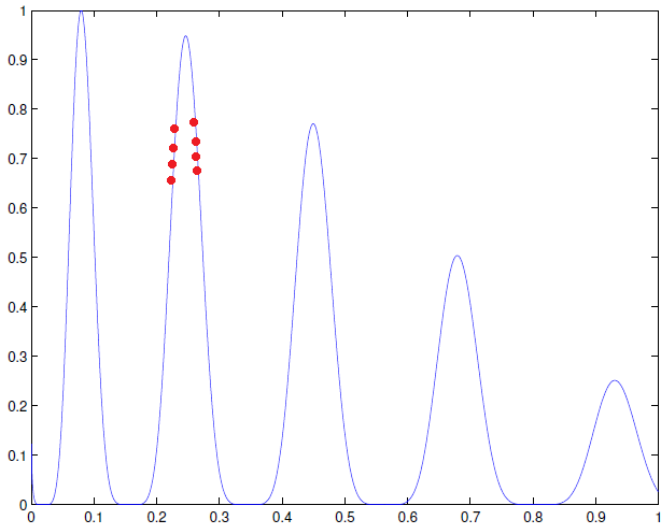

Figura 5.3: Pérdida de la diversidad

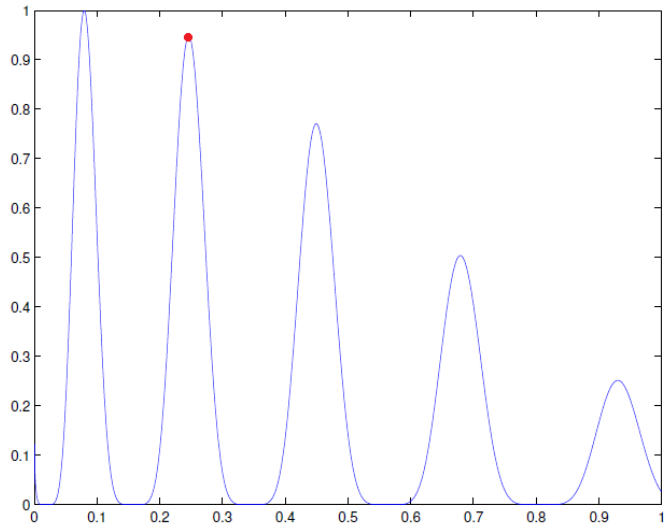

Figura 5.4: Convergencia prematura hacia un óptimo local

Cuando se trabaja con algoritmos de optimización frente a problemas multimodales es habitual hablar de exploración y explotación. La *exploración* se refiere a la tarea de examinar eficientemente todo el espacio de búsqueda y encontrar nuevas soluciones potencialmente mejores. La *explotación*, es el proceso de mejorar los rasgos de las mejores soluciones encontradas durante el proceso de exploración, con el objetivo de encontrar soluciones de mayor calidad.

Los algoritmos que favorecen la explotación poseen una mayor velocidad de convergencia, pero suelen quedarse atrapados en un óptimo local. Los que fomentan en exceso la exploración pueden no llegar a converger hacia la solución óptima.

Para evitar que un algoritmo genético se quede atrapado en un óptimo local debe mantenerse la diversidad de los individuos en cada generación, favoreciendo así la exploración de todo el espacio de búsqueda.

Existen distintas estrategias para mantener la diversidad. Algunos algoritmos utilizan una separación geográfica de los individuos a lo largo del espacio de búsqueda trabajando en paralelo con poblaciones aisladas, que cada cierto número de generaciones intercambian individuos entre ellas. Sin embargo, las estrategias más comunes, estudiadas en este TFM, utilizan el concepto de nicho o especie dentro de una población. Estas especies o nichos están formadas por individuos muy similares entre sí.

5.3. Estrategias basadas en nichos

La idea bajo este tipo de técnicas se basa en la modelización de los mecanismos que tienen lugar en los ecosistemas naturales. En ellos, los animales compiten para sobrevivir especializándose cada especie en diferentes aspectos. Por lo tanto, un nicho se puede ver como un subespacio dentro del ecosistema que puede soportar diferentes tipos de vida, ya que es posible la coexistencia de especies muy diferentes dentro del mismo nicho.

Una especie podría ser definida como un conjunto de individuos de similares características capaz de relacionarse con individuos pertenecientes a la misma especie. Evidentemente, para cada nicho, los recursos físicos son limitados y deben ser compartidos entre toda la población de ese nicho.

Por lo tanto, los algoritmos basados en nichos tratan de conseguir que dentro del entorno, es decir, en el espacio de búsqueda, existan diferentes nichos y especies que compitan por los diferentes recursos.

A continuación, se expondrán algunas de las estrategias basadas en nichos existentes en la actualidad.

5.3.1. Penalización de la calidad (*Fitness sharing*)

El método de penalización de la calidad o f*itness sharing* es probablemente el método más conocido de las estrategias basadas en nichos. Fue propuesto por (Goldberg y Richardson, 1987) y está basado en la penalización de las áreas del espacio de búsqueda con más soluciones en la población.

De esta manera, la técnica de Fitness Sharing, modifica la estructura del espacio de búsqueda, reduciendo la calidad de aquellos individuos pertenecientes a áreas con una alta densidad de individuos.

Para conseguir esto, se define una nueva calidad que será utilizada en el proceso de selección, a partir de la Ecuación 5.1:

$$f_i^* = \frac{f_i}{Sh(i)}$$
$$Sh(i) = \sum_{j=1}^{N} Sh(d(i,j)) \qquad (5.1)$$
$$con\ i \in \{1, N\}\ y\ N = tamaño\ de\ la\ población$$

Donde f_i^* es la calidad modificada o *shared fitness* y $Sh(d(i,j))$ es la función de modificación o *sharing function,* definida según la Ecuación 5.2:

$$Sh\big(d(i,j)\big) = \begin{cases} 1 - \left(\frac{d(i,j)}{\sigma_{share}}\right)^{\alpha} & Si\ d(i,j) < \sigma_{share} \\ 0 & En\ otro\ caso \end{cases} \qquad (5.2)$$

A partir de estas funciones, el objetivo de esta estrategia es dividir el espacio de búsqueda en distintas regiones o nichos. Para lograrlo, el algoritmo, para cada solución busca en la población cuáles son sus soluciones vecinas. Una solución *i* se considera vecina de otra solución *j* si su distancia $d(i,j)$ es menor que un radio parámetro del algoritmo σ_{share}.

Si existen muchas soluciones en su nicho el valor de la función de modificación será elevado penalizando la calidad. De otra forma, si existen pocas soluciones en su nicho, la función de modificación tomará un valor reducido y su calidad apenas se verá modificada.

A partir de este método, otros autores han modificado algunas partes del algoritmo para mejorarlo. Por ejemplo, (Oei et al., 1991) modificó el proceso para crear el método denominado *Continuously Updated Sharing (CUS).* La diferencia fundamental con el método original es que en lugar de penalizar las soluciones en la población de partida, se penalizan aquellas soluciones elegidas por el proceso de selección. De esta manera, la calidad de una zona donde existan varios padres, elegidos por el proceso de selección, será penalizada limitando la posibilidad de que nuevos padres similares sean elegidos.

Otros métodos basados en el concepto de penalización de los individuos son el *Adaptive Niche Hierarchical Genetic Algorithm (ANHGA)* y el *Niche Identification Techniques (NIT).*

5.3.2. Aclarado de la población (*Clearing*)

Los métodos de aclarado de la población o *clearing*, introducidos por (Pétrowski, 1996), utilizan el concepto de recursos limitados para evitar que dos soluciones muy diferentes compitan entre si. En los ecosistemas naturales, los individuos de una misma especie compiten por los recursos existentes. Sin embargo, dos especies con diferentes características pueden convivir en el mismo entorno, ya que no tienen que competir por los mismos recursos.

En primer lugar, se clasifican los individuos presentes en la población según su calidad de forma decreciente. De esta manera, el primer individuo será el mejor y el último el peor. Se compara al primer individuo (individuo dominante) con el resto de la población de manera descendente. Aquellos individuos que están dentro de su radio de nicho (individuos dominados) son eliminados. Este proceso continua con el segundo individuo que aún no haya sido eliminado, y eliminará los individuos dominados por él. El proceso termina cuando todos los individuos han sido eliminados.

Como resultado del proceso anteriormente expuesto se obtiene una subpoblación de individuos dominantes sobre la que se llevará a cabo la selección. Así se conserva la diversidad, ya que la selección se realiza sobre individuos que pertenecen a distintos nichos, es decir que son muy diferentes.

Existen algoritmos derivados de este concepto como *Restricted competition selection (RCS)* y *Restricted competition selection with pattern search (RCS-PSM).*

Dos de los algoritmos analizados en este TFM pertenecen a esta categoría y serán explicados con mayor detalle posteriormente.

5.3.3. Agrupamiento (*Crowding*)

Los métodos de agrupamiento o *crowding*, originalmente desarrollados por (De Jong, 1975), se basan en la utilización de un torneo entre padres e hijos para formar la siguiente generación.

En este algoritmo no existe proceso de selección, dividiéndose la población inicial en pares para reproducirse. A continuación, los hijos generados compiten contra los padres en un torneo obteniéndose el individuo que formará parte de la siguiente generación.

Antes de realizarse el torneo, se determina que combinación padre (P) hijo (H) es la más similar entre las dos posibilidades: (P1-C1 y P2-C2) o (P1-C2 y P2-C1). Una vez conocido esto, el padre compite contra su hijo para determinar que individuo formará parte de la siguiente generación. El criterio de la competición es exclusivamente la calidad de los individuos.

En función de la estrategia de remplazo, pueden establecerse dos tipos de estrategias: *crowding determinístico* y *crowding no determinístico.* En el crowding determinístico siempre se escoge como ganador al individuo con mayor calidad. Sin embargo, en el crowding no determinístico, la elección se realiza a partir de una probabilidad. Ajustando el valor de esta probabilidad, se puede actuar sobre la presión selectiva otorgando mayores oportunidades de supervivencia a los individuos con peor calidad.

Existen otros algoritmos basados en este concepto, como por ejemplo *Multi-Niche Crowding (MNC)* que modifica tanto el sistema de selección como el de sustitución. Este método introduce una mayor presión selectiva efectuando torneos entre grupos de individuos para seleccionar a los padres. En lugar de utilizar a los padres en el reemplazo, se utilizan varios grupos de individuos de la población inicial. De estos individuos, se selecciona a los individuos que compartan más características con el hijo en cada grupo.

5.3.4. Competición de especies (*Species competition*)

Los métodos de competición de especies o s*pecies competition* se basan en la división de la población en especies y la localización del mejor individuo de cada una de ellas en función de su calidad. Este individuo, denominado semilla, se utiliza para distribuir al resto los individuos de la especie en función de su proximidad.

En este algoritmo, una especie, es una subpoblación de individuos en la que la distancia entre dos individuos cualesquiera es menor que una magnitud, parámetro del algoritmo, denominada radio de la especie, σ_s. Dentro de cada especie, existe un individuo con mayor calidad que el resto denominado semilla.

Una vez determinadas las especies y sus semillas, se lleva a cabo los procesos de selección, cruce y mutación, generándose un conjunto de hijos.

En el proceso de sustitución, cada semilla, es comparada con los individuos hijos que se encuentren dentro de su radio de especie σ_s, si existe algún individuo de peor calidad en el conjunto es remplazado por la semilla.

Como cada semilla pertenece a una especie diferente, es decir a un conjunto de individuos que son muy diferentes al resto, se mantiene la diversidad.

5.4. Algoritmos multimodales estudiados

El objetivo de este TFM es comparar la bondad de tres algoritmos genéticos multimodales frente a un problema real. Los algoritmos elegidos para el estudio son dos del tipo aclarado de la población (Clearing Simple y Restricted Competition Selection) y uno del tipo competición de especies (Species Conserving Genetic Algorithm). En este apartado, se describirá el funcionamiento de cada uno de estos algoritmos haciendo uso de diagramas de flujo y pseudocódigo.

El problema de optimización que tratan de resolver los algoritmos consiste en elegir los parámetros óptimos de dos TMDs para minimizar el máximo global de todas las FRFs en aceleración de un edificio de dos plantas. Como se vio en el Capítulo 2, un TMD tiene tres variables sobre las que el diseñador puede actuar: masa m_j, rigidez k_j y constante de amortiguamiento c_j, haciendo referencia el subíndice j a que TMD se refiere. Además, como cada TMD debe instalarse en una planta concreta, a las tres variables del TMD debe añadirse la posición p_j, que indica la planta donde se instalará el TMD.

Teniendo en cuenta que para cada TMD podemos actuar sobre cuatro variables, una solución candidata al problema o individuo constará de ocho variables, cuatro por TMD, como se muestra en la Figura 5.5. Debido a que seis de las variables están definidas como números reales, la *codificación del problema* será real.

| m_1 | m_2 | k_1 | k_2 | c_1 | c_2 | p_1 | p_2 |

Figura 5.5: Codificación de un individuo (solución al problema)

Las restricciones de diseño para las variables imponen que, la suma de m_1 y m_2 no sea mayor que 0.15, las constantes de rigidez k_1 y k_2 deben moverse entre 0.5 y 6 y las constantes de amortiguamiento c_1 y c_2 entre 0 y 0.5. Por último, las posiciones p_1 y p_2, solo podrán tomar los valores enteros 1 y 2, dado que se trata de un edificio de dos plantas y los TMDs pueden instalarse bien el piso 1 o bien el piso 2.

Como se expuso en el capítulo anterior, un algoritmo genético está formado por cinco operaciones básicas: generación de la población inicial, evaluación, selección, cruce y mutación y sustitución. Los tres algoritmos estudiados comparten los procesos generación de la población inicial, evaluación, cruce y mutación, encontrándose las principales diferencias en los procesos de selección y sustitución.

La *generación de la población inicial* se realiza de manera aleatoria mediante una función que obtiene cada individuo teniendo en cuenta las restricciones de cada variable del problema. Al emplear la generación aleatoria, los individuos que forman parte de la primera generación están distribuidos con la misma densidad dentro del espacio de búsqueda, otorgando mayor posibilidad al algoritmo de encontrar todos los óptimos del problema.

Como salida, la función retorna una matriz de 8xN individuos, siendo N el tamaño de la población. (Figura 5.6)

$$\begin{matrix} {m_1}^1 & {m_2}^1 & {k_1}^1 & {k_2}^1 & {c_1}^1 & {c_2}^1 & {p_1}^1 & {p_2}^1 \\ {m_1}^2 & {m_2}^2 & {k_1}^2 & {k_2}^2 & {c_1}^2 & {c_2}^2 & {p_1}^2 & {p_2}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ {m_1}^i & {m_2}^i & {k_1}^i & {k_2}^i & {c_1}^i & {c_2}^i & {p_1}^i & {p_2}^i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ {m_1}^N & {m_2}^N & {k_1}^N & {k_2}^N & {c_1}^N & {c_2}^N & {p_1}^N & {p_2}^N \end{matrix}$$

Figura 5.6: Población inicial generada

La *evaluación* se lleva a cabo por una función que calcula el máximo global de las FRFs de los pisos 1 y 2, elaborada por (Magdaleno, 2017). Este valor, representa la calidad o adaptación del individuo al problema y por simplicidad a la hora de desarrollar los algoritmos se almacena añadiendo una novena columna junto a las distintas variables del individuo, como puede verse en la Figura 5.7, denotando la calidad o fitness del individuo i-esimo como f^i.

$$\begin{matrix} {m_1}^1 & {m_2}^1 & {k_1}^1 & {k_2}^1 & {c_1}^1 & {c_2}^1 & {p_1}^1 & {p_2}^1 & f^1 \\ {m_1}^2 & {m_2}^2 & {k_1}^2 & {k_2}^2 & {c_1}^2 & {c_2}^2 & {p_1}^2 & {p_2}^2 & f^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ {m_1}^i & {m_2}^i & {k_1}^i & {k_2}^i & {c_1}^i & {c_2}^i & {p_1}^i & {p_2}^i & f^i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ {m_1}^N & {m_2}^N & {k_1}^N & {k_2}^N & {c_1}^N & {c_2}^N & {p_1}^N & {p_2}^N & f^N \end{matrix}$$

Figura 5.7: Población de individuos tras la evaluación

Los *operadores cruce* que forman parte del estudio están basados en entornos. Como se expuso en el Capítulo 4, los operadores cruce basados en entornos son válidos cuando trabajamos con codificación real y eligen los nuevos genes de los hijos a partir de valores tomados del entorno de los genes de los padres. Para cada algoritmo, se utilizarán tres tipos de cruce: BLX, SBX y PNX. La diferencia entre estos tres tipos de cruce radica en cómo se genera el entorno de donde se obtendrán los genes que formarán los hijos.

Como se verá más adelante, que se utilicen tres tipos de cruce no significa que estos funcionen simultáneamente, si no que a partir de cada tipo de cruce se genera una versión del algoritmo diferente. Estas versiones serán ejecutadas una por una para después evaluar el beneficio de utilizar un tipo de cruce u otro con un determinado algoritmo.

Como *operador mutación*, los algoritmos emplean la mutación no uniforme, donde los genes mutados toman un valor comprendido entre los límites inferior y superior de la variable. Una ampliación de este concepto puede encontrarse en el Capítulo 4: Operadores mutación.

A modo de resumen, los tres algoritmos que forman parte del estudio utilizan codificación real, generación de la población inicial aleatoria, operadores cruce basados en entornos y mutación no uniforme.

Una vez expuesta la parte común en los tres algoritmos, se detallarán los procesos de selección y sustitución llevados a cabo en cada algoritmo. Estos procesos tienen en común que su misión es mantener la diversidad en todo momento para así poder encontrar el mayor número de óptimos globales y locales posibles.

5.4.1. Clearing

Como se comentó anteriormente, los algoritmos de este tipo utilizan el concepto de nicho para llevar a cabo una limpieza de la población eliminando parte de los individuos muy similares entre si, es decir los individuos pertenecientes al mismo nicho. Los individuos que sobreviven en cada nicho son los mejor adaptados y su cantidad se controla con un parámetro del algoritmo denominado *tamaño de nicho*, *k*.

El *radio de nicho* o σ_{share} es el segundo parámetro del algoritmo, utilizado para determinar si dos individuos pertenecen al mismo nicho. Para ello se evalúa la distancia entre ellos $d(i,j)$ y si esta distancia es menor que el radio de nicho σ_{share}, los individuos se encuentran en el mismo nicho. En nuestro caso la función *distancia* $d(i,j)$ es la distancia euclídea entre dos puntos, pero pueden usarse otras distancias a conveniencia.

Siguiendo esta forma de proceder, el algoritmo Clearing, a partir de la población original *P(t)* de tamaño *N*, obtiene una población *P'(t)* de tamaño *N'* formada por los individuos dominantes de cada nicho. Este proceso se lleva a cabo antes del proceso de selección que se efectuará únicamente sobre la población de individuos dominantes *P'(t)*.

La función que obtiene los individuos dominantes de cada nicho puede verse en la Figura 5.8:

```
Function Clearing (σ_share, k)
    OrdenarPoblacion(P)
    For i=1 to N
        If ( f^i > 0)
            nGanadores = 1
            For j=i+1 to N
                If [(f^i > 0) & (d(i,j) < σ_share)]
                    If (nGanadores < k)
                        nGanadores = nGanadores + 1
                    else
                        f^i = 0
                    Endif
                Endif
            Endfor
        Endif
    Endfor
End
```

Figura 5.8: Pseudocódigo del proceso Clearing

Tras aplicar el proceso de limpieza o clearing se lleva a cabo la selección por torneo sobre la población *P'(t)* para obtener los individuos padres. Una vez elegidos los padres, tienen lugar los procesos de cruce y mutación obteniéndose los individuos hijos.

En el proceso de sustitución, las soluciones de la población *P '(t)* con una calidad superior a la media de la calidad original de la población *P(t)*, antes del proceso de limpieza, son seleccionadas para sobrevivir y pasar directamente a la siguiente generación. El resto de posiciones hasta llegar al tamaño de la población, *N*, son ocupadas por los hijos mejor de mayor calidad.

El proceso completo representado mediante un diagrama de flujo puede verse en la Figura 5.8:

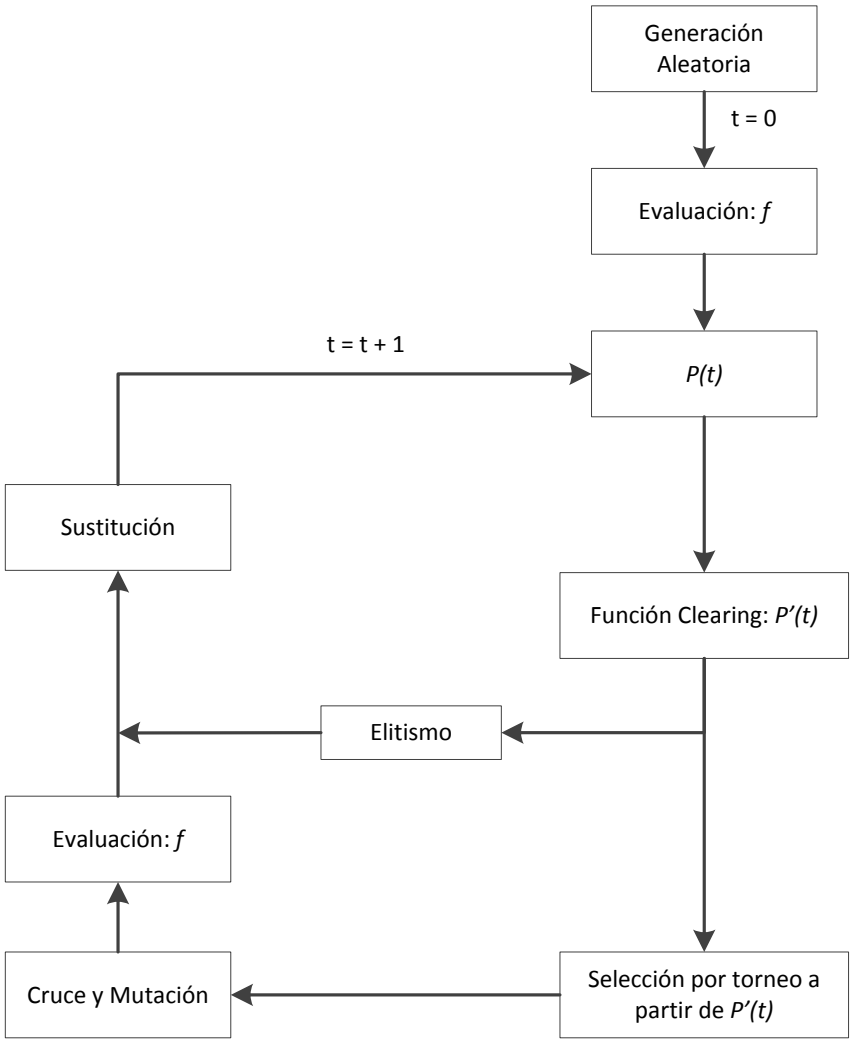

Figura 5.9: Diagrama de flujo del algoritmo Clearing

Este algoritmo ha sido desarrollado por (Pétrowski, 1996) estudiando en trabajos posteriores (Pétrowski, 1997) la bondad del proceso clearing frente a distintos problemas de optimización.

5.4.2. Restricted Competition Selection (RCS)

Este método, desarrollado por (Lee et al., 1999), utiliza el principio de clearing o limpieza de la población en el proceso de sustitución en lugar de en el proceso de selección.

De esta manera, todos los individuos de la población entran en el proceso de selección, agrupándose aleatoriamente en parejas para reproducirse. Así, el algoritmo RCS elimina la presión sobre la selección.

Sin embargo, el proceso de sustitución, trata de limitar la competencia entre individuos de diferentes nichos, utilizando el proceso clearing donde solo los mejores individuos de cada nicho sobreviven.

Para ello, se utilizan dos grupos: el conjunto élite y el conjunto competición. El conjunto elite se obtiene a partir de la población original *P(t)* con el objetivo de mantener las soluciones de calidad encontradas en generaciones anteriores. Para obtener estas soluciones, se lleva a cabo el proceso clearing sobre la población *P(t)* y se conservan los mejores *M* individuos, siendo *M* la dimensión del conjunto élite.

El conjunto competición, se forma a partir de la unión del conjunto élite y la población de hijos creados tras el proceso de cruce y mutación, resultando un conjunto de dimensión *N + M*. De este grupo, se obtienen los *N* individuos que formaran parte de la siguiente generación. Estos *N* individuos son los individuos dominantes de cada nicho clasificados de mejor a menor calidad.

El proceso completo puede verse en el diagrama de flujo de la Figura 5.10:

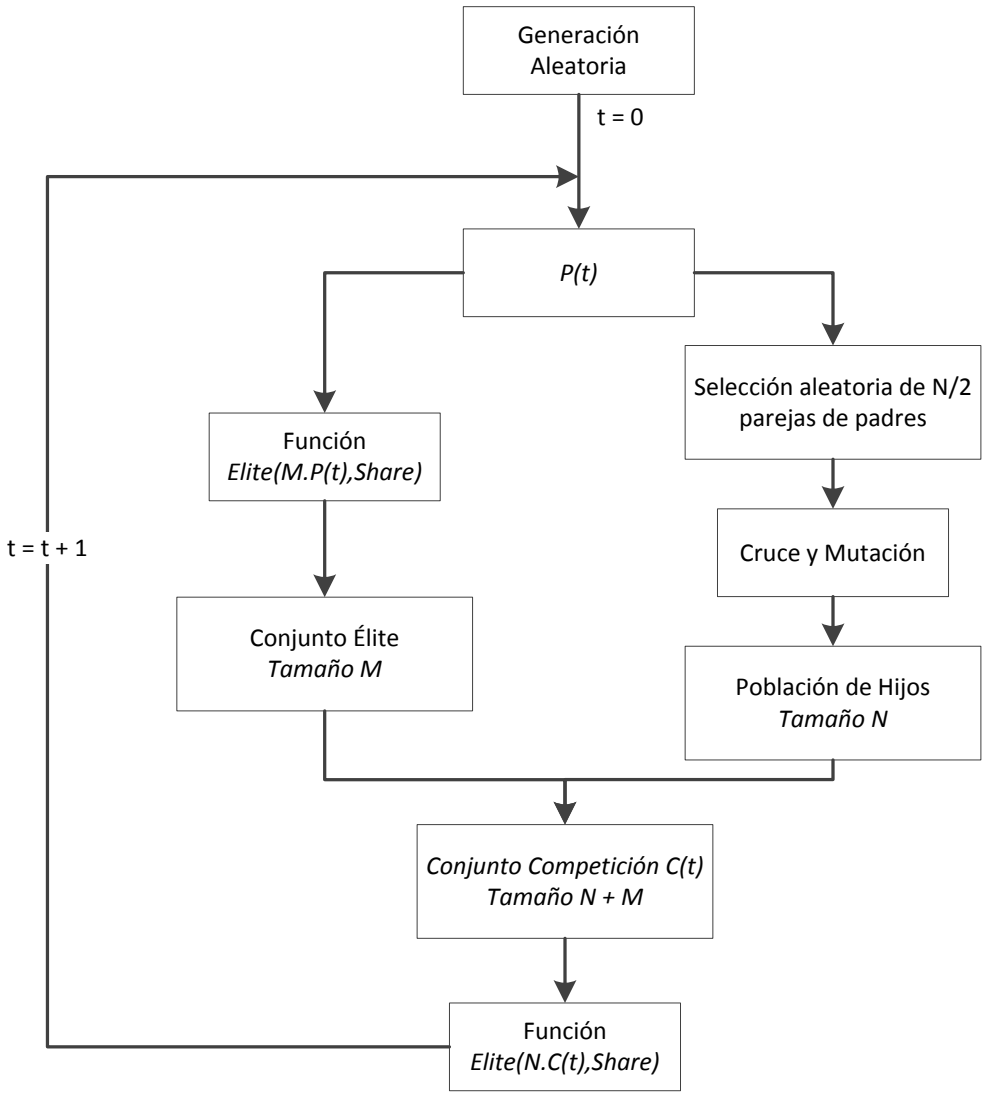

Figura 5.10: Diagrama de flujo del algoritmo RCS

Donde la Función *Elite* obtiene el mejor individuo de cada nicho en SET hasta completar una población de tamaño A. El pseudocódigo de esta función se muestra en la Figura 5.11:

```
Function Elite (A, SET, σ_share)
    OrdenarPoblacion(SET)
    MejorIndividuo(SET) → ConjuntoElite
    nIndividuosElite = 1

    For i=2 to size(SET)
        Mejor = True
        For j=1 to nIndividuosElite
            If (d(i,j) < σ_share)
                Mejor = False
            Endif
        Endfor

        If Mejor = True
            Individuo(i) → ConjuntoElite
            nIndividuosElite++
        Endif

        If nIndividuosElite == A
            i = size(SET)
        Endif
    Endfor
End
```

Figura 5.11: Pseudocódigo del proceso Elite

5.4.3. Species Conserving Genetic Algorithm (SCGA)

Este algoritmo, desarrollado originalmente por (Lee et al., 2002), se basa en la división de la población en especies y la localización de la mejor solución de cada una de ellas en función de su calidad, a la que llamaremos semilla. Los individuos pertenecientes a la siguiente generación se distribuyen en según su proximidad a las semillas de la generación anterior. A partir de esta distribución se lleva a cabo el proceso de sustitución.

La primera tarea del algoritmo consiste en identificar las especies presentes en la población, así como sus semillas. Estas semillas constituirán el conjunto semilla, denotado por X_s. Para lograrlo, se ordena la población *P(t)* de mayor a menor calidad eligiendo el primer individuo como la semilla de la primera especie. Una vez identificado este individuo comienza un proceso iterativo donde se comprueba si el siguiente individuo elegido pertenece a alguna especie. Si no pertenece a ninguna especie, el individuo pasará a formar parte del conjunto semilla.

Para que dos individuos pertenezcan a la misma especie, la distancia entre ellos debe ser menor que un radio, denominado *radio de nicho* o σ_{share}, de forma similar a como trabajan los dos algoritmos comentados anteriormente.

Una vez localizadas las especies y sus individuos dominantes o semillas, la población original pasa al proceso de selección. En este caso se lleva a cabo una selección por torneo mediante la competición de dos individuos de la población seleccionados de manera aleatoria. El criterio de competición es únicamente su adaptación al problema o fitness.

En este punto el algoritmo cuenta con una población de individuos padres procedentes de la selección por torneo y el conjunto semilla al que pertenecen los individuos dominantes. El

siguiente paso es utilizar los operadores cruce y mutación sobre el conjunto de individuos padres para generar el conjunto de individuos hijos o siguiente generación, *P(t+1)*.

En el proceso de sustitución, se lleva a cabo el proceso de *conservación de las especies*, que consiste en distribuir los hijos generados, *P(t+1)*, en torno a las semillas encontradas en la generación anterior, X_s. Después de este proceso de distribución, obtenemos una serie de individuos de la población *P(t+1)* que pertenecen a una especie concreta y otra serie de individuos que no pertenecen a ninguna especie. Si la semilla, es mejor que alguno de los nuevos individuos de su especie, la semilla pasa a formar parte de la generación siguiente remplazando al peor individuo.

El diagrama de flujo del algoritmo puede verse en la Figura 5.11.

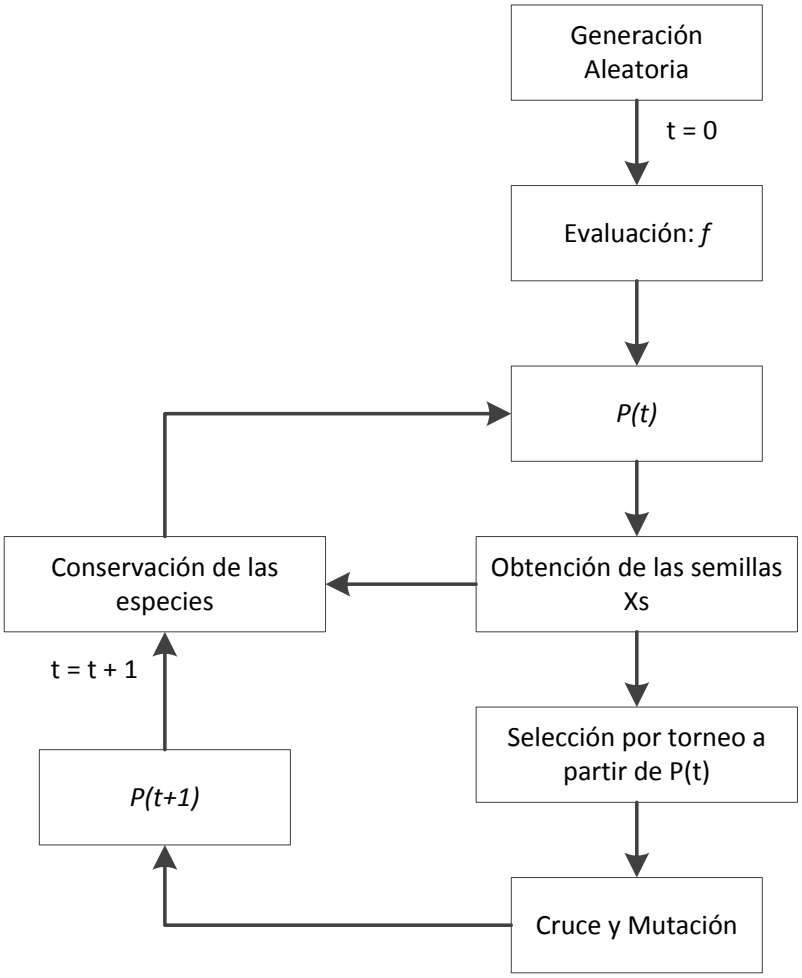

Figura 5.12: Diagrama de flujo del algoritmo SCGA

6. ESTUDIO EXPERIMENTAL

Este capítulo tiene como objetivo exponer la metodología experimental y resultados obtenidos en este TFM. En primer lugar, se detallará como se han llevado a cabo los experimentos para cada uno de los tres algoritmos. A continuación, a partir de los resultados obtenidos, se comprobará cuáles son los parámetros más adecuados para cada algoritmo frente al problema que se está estudiando. Una vez determinados los parámetros más adecuados para cada algoritmo, se comparará la bondad de cada uno de ellos frente a este tipo de problemas. Por último, se compararán los tres algoritmos genéticos multimodales estudiados frente a un algoritmo genético simple.

6.1. Metodología experimental

Como ya se ha comentado anteriormente, el estudio se lleva a cabo sobre un modelo de un edificio de dos plantas donde se quiere instalar dos TMDs. Sobre este modelo, se han llevado a cabo una serie de optimizaciones con los distintos algoritmos expuestos en el Capítulo 5, variando sus parámetros característicos y el operador cruce.

Los parámetros que variarán en cada algoritmo y sus valores son los siguientes:

Clearing: el número de ganadores permitidos en cada nicho, *k*, tomará dos valores 1 o 5. Para el radio de nicho, σ_{share}, se analizarán los valores de 0, 10, 20 y 30.

RCS: el tamaño del conjunto *M*, tomará dos valores 50 o 100. Para el radio de nicho, σ_{share}, se analizarán los valores de 0, 10, 20 y 30.

SCGA: Para el radio de nicho, σ_{share}, se analizarán los valores de 0, 10, 20 y 30, al igual que en los otros dos algoritmos.

Los operadores cruce que forman parte del estudio son: BLX, PNX y SBX.

De esta manera, para cada algoritmo tenemos una serie de posibles combinaciones de parámetros y operador cruce representada en la Tabla 6.1.

La notación que se utilizará de aquí en adelante para referirnos a una configuración concreta es la siguiente:

- Clearing: CL – *Tipo de cruce* – *k* – σ_{share}
- RCS: RCS – *Tipo de cruce* – *M* – σ_{share}
- SCGA: SCGA – *Tipo de cruce* – σ_{share}

Además, los parámetros habituales de cualquier algoritmo genético, tamaño de la población, probabilidad de cruce y probabilidad de mutación son compartidos por todos los experimentos llevados a cabo y toman los valores de:

- Tamaño de la población: 100 individuos
- Probabilidad de cruce: 90 %
- Probabilidad de mutación: 10%

El número máximo de generaciones para todas las ejecuciones es 100000, valor suficientemente elevado como para que los algoritmos converjan hacia un óptimo.

Por motivos estadísticos, se han realizado 20 ejecuciones sobre cada posible configuración del algoritmo para más adelante estudiar la calidad de los resultados a partir de la media de los óptimos obtenidos por cada configuración.

Clearing			RCS			SCGA	
Cruce	***k***	σ_{share}	**Cruce**	***M***	σ_{share}	**Cruce**	σ_{share}
BLX	1	0	BLX	50	0	BLX	0
BLX	5	0	BLX	100	0	BLX	10
BLX	1	10	BLX	50	10	BLX	20
BLX	5	10	BLX	100	10	BLX	30
BLX	1	20	BLX	50	20	PNX	0
BLX	5	20	BLX	100	20	PNX	10
BLX	1	30	BLX	50	30	PNX	20
BLX	5	30	BLX	100	30	PNX	30
PNX	1	0	PNX	50	0	SBX	0
PNX	5	0	PNX	100	0	SBX	10
PNX	1	10	PNX	50	10	SBX	20
PNX	5	10	PNX	100	10	SBX	30
PNX	1	20	PNX	50	20		
PNX	5	20	PNX	100	20		
PNX	1	30	PNX	50	30		
PNX	5	30	PNX	100	30		
SBX	1	0	SBX	50	0		
SBX	5	0	SBX	100	0		
SBX	1	10	SBX	50	10		
SBX	5	10	SBX	100	10		
SBX	1	20	SBX	50	20		
SBX	5	20	SBX	100	20		
SBX	1	30	SBX	50	30		
SBX	5	30	SBX	100	30		

Tabla 6.1: Combinaciones de parámetros y cruces estudiadas

6.2. Influencia de los parámetros y tipo de cruce

El objetivo de este apartado es analizar la influencia de los parámetros característicos de cada algoritmo sobre su eficacia y efectividad en función de dos variables: la calidad de la solución y el número de óptimos encontrados. Para ello, se analizarán los siguientes valores:

Valor medio: calculado como la media aritmética de la mejor solución alcanzada en cada una de las 20 ejecuciones.

Número medio de óptimos encontrados: calculado como la media aritmética de los óptimos encontrados en cada una de las 20 ejecuciones.

Debido al problema estudiado, el número máximo de óptimos es conocido, tomando el valor de 4, uno para cada posible modo de instalación de los TMDs atendiendo a su localización. Los dos en una planta (primera o segunda) o cada uno en plantas diferentes.

Además, se sabe que la mejor manera de instalar los dos TMDs es cada uno en una planta, así, que esta será el modo estudiado para determinar la influencia de los parámetros.

A partir de estos criterios y los resultados experimentales, que pueden verse en los Anexos I, II y II, se han confeccionado las Tablas 6.2, 6.3 y 6.4, donde para cada configuración del algoritmo se muestran los resultados obtenidos.

Para el algoritmo Clearing los resultados se resumen en la Tabla 6.2:

CLEARING		
Configuración	Valor Medio	Nº medio de óptimos
CL-BLX-1-0	8,217641869	1
CL-BLX-5-0	8,217298802	1
CL-BLX-1-10	8,345563934	3,8
CL-BLX-5-10	8,23070349	2,75
CL-BLX-1-20	8,339104884	4
CL-BLX-5-20	8,236456002	3,85
CL-BLX-1-30	8,255922719	4
CL-BLX-5-30	8,24261531	4
CL-PNX-1-0	8,216371129	1
CL-PNX-5-0	8,216505396	1
CL-PNX-1-10	8,366274332	3,35
CL-PNX-5-10	8,230767518	2,8
CL-PNX-1-20	8,309847076	4
CL-PNX-5-20	8,22760581	3,8
CL-PNX-1-30	8,256303314	4
CL-PNX-5-30	8,345123007	4
CL-SBX-1-0	8,219229066	1
CL-SBX-5-0	8,217994871	1
CL-SBX-1-10	8,56269631	3,95
CL-SBX-5-10	8,403184721	2,8
CL-SBX-1-20	8,414233824	4
CL-SBX-5-20	8,253139238	4
CL-SBX-1-30	8,273030066	3,9
CL-SBX-5-30	8,227771145	3,9

Tabla 6.2: Clearing - Valor medio y Nº de óptimos encontrados

A la vista de los resultados, la mejor selección de parámetros para el algoritmo Clearing es una combinación de operador cruce PNX, tamaño de nicho *k=1* y radio de nicho, σ_{share}=0. Sin embargo, esta combinación solo es capaz de encontrar el óptimo global del problema, no llegando a encontrar en ningún caso otros óptimos locales. Esto es debido a que, al utilizar radio de nicho igual a cero, el algoritmo Clearing favorece la explotación frente a la exploración del espacio de búsqueda.

En el Anexo IV, se muestra la evolución del *valor medio* y el *Nº medio de óptimos*, frente al radio de nicho para los tres operadores cruces estudiados y los dos tamaños de nicho *k=1* o *k=5*. A modo de ejemplo, en las figuras 6.1 y 6.2 muestran esta evolución para el operador BLX y los tamaños de nicho *k=1* o *k=5*. El comportamiento para los otros operadores cruce es similar y puede consultarse en el Anexo IV.

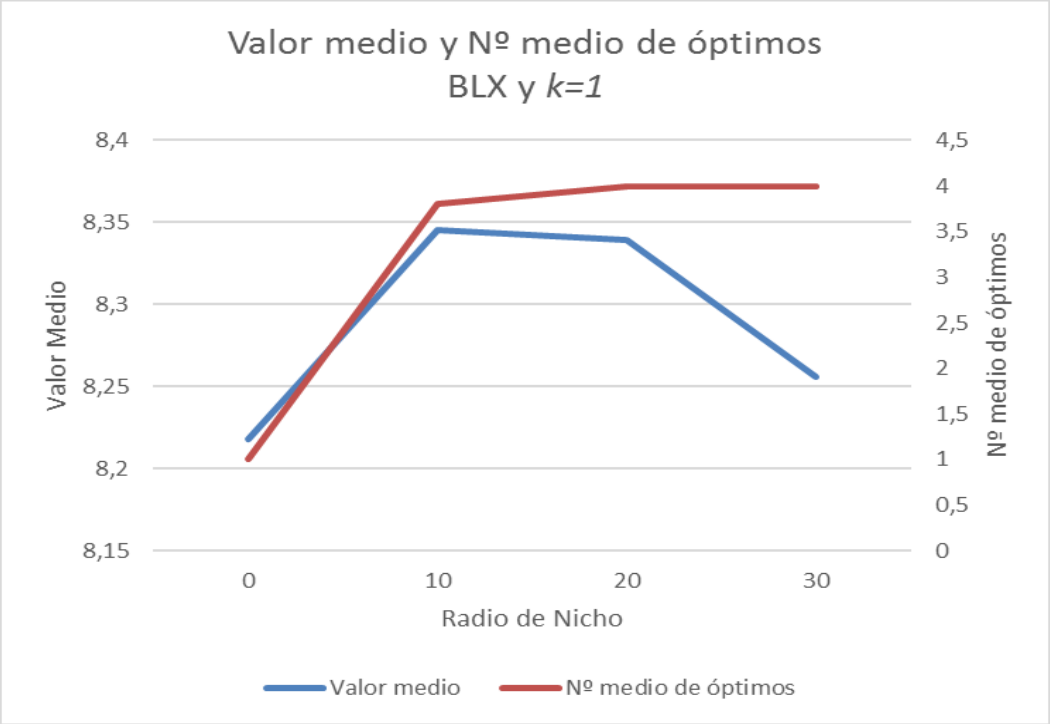

Figura 6.1: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para BLX y k=1

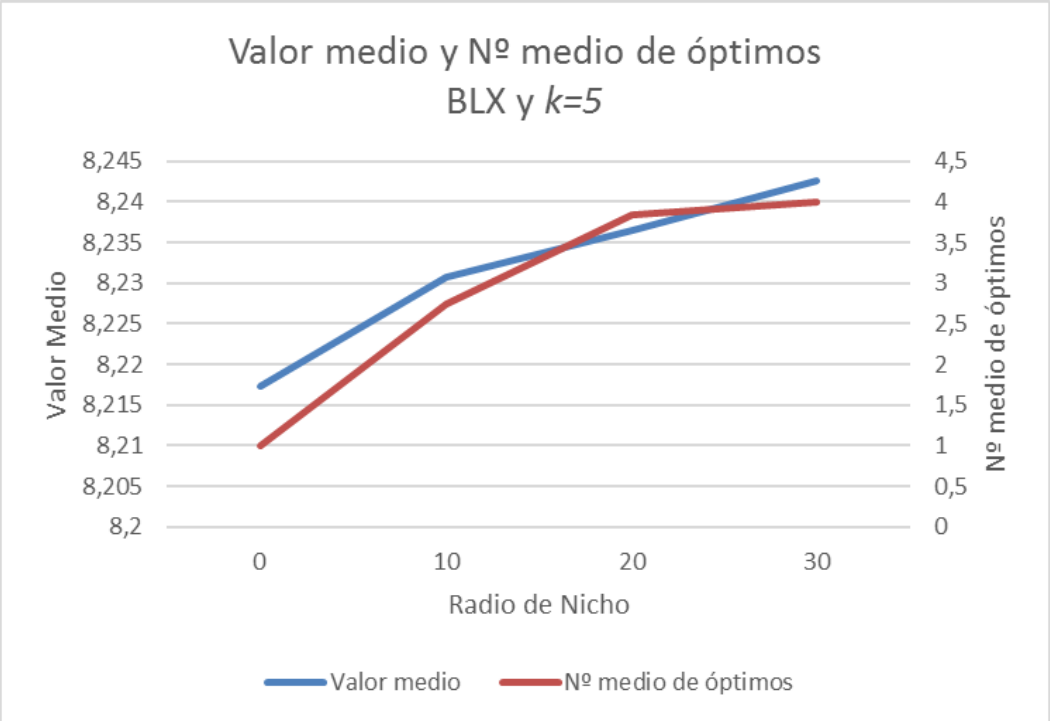

Figura 6.2: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para BLX y k=5

Para tamaño de nicho *k=1*, si se desea obtener la totalidad de los óptimos del problema ha de utilizarse un valor para el radio de nicho elevado. Por otro lado, si se desea obtener un buen compromiso entre cantidad de soluciones encontradas y calidad de estas, es decir entre exploración y explotación, se deberá elegir un valor para el tamaño de nicho de 30 o superior.

Para tamaño de nicho *k=5*, el comportamiento es diferente. Aumentar el radio de nicho hace que se favorezca la exploración del espacio de búsqueda encontrando todas las soluciones, sin embargo, empeora la calidad de las soluciones encontradas.

A modo de resumen, para el algoritmo Clearing, el mejor operador cruce es el *PNX*. Si se desea favorecer la explotación y obtener la mejor solución posible al problema debemos elegir un radio de nicho igual a 0. Si se desea un buen compromiso entre exploración y explotación, debemos elegir un radio de nicho 30 y tamaño de nicho igual a 1.

Para el algoritmo RCS los resultados se resumen en la Tabla 6.3:

RCS		
Configuración	Valor Medio	Nº medio de óptimos
RCS-BLX-50-0	8,217053991	1
RCS-BLX-100-0	8,216805412	1
RCS-BLX-50-10	8,736873739	4
RCS-BLX-100-10	8,65902253	4
RCS-BLX-50-20	8,764523723	4
RCS-BLX-100-20	8,997915007	4
RCS-BLX-50-30	9,181758499	4
RCS-BLX-100-30	9,181758499	4
RCS-PNX-50-0	8,216309648	1
RCS-PNX-100-0	8,216285589	1
RCS-PNX-50-10	8,574594226	4
RCS-PNX-100-10	8,624257702	4
RCS-PNX-50-20	8,530799647	4
RCS-PNX-100-20	9,073496503	4
RCS-PNX-50-30	8,608109491	4
RCS-PNX-100-30	8,608109491	4
RCS-SBX-50-0	8,221249323	1
RCS-SBX-100-0	8,221675685	1
RCS-SBX-50-10	8,972667263	4
RCS-SBX-100-10	8,796860465	4
RCS-SBX-50-20	8,918919842	4
RCS-SBX-100-20	8,878702882	4
RCS-SBX-50-30	9,818916402	4
RCS-SBX-100-30	9,818916402	4

Tabla 6.3: RCS - Valor medio y Nº de óptimos encontrados

Del mismo modo que para el algoritmo Clearing, en el RCS las mejores soluciones se obtienen utilizando el operador cruce PNX y radio de nicho igual a cero. Como contrapartida, el algoritmo solo encuentra una solución de las cuatro posibles.

En el Anexo V, se muestra la evolución del valor medio y el Nº medio de óptimos, frente al radio de nicho para los tres operadores cruces estudiados y los dos tamaños del conjunto *M*. A modo de ejemplo, las figuras 6.3 y 6.4 muestran esta evolución para el operador BLX y los tamaños del conjunto *M=50 y M=100*. El comportamiento para los otros operadores cruce es similar y puede consultarse en el Anexo V.

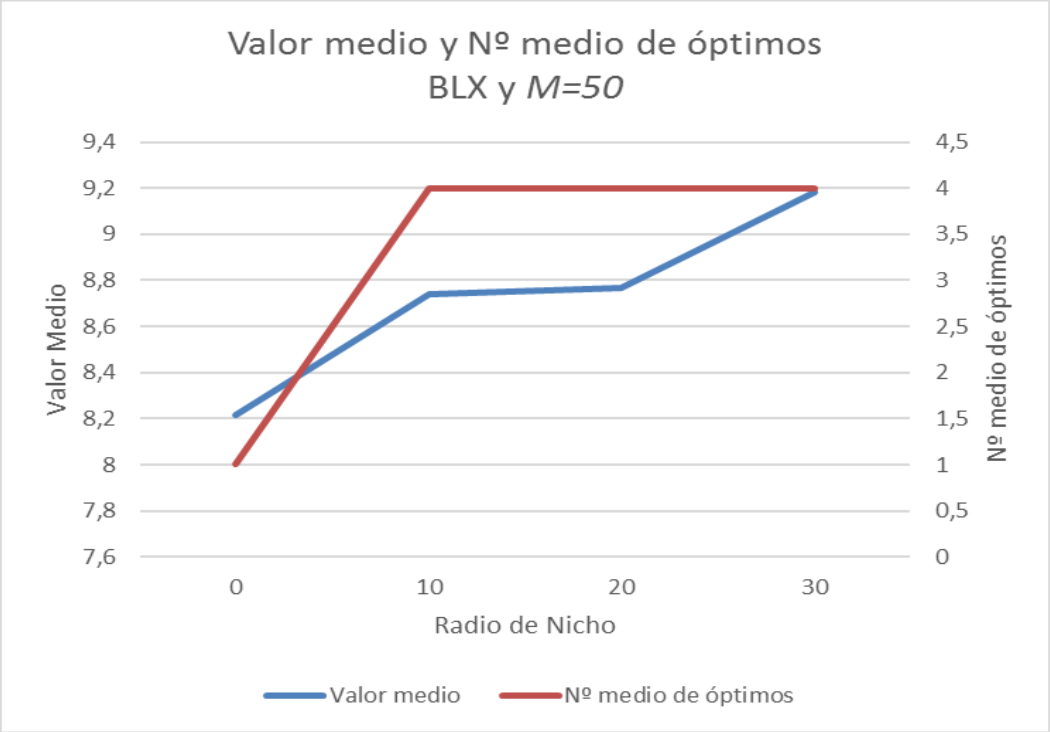

Figura 6.3: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para BLX y M=50

La principal conclusión que puede extraerse de las figuras 6.3 y 6.4, es que al aumentar el radio de nicho la calidad de la solución empeora notablemente. Este hecho ha sido observado para todos los operadores cruce estudiados y ambos tamaños del conjunto *M*.

Por otro lado, al aumentar el radio de nicho, el algoritmo favorece la exploración del espacio de búsqueda, obteniendo todas las soluciones al problema.

Si se desea obtener un buen compromiso entre exploración y explotación, para el algoritmo RCS, se debe elegir un radio de nicho pequeño, pero no igual a cero. De esta manera, se obtendrán todas las soluciones al problema con una calidad aceptable.

En cuanto al tamaño del conjunto *M*, para la mayoría de ejecuciones se obtiene un mejor resultado utilizando un tamaño de conjunto *M* igual a 50. Además, trabajar con un tamaño más reducido de individuos reduce la carga computacional.

A modo de resumen, para el algoritmo RCS, el mejor operador cruce es el PNX. Si se desea favorecer la explotación y obtener la mejor solución posible al problema debemos elegir un radio de nicho igual a 0. Si se desea un buen compromiso entre exploración y explotación, debemos elegir un radio de nicho 10. Sobre el tamaño del conjunto *M*, es más beneficioso trabajar con un tamaño de 50.

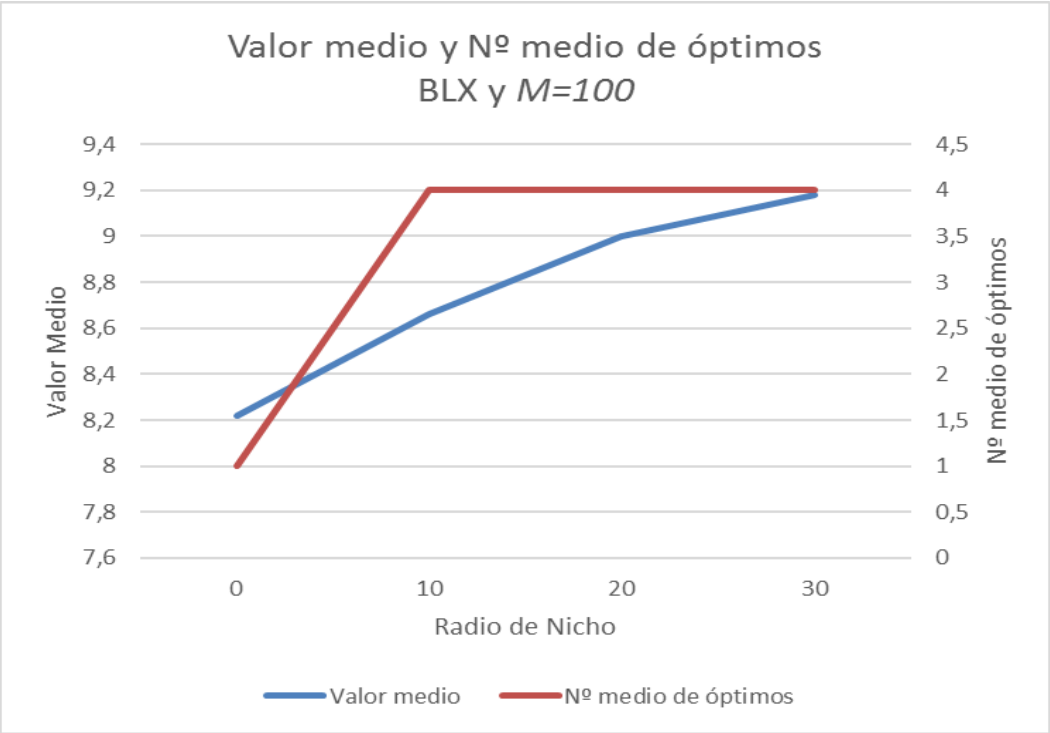

Figura 6.4: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para BLX y M=100

Para el algoritmo SCGA los resultados se resumen en la Tabla 6.3:

SCGA		
Configuración	Valor Medio	Nº medio de óptimos
SCGA-BLX-0	29,56668501	4
SCGA -BLX-10	8,638775106	4
SCGA -BLX-20	11,43523791	4
SCGA -BLX-30	10,39043158	4
SCGA -PNX-0	26,76673046	4
SCGA - PNX -10	8,349667091	4
SCGA - PNX -20	10,64259648	4
SCGA - PNX -30	10,50007562	4
SCGA - PNX -0	29,29843272	4
SCGA - PNX -10	8,746914503	4
SCGA - PNX -20	11,63078353	4
SCGA - PNX -30	12,7718794	4

Tabla 6.4: SCGA - Valor medio y Nº de óptimos encontrados

Al igual que para los dos algoritmos estudiados anteriormente, las mejores soluciones se obtienen utilizando el operador cruce PNX. Sin embargo, el radio de nicho en este caso ha de ser 10 y no 0. Además, este algoritmo encuentra en todas las ejecuciones las 4 soluciones del problema, favoreciendo la exploración del espacio de búsqueda en detrimento de la calidad de las soluciones encontradas.

En el Anexo VI, se muestra la evolución del valor medio y el Nº medio de óptimos, frente al radio de nicho para los tres operadores cruces estudiados. A modo de ejemplo, la figura 6.5 muestra esta evolución para el operador BLX. El comportamiento para los otros operadores cruce es similar y puede consultarse en el Anexo VI.

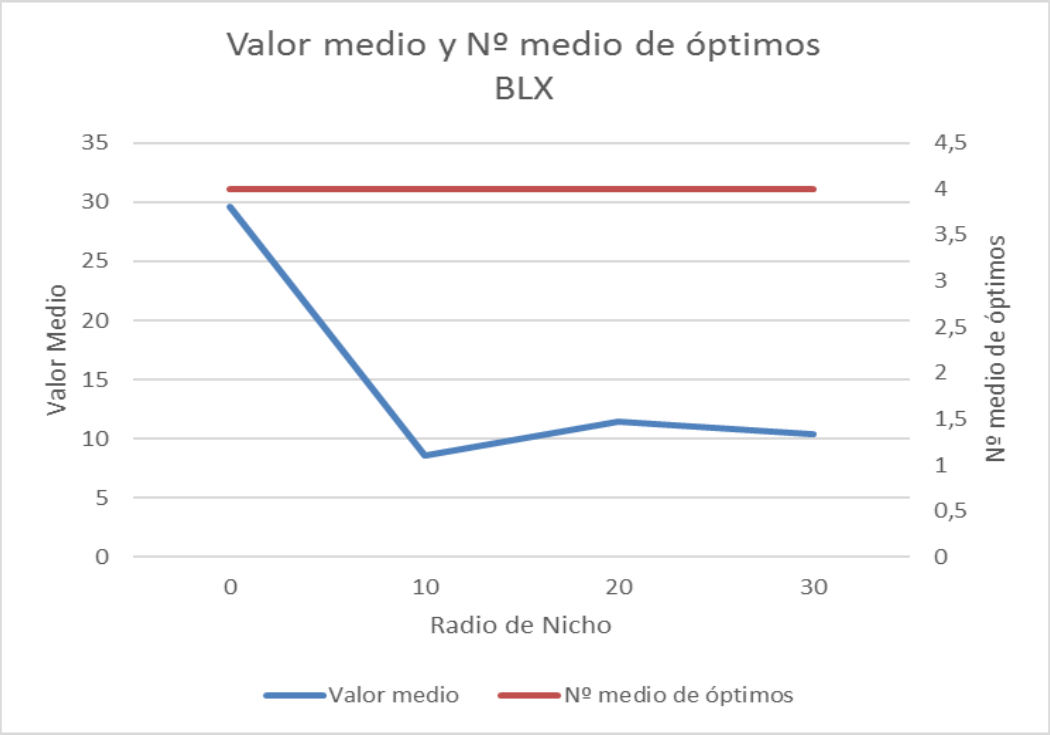

Figura 6.5: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para BLX

Como puede verse en la Figura 6.5, la mejor solución obtenida corresponde al radio de nicho igual a 10. Este hecho, ha sido observado para los tres operadores cruce estudiados y puede consultarse de forma gráfica en el Anexo VI.

De esta manera, para obtener los mejores resultados con el algoritmo SCGA, debe elegirse el operador cruce PNX con un radio de nicho igual a 10.

6.3. Estudio comparativo entre los tres algoritmos

Una vez determinada la influencia de los parámetros de los algoritmos sobre la calidad de los resultados y el número de soluciones encontradas, en este apartado se compararán los tres algoritmos que forman parte del estudio para determinar cuál de los tres funciona mejor frente a este tipo de problemas.

Para ello, conocidas las mejores configuraciones para cada algoritmo, tanto para favorecer la explotación como para obtener un buen compromiso entre exploración y explotación, se expondrán los resultados obtenidos a partir de esas configuraciones y se decidirá que algoritmo obtiene mejor resultados.

En las Tablas 6.5 y 6.6, se muestran las mejores configuraciones para cada uno de los tres algoritmos, tanto en términos de explotación (Tabla 6.5) como en términos de compromiso entre exploración y explotación (Tabla 6.6).

CONFIGURACIONES PARA FAVORECER LA EXPLOTACIÓN							
CLEARING			RCS			SCGA	
Cruce	***k***	σ_{share}	**Cruce**	***M***	σ_{share}	**Cruce**	σ_{share}
PNX	1	0	PNX	50	0	PNX	10

Tabla 6.5: Mejores configuraciones de los algoritmos para favorecer la explotación

COMPROMISO ENTRE EXPLORACIÓN Y EXPLOTACIÓN							
CLEARING			RCS			SCGA	
Cruce	***k***	σ_{share}	**Cruce**	***M***	σ_{share}	**Cruce**	σ_{share}
PNX	5	20	PNX	50	10	PNX	10

Tabla 6.6: Mejores configuraciones para obtener un compromiso entre exploración y explotación

A partir de estas configuraciones los resultados obtenidos se muestran en la siguiente serie de tablas. (Tablas 6.7, 6.8, 6.9: Resultados para favorecer la explotación. Tablas 6.10, 6.11, 6.9: Resultados de compromiso entre exploración y explotación).

CL-PNX-1-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,216435662	8,216371129	
Mínimo		8,21624327	8,216260291	
Desviación		2,016185379	1,8349863	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

Tabla 6.7: Clearing - Resultados obtenidos para PNX, radio 0, k = 1

RCS-PNX-50-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,216309648	8,216311958	
Mínimo		8,216252169	8,216253671	
Desviación		2,099254698	1,735327939	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

Tabla 6.8: RCS - Resultados obtenidos para PNX, radio 0, M = 50

A partir de los resultados obtenidos, puede decirse que tanto el Clearing como el RCS son buenos algoritmos para encontrar el óptimo global del problema. Ambos algoritmos pertenecen a la misma familia y presentan resultados similares, tanto en las medias como en los óptimos obtenidos. En cuanto al algoritmo SCGA, no llega a obtener resultados de tanta calidad como los dos anteriores. Concluimos con que el mejor algoritmo para obtener el óptimo global del problema es el RCS.

SCGA-PNX-10				
CONFIGURACIONES				
	11	12	21	22
Media	21,96608474	8,349667091	8,461097504	12,29286775
Mínimo	20,44249488	8,263966353	8,289725711	12,00138443
Desviación	7,502000162	2,847992559	2,886994581	4,193246863
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

Tabla 6.9: SCGA - Resultados obtenidos para PNX y radio 10

CL-PNX-5-20				
CONFIGURACIONES				
	11	12	21	22
Media	40,26775407	8,22760581	8,23555094	12,46675396
Mínimo	35,91467701	8,217388685	8,216545701	11,93601848
Desviación	12,0671891	2,586629446	2,647705201	4,027078079
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	3,8	2	4	0,523148364

Tabla 6.10: Clearing - Resultados obtenidos para PNX, radio 20, k = 5

RCS-PNX-50-10				
CONFIGURACIONES				
	11	12	21	22
Media	38,47036932	8,574594226	8,693273511	12,4858995
Mínimo	20,67481386	8,355981341	8,359827216	12,19239814
Desviación	12,46839857	2,757560856	2,795559441	4,014754196
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

Tabla 6.11: RCS - Resultados obtenidos para PNX, radio 10, M = 50

En el caso de buscar un buen compromiso entre exploración y explotación los tres algoritmos se comportan bien si se eligen los parámetros adecuados. El SCGA destaca por su capacidad para obtener todos los óptimos del problema con buena calidad, a diferencia del Clearing o RCS donde las medias para la configuración 11 del problema no son buenas.

De esta manera, si se desea obtener el óptimo global del problema sin importar los demás óptimos locales, el mejor algoritmo es el RCS con radio 0 y *M=50.* Si se desea obtener todos los óptimos del problema con una calidad aceptable, el mejor algoritmo es el SCGA con radio 10. Para cualquiera de los objetivos, el mejor operador cruce en todos los casos es el operador PNX.

6.4. Estudio comparativo frente a un AG simple

En este apartado se compararán los tres algoritmos multimodales estudiados frente a un algoritmo genético simple. Los resultados del algoritmo genético simple para los tres operadores cruce estudiados se resumen a continuación (Tablas 6.12, 6.13 y 6.14):

AG Simple - SBX				
CONFIGURACIONES				
	11	12	21	22
Media	41,25152059	10,43356617	24,8243367	28,42020904
Mínimo	39,13783998	8,216407662	8,216355881	27,12150684
Desviación	14,49145002	4,33484435	8,720165758	10,19840433
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	3,3	2	4	0,571240571

Tabla 6.12: AG Simple – Resultados para operador cruce SBX

AG Simple - PNX				
CONFIGURACIONES				
	11	12	21	22
Media	40,21460569	20,78291655	13,14909806	27,11209216
Mínimo	28,19367738	8,216505348	8,216624191	21,78351227
Desviación	13,85721485	7,957457733	5,646739866	9,752617735
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	3,25	2	4	0,550119604

Tabla 6.13: AG Simple – Resultados para operador cruce PNX

AG Simple - SBX				
CONFIGURACIONES				
	11	12	21	22
Media	41,04133686	15,31912122	8,216457362	28,03487297
Mínimo	34,11599861	8,216378807	8,21633091	24,66821546
Desviación	14,10742434	6,150368972	2,174825295	10,07276362
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	3,05	2	4	0,510417786

Tabla 6.14: AG Simple – Resultados para operador cruce BLX

En este caso, el operador cruce que da mejores resultados es el SBX, encontrando uno de los óptimos globales al problema la mayoría de las ejecuciones. Sin embargo, los óptimos locales son de peor calidad en todos los casos. Si se utilizan cualquiera de los otros dos operadores cruce, los resultados empeoran notablemente.

A modo de ejemplo, en la Tabla 6.15, se muestra un estudio comparativo entre un algoritmo multimodal (SCGA-PNX-10) y los tres algoritmos genéticos simples.

AG Multimodal frente AG Simples				
	CONFIGURACIONES			
	11	12	21	22
SCGA-PNX-10	21,96608474	8,349667091	8,461097504	12,29286775
AG Simple - SBX	41,25152059	10,43356617	24,8243367	28,42020904
AG Simple - PNX	40,21460569	20,78291655	13,14909806	27,11209216
AG Simple - BLX	41,04133686	15,31912122	8,216457362	28,03487297

Tabla 6.15: Estudio comparativo entre un AG Multimodal y AG Simples

Como puede verse, el algoritmo genético multimodal, obtiene todos los óptimos del problema con buena calidad. Sin embargo, los algoritmos genéticos simples, en la mayoría de los casos no obtienen ninguno de los óptimos con suficiente calidad.

7. CONCLUSIONES Y LINEAS FUTURAS

Este capítulo tiene como objetivo exponer algunas de las conclusiones a las que se ha llegado realizando este TFM. Además, también se desarrollará un breve apartado sobre líneas futuras que pueden servir de base para nuevos trabajos.

7.1. Conclusiones

A partir de lo expuesto en los Capítulos 5 y 6 pueden extraerse una serie de conclusiones interesantes.

En primer lugar, queda patente que los Algoritmos Genéticos Multimodales son una herramienta poderosa cuando se desea optimizar funciones en un dominio multimodal, encontrando en la mayoría de los casos todos los óptimos del problema. Dado que la optimización de TMDs a escala real es un problema multimodal ha podido constatarse que los AGMM son adecuados para resolver este tipo de problemas si se configuran correctamente.

En segundo lugar, ha podido comprobarse que los parámetros utilizados para configurar los algoritmos tienen un gran impacto en los resultados. Existen dos modos posibles de configuración de los algoritmos, pueden configurarse para favorecer la explotación únicamente o para lograr un compromiso entre exploración y explotación. Cada uno de estos modos exige una configuración diferente.

En tercer lugar, se ha comprobado la influencia del operador cruce sobre la calidad de los resultados. Para todos los AGMM estudiados, el operador cruce que mejor se ha comportado ha sido el PNX.

En cuarto lugar, aunque todos los AGMM estudiados se han comportado de manera satisfactoria, si se desea favorecer la explotación y encontrar el óptimo global del problema el mejor algoritmo es el RCS. Sin embargo, si se desea un buen compromiso entre explotación y exploración, el SCGA funciona correctamente encontrando todos los óptimos del problema con suficiente calidad.

En quinto lugar, se ha comprobado y verificado en la literatura que los AGMM funcionan mejor que los AGS. La mayor ventaja de los AGMM es que además de encontrar gran cantidad de óptimos del problema, no se quedan atascados en óptimos locales, al contrario que los AGS.

En sexto lugar, la implementación de estos algoritmos sobre un lenguaje como Matlab, que permite trabajar con matrices de una forma muy simple y ágil, tiene grandes ventajas frente a lenguajes clásicos como C, tanto en el tiempo como en la simplicidad en el desarrollo.

7.2. Líneas futuras

Las líneas futuras o posibles campos de investigación dentro de los algoritmos genéticos y la optimización de estructuras son varias. Las dos inmediatas son las siguientes.

Estudiar el comportamiento de estos algoritmos y la influencia de sus parámetros frente a problemas de mayor escala o de diferente naturaleza (chimeneas, pasarelas, etc), ya que las optimizaciones llevadas a cabo en este trabajo se han efectuado sobre un modelo de dos plantas. Para cualquier otra estructura o una estructura similar a la estudiada de dimensiones muy superiores deberán hacerse las comprobaciones oportunas tanto en la influencia de los parámetros como en la influencia del operador cruce.

Estudiar el comportamiento de estos algoritmos y la influencia de sus parámetros frente a diferentes funciones de coste, como minimizar la respuesta temporal en lugar de la respuesta en frecuencia de la estructura.

Estudiar otros algoritmos genéticos más avanzados para abordar la optimización multiobjetivo, añadiendo variables como el coste de los equipos a instalar a la optimización.

REFERENCIAS

Aardal, K., Van Hoesel, S., Koster, A., Mannino, C., & Sassano, A. (2007). Models and solutions techniques for frequency assignment problems. *Annals of Operations Research*, 79-129.

Bennage, W., & Dhingra, A. (1995). Optimization of truss topology using tabu search.

Brindle, A. (1981). *Genetic algorithms for function optimization.* University of Alberta, Department of Computer Science.

Cheol-Gyun, L., Dong-Hyeok, C., & Hyun-Kyo, J. (s.f.). Niching genetic algorithm with restricted competition selection for multimodal function optimization. *IEEE Transactions on Magnetics*.

Chrysostomou, D., Gasteratos, A., Nalpantidis, L., & Sirakoulis, G. (2012). Multi-view 3D scene reconstruction using ant colony optimization technique. *Measurement Science and Technology*, Volume 23, Number 11.

Darwin, C. (1859). *On the origin of species by means of natural selection.* London: John Murray.

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.*

Farshidianfar, A., & Soheili, S. (2013). Ant colony optimization of tuned mass dampers for earthquake oscillations of high-rise structures including soil–structure interaction. *Soil Dynamics and Earthquake Engineering*, Volume 51, 14-22.

Feo, T., & Resende, M. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization March*, Volume 6, Issue 2, pp 109–133.

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

Gefenstette, J. (1995). Adaptive selection methods for genetic algorithms. *Proceedings of the first International Conference on Genetic Algorithms and their applications*, 101-111.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, Vol. 13, pp. 533-549.

Goldberg, D., & Deb, K. (1991). A comparative analysis on selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, 69-93.

Goldberg. , D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Boston, MA, USA: .Addison-Wesley Longman Publishing Co.

Greco, R., Marano, G., & Fiore, A. (2016). Performance-cost optimization of tuned mass damper under low-moderate seismic actions. *Structural Design of Tall and Special Buildings*, 25:1103–1122.

Hadi, M., & Arfiadi, Y. (1998). Optimum Design of Absorber for MDOF Structures. *Journal of Structural Engineering*, Vol. 124, Issue 11.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.*

Jiménez, J., & Sáez, A. (2015). Estimating robust optimum parameters of tuned mass dampers using multiobjective genetic algorithms. *3rd International Conference on Mechanical Models in Structural Engineering*, 245-252.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks* , Vol. IV: 1942–1948.

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 lasintonización óptima de TMDs múltiples enestructuras esbeltas.* Valladolid: Universidad de Valladolid.

Mahdavi, V., & Kaveh, A. (2013). Optimal design of structures with multiple natural frequency constraints using a hybridized BB-BC/Quasi-Newton algorithm. *Periodica Polytechnica Civil Engineering*, Vol.57,pp. 11.

Mahdavi, V., & Kaveh, A. (2014). Colliding bodies optimization: A novel meta-heuristic method. *Computers & Structures*, Vol. 139, pp. 18-27.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., & Teller, E. (s.f.). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, Volume 21, Issue 6, p.1087-1092.

Ormondroyd, J., & Den Hartog, J. (1928). The theory of the dynamic vibration absorber. *Transactions of the American Society of Mechanical Engineering*, 50:9–22.

Pérez, E. (2010). *Guía pararecién llegados a los algoritmos genéticos.* Valladolid: Unviersidad de Valladolid.

Pérez, E., & Herrera, F. (2007). Algoritmos genéticos multimodales: Un estudiosobre la parametrización del método clearingaplicado al problema "job shop". *V Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, 851-858.

Petrowski, A. (1996). A clearing procedure as a niching method for genetic algorithms. *Proceedings of IEEE International Conference on Evolutionary Computation*.

Pourzeynali, S., Salimi, A., & Eimani Kalesar, H. (2013). Robust multi-objective optimization design of TMD control device to reduce tall building responses against earthquake excitations using genetic algorithms. *Original Research Article Scientia Iranica*, Volume 20, Issue 2, 207-221.

Rajeev , S., & Krishnamoorthy, C. (1992). Discrete optimization of structures using genetic algorithms. *ASCE Journal of Structural Engineering*, Vol 118, pp. 1233-1250.

Sadek, F., Mohraz, B., Taylor, A., & Chung, R. (1997). A method of estimating the parameters of tuned mass dampers for seismic applications. *Earthq Eng Struct D*, 26(6):617–35.

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*.

Shim, P., & Manoochehri, S. (1997). Generating optimal configurations in structural design using simulated annealing.

Singh, M., Singh, S., & Moreschi, L. (2002). Tuned mass dampers for response control of torsional buildings. *Earthquake Engineering and Structural Dynamics.*, 31:4, 749–769.

Talatahari, A., & Kaveh, A. (2010). Novel heuristic optimization method: charged system search. *Acta Mechanica*, Vol. 213, pp. 267-289.

Talatahari, S., & Kaveh, A. (2009). Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. *Computers & Structructures*, Vol. 87, pp. 267-283.

Warburton, G. (1982). Optimal absorber parameters for various combinations of response and. *Earthq Eng and Structl Dyn*, 10: 381-401.

Wei-Min, Q., Wei-You, C., Qiao-Ling, J., Yuan-Chu, C., & Feng, P. (2005). Study and application of improved hierarchy genetic algorithm based on adaptive niches. *Machine Learning and Cybernetics*.

ANEXO I:
RESULTADOS EXPERIMENTALES CLEARING

Cruce BLX:

CL-BLX-1-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,217641869	8,217979836	
Mínimo		8,216313211	8,216388803	
Desviación		1,928601245	1,9286806	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

CL-BLX-1-10				
CONFIGURACIONES				
	11	12	21	22
Media	45,0496457	8,386249551	8,345563934	12,20607696
Mínimo	32,94934091	8,271419825	8,231722986	12,04899108
Desviación	13,32045188	2,696275656	2,683169403	3,92432917
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	3,8	3	4	0,410391341

CL-BLX-1-20				
CONFIGURACIONES				
	11	12	21	22
Media	20,70758734	8,339104884	8,348345524	12,22729924
Mínimo	20,27755182	8,243168292	8,235140011	12,01224877
Desviación	6,658918465	2,681058619	2,684138838	3,931282705
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

CL-BLX-1-30				
CONFIGURACIONES				
	11	12	21	22
Media	20,40907007	8,302305061	8,255922719	12,25106612
Mínimo	20,13139298	8,221203306	8,221441237	11,94727406
Desviación	6,561921472	2,669271516	2,654288694	3,939541001
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

CL-BLX-5-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,217345424	8,217298802	
Mínimo		8,216258011	8,21633711	
Desviación		2,016408639	1,835193507	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

CL-BLX-5-10				
CONFIGURACIONES				
	11	12	21	22
Media	38,18896829	8,23070349	8,231995015	21,81291149
Mínimo	25,23333316	8,217629259	8,217655749	13,84540011
Desviación	8,139661221	2,257190716	2,257545071	7,462363713
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	2,75	1	4	1,069923755

CL-BLX-5-20				
CONFIGURACIONES				
	11	12	21	22
Media	41,09481566	8,236456002	8,240486077	12,32010651
Mínimo	33,61667396	8,217782457	8,216839496	11,95565758
Desviación	12,68402293	2,589424693	2,649292261	3,973763446
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	3,85	2	4	0,489360485

CL-BLX-5-30				
CONFIGURACIONES				
	11	12	21	22
Media	37,74036161	8,24261531	8,287931363	12,22096452
Mínimo	29,45958515	8,218035753	8,217408132	11,9153164
Desviación	12,24274357	2,65001917	2,665029175	3,929912862
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

Cruce PNX:

CL-PNX-1-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,216435662	8,216371129	
Mínimo		8,21624327	8,216260291	
Desviación		2,016185379	1,8349863	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

CL-PNX-1-10				
CONFIGURACIONES				
	11	12	21	22
Media	47,46200001	8,415740436	8,366274332	12,11020522
Mínimo	38,36712022	8,230536024	8,221801914	11,91581945
Desviación	11,22700793	2,585682364	2,63254688	3,893545229
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	3,35	2	4	0,74515982

CL-PNX-1-20				
CONFIGURACIONES				
	11	12	21	22
Media	20,75516225	8,309847076	8,330224693	12,20310133
Mínimo	20,20957946	8,243955993	8,226552883	11,91128283
Desviación	6,674438193	2,671636838	2,678327334	3,923866083
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

CL-PNX-1-30				
CONFIGURACIONES				
	11	12	21	22
Media	20,46829641	8,256303314	8,25880088	12,71057767
Mínimo	20,13869297	8,219719722	8,220110307	12,03137157
Desviación	6,581696394	2,654420713	2,655216696	4,089592209
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

CL-PNX-5-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,216505396	8,216643723	
Mínimo		8,216283855	8,216284947	
Desviación		1,735368795	2,099340058	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

CL-PNX-5-10				
CONFIGURACIONES				
	11	12	21	22
Media	39,23912218	8,316523833	8,230767518	18,62700069
Mínimo	33,57944375	8,216330677	8,218051362	13,79005831
Desviación	7,80082704	2,491714878	2,181974999	6,063659088
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	2,8	1	4	1,00524938

CL-PNX-5-20				
CONFIGURACIONES				
	11	12	21	22
Media	40,26775407	8,22760581	8,23555094	12,46675396
Mínimo	35,91467701	8,217388685	8,216545701	11,93601848
Desviación	12,0671891	2,586629446	2,647705201	4,027078079
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	3,8	2	4	0,523148364

CL-PNX-5-20				
CONFIGURACIONES				
	11	12	21	22
Media	39,77153513	8,261027615	8,345123007	12,39072499
Mínimo	33,58187825	8,216478784	8,216582349	11,94382368
Desviación	12,84486965	2,6562103	2,685525623	3,985647297
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

Cruce SBX:

CL-SBX-1-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,220282367	8,219229066	
Mínimo		8,216560836	8,216563749	
Desviación		2,100269827	1,735944095	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

CL-SBX-1-10				
CONFIGURACIONES				
	11	12	21	22
Media	43,67121164	11,78182137	8,56269631	12,25270634
Mínimo	35,5126401	8,24091635	8,222688674	11,92430377
Desviación	13,84747532	5,02726737	2,758785802	3,940170241
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	3,95	3	4	0,223606798

CL-SBX-1-20				
CONFIGURACIONES				
	11	12	21	22
Media	21,03276119	8,49615444	8,414233824	12,26654246
Mínimo	20,16226023	8,221164732	8,221470553	11,92519671
Desviación	6,765011978	2,734522288	2,706434099	3,944641833
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

CL-SBX-1-30				
CONFIGURACIONES				
	11	12	21	22
Media	20,66281385	8,273030066	8,343812359	12,97843556
Mínimo	20,19460914	8,218918586	8,221598274	12,0301576
Desviación	6,644633663	2,660084184	2,683309042	4,176129325
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

CL-SBX-5-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,217994871	8,220193521	
Mínimo		8,216335705	8,217220712	
Desviación		1,928684101	1,929200133	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

CL-SBX-5-10				
CONFIGURACIONES				
	11	12	21	22
Media	40,17935991	8,403184721	8,434451552	19,19466472
Mínimo	37,13425115	8,216305449	8,21638385	12,99269605
Desviación	8,497436349	2,456291032	2,160499745	6,277437997
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	2,8	2	4	0,951453182

CL-SBX-5-20				
CONFIGURACIONES				
	11	12	21	22
Media	39,11222648	8,259586087	8,253139238	12,30324234
Mínimo	32,64076182	8,217894836	8,217736438	11,91993077
Desviación	12,33438134	2,655634506	2,594733379	3,957071111
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	3,9	2	4	0,447213595

CL-SBX-5-30				
CONFIGURACIONES				
	11	12	21	22
Media	36,46807662	8,227771145	8,308978431	13,10702268
Mínimo	24,75480002	8,216607573	8,216733832	11,98339162
Desviación	11,80403514	2,586686585	2,613018533	4,219501592
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	3,9	3	4	0,307793506

ANEXO II:
RESULTADOS EXPERIMENTALES RCS

Cruce BLX:

RCS-BLX-50-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,217053991	8,218243792	
Mínimo		8,21639046	8,216719469	
Desviación		1,928463268	1,928742515	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

RCS-BLX-100-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,216805412	8,216840936	
Mínimo		8,216354011	8,216259658	
Desviación		2,016276113	1,835091226	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

RCS-BLX-50-10				
CONFIGURACIONES				
	11	12	21	22
Media	48,39715935	8,870732097	8,736873739	12,80498387
Mínimo	29,31768762	8,439595136	8,383851602	12,27412698
Desviación	15,15703736	2,853067175	2,809666603	4,11853559
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-BLX-100-10				
CONFIGURACIONES				
	11	12	21	22
Media	21,07150084	8,739646726	8,65902253	12,6408867
Mínimo	20,46366435	8,516113762	8,391234627	12,34868913
Desviación	6,454520437	2,810197694	2,784574215	4,06459302
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-BLX-50-20				
CONFIGURACIONES				
	11	12	21	22
Media	21,50061436	8,834155358	8,764523723	12,98833515
Mínimo	20,61101126	8,452398925	8,431343672	12,28852237
Desviación	6,587348849	2,841330791	2,818364054	4,177623228
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-BLX-100-20				
CONFIGURACIONES				
	11	12	21	22
Media	21,75089817	8,997915007	9,171059781	13,48914102
Mínimo	20,74595104	8,589723977	8,635959133	12,12747487
Desviación	6,663092393	2,894401965	2,951035437	4,345904775
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-BLX-50-30				
CONFIGURACIONES				
	11	12	21	22
Media	22,5790618	9,249641311	9,181758499	14,01729404
Mínimo	21,19029153	8,344903802	8,392251565	12,79599551
Desviación	6,923433734	2,979924563	2,956865289	4,514247613
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-BLX-100-30				
CONFIGURACIONES				
	11	12	21	22
Media	22,5790618	9,249641311	9,181758499	14,01729404
Mínimo	21,19029153	8,344903802	8,392251565	12,79599551
Desviación	6,923433734	2,979924563	2,956865289	4,514247613
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

Cruce PNX:

RCS-PNX-50-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,216309648	8,216311958	
Mínimo		8,216252169	8,216253671	
Desviación		2,099254698	1,735327939	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

RCS-PNX-100-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,216285589	8,216325876	
Mínimo		8,2162479	8,216245192	
Desviación		1,628186839	2,178139122	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

RCS-PNX-50-10				
CONFIGURACIONES				
	11	12	21	22
Media	38,47036932	8,574594226	8,693273511	12,4858995
Mínimo	20,67481386	8,355981341	8,359827216	12,19239814
Desviación	12,46839857	2,757560856	2,795559441	4,014754196
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-PNX-100-10				
CONFIGURACIONES				
	11	12	21	22
Media	20,63353178	8,624257702	8,637784131	12,5783431
Mínimo	20,3053705	8,36033689	8,407267773	12,03215272
Desviación	6,320076994	2,773294467	2,777913901	4,045449294
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-PNX-50-20				
CONFIGURACIONES				
	11	12	21	22
Media	20,90268968	8,650897338	8,530799647	12,74848286
Mínimo	20,35795654	8,341052751	8,250901957	12,12942993
Desviación	6,402954598	2,783202107	2,743650405	4,101182986
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-PNX-100-20				
CONFIGURACIONES				
	11	12	21	22
Media	21,01432981	9,144287887	9,073496503	13,03494936
Mínimo	20,39303444	8,479743149	8,329223614	12,2497417
Desviación	6,438146265	2,944943565	2,920948538	4,193879509
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-PNX-50-30				
CONFIGURACIONES				
	11	12	21	22
Media	21,18687633	8,669009666	8,608109491	13,56366835
Mínimo	20,3869114	8,272716286	8,271366038	12,45078527
Desviación	6,492022854	2,789630223	2,768906668	4,368062438
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-PNX-100-30				
CONFIGURACIONES				
	11	12	21	22
Media	21,18687633	8,669009666	8,608109491	13,56366835
Mínimo	20,3869114	8,272716286	8,271366038	12,45078527
Desviación	6,492022854	2,789630223	2,768906668	4,368062438
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

Cruce SBX:

RCS-SBX-50-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,222144007	8,221249323	
Mínimo		8,216876969	8,216856785	
Desviación		1,929657922	1,929447949	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

RCS-SBX-100-0				
CONFIGURACIONES				
	11	12	21	22
Media		8,221864426	8,221675685	
Mínimo		8,217330573	8,21908301	
Desviación		2,017517619	1,836171023	
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	1	1	1	0

RCS-SBX-50-10				
CONFIGURACIONES				
	11	12	21	22
Media	22,59205114	8,972667263	9,156431795	12,9571752
Mínimo	21,58232089	8,520837701	8,375884236	12,26081028
Desviación	6,923773565	2,887783883	2,947911981	4,167914237
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-SBX-100-10				
CONFIGURACIONES				
	11	12	21	22
Media	21,70800648	8,929109434	8,796860465	12,62198251
Mínimo	20,64501414	8,456124431	8,296028195	12,09608744
Desviación	6,65170068	2,873250683	2,829390736	4,059274115
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-SBX-50-20				
CONFIGURACIONES				
	11	12	21	22
Media	21,91177675	9,077445709	8,918919842	13,27089636
Mínimo	21,04845749	8,514593535	8,311816529	12,44749175
Desviación	6,712557022	2,921184061	2,869975494	4,271244443
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-SBX-100-20				
CONFIGURACIONES				
	11	12	21	22
Media	21,95069098	8,878702882	8,963067637	13,33617974
Mínimo	20,95624905	8,51362918	8,359678614	12,39757557
Desviación	6,726064555	2,855761764	2,884036203	4,295322005
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-SBX-50-30				
CONFIGURACIONES				
	11	12	21	22
Media	23,64431883	9,818916402	10,20979417	14,33388062
Mínimo	21,22702528	8,229576503	8,347950773	12,56960914
Desviación	7,257358996	3,185069716	3,32893175	4,625070879
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

RCS-SBX-100-30				
CONFIGURACIONES				
	11	12	21	22
Media	23,64431883	9,818916402	10,20979417	14,33388062
Mínimo	21,22702528	8,229576503	8,347950773	12,56960914
Desviación	7,257358996	3,185069716	3,32893175	4,625070879
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

ANEXO III:
RESULTADOS EXPERIMENTALES SCGA

Cruce BLX:

SCGA-BLX-0				
CONFIGURACIONES				
	11	12	21	22
Media	54,67622032	29,56668501	30,85381699	37,58460997
Mínimo	34,70890304	20,4007421	17,95546615	29,78638281
Desviación	18,95431306	10,38145059	10,83445602	12,90347554
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

SCGA-BLX 10				
CONFIGURACIONES				
	11	12	21	22
Media	23,17191095	8,638775106	8,482237807	12,52329406
Mínimo	21,77026219	8,293829513	8,286379136	12,05842349
Desviación	7,910759209	2,949308415	2,893964531	4,272423591
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

SCGA-BLX 20				
CONFIGURACIONES				
	11	12	21	22
Media	34,69876867	11,43523791	10,53554426	15,14489203
Mínimo	23,96637461	8,216655394	8,216529322	13,17284179
Desviación	12,40921946	4,071091443	3,789555261	5,187878907
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

SCGA-BLX 30				
CONFIGURACIONES				
	11	12	21	22
Media	26,77357106	10,39043158	12,38557245	14,5969265
Mínimo	21,70854965	8,216363595	8,216353832	12,82922343
Desviación	9,299286648	3,738735519	4,429738222	5,002490969
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

Cruce PNX:

SCGA-PNX 0				
CONFIGURACIONES				
	11	12	21	22
Media	54,7018414	26,76673046	29,43422336	37,03392403
Mínimo	33,66884881	13,81671467	14,30040984	24,78239251
Desviación	19,11501854	9,411747183	10,2700165	12,8263666
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

SCGA-PNX 10				
CONFIGURACIONES				
	11	12	21	22
Media	21,96608474	8,349667091	8,461097504	12,29286775
Mínimo	20,44249488	8,263966353	8,289725711	12,00138443
Desviación	7,502000162	2,847992559	2,886994581	4,193246863
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

SCGA-PNX-20				
CONFIGURACIONES				
	11	12	21	22
Media	35,59489215	10,64259648	13,58797189	15,17106168
Mínimo	26,79168992	8,217940285	8,217210667	13,11975733
Desviación	12,45883082	3,854487431	4,925018299	5,190459486
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

SCGA-PNX-30				
CONFIGURACIONES				
	11	12	21	22
Media	27,26969973	10,50007562	14,02844636	14,92845718
Mínimo	22,57322723	8,216797378	8,216709222	13,15828627
Desviación	9,380803208	3,790638985	5,218210743	5,111945091
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

Cruce SBX:

SCGA-SBX 0				
CONFIGURACIONES				
	11	12	21	22
Media	52,82816194	29,29843272	26,88386857	36,99519237
Mínimo	27,24883218	16,74341332	17,13461998	27,27227475
Desviación	18,4967316	10,18394822	9,279363638	12,75169341
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

SCGA-SBX 10				
CONFIGURACIONES				
	11	12	21	22
Media	24,86623769	8,746914503	8,786015215	12,70181049
Mínimo	21,93119147	8,395234227	8,341614348	12,23100917
Desviación	8,506281745	2,985875727	3,001892922	4,333389143
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

SCGA-SBX-20				
CONFIGURACIONES				
	11	12	21	22
Media	36,81067883	11,63078353	14,08489555	16,59230495
Mínimo	25,34321603	8,216459632	8,216915145	13,84226344
Desviación	12,64569239	4,378214141	5,177568942	5,690738592
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

SCGA-SBX-30				
CONFIGURACIONES				
	11	12	21	22
Media	30,76171496	12,7718794	14,48489412	17,60406724
Mínimo	22,9056056	8,216550515	8,216649631	15,69311711
Desviación	10,55818513	4,878798698	5,377749147	6,009043667
SOLUCIONES ENCONTRADAS				
	Media	Mínimo	Máximo	Desviación
Nº Soluciones	4	4	4	0

ANEXO IV:
GRÁFICAS CLEARING

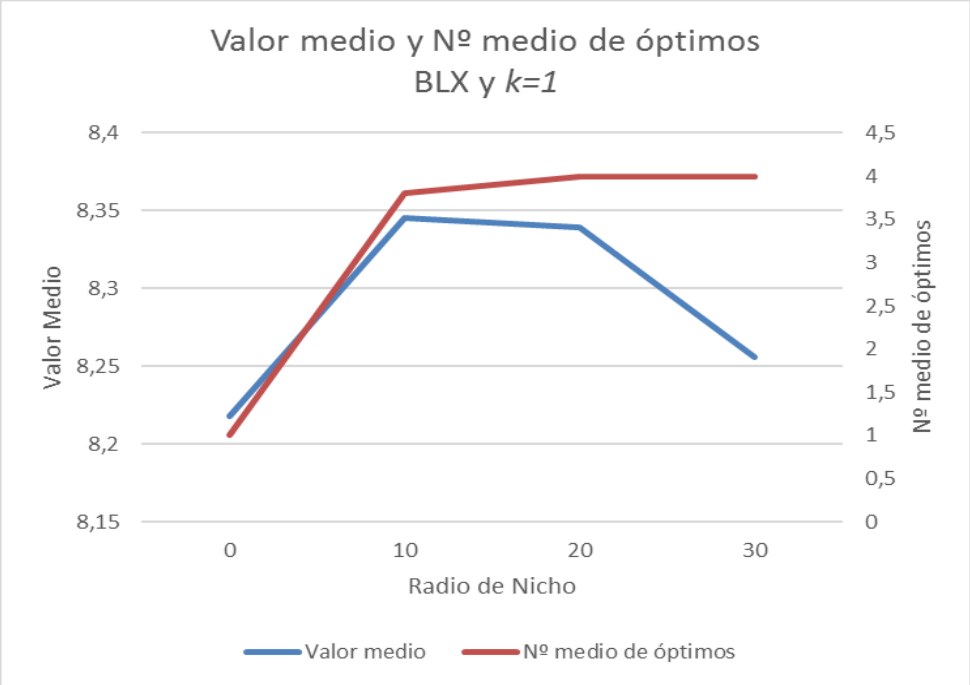

Figura AIV.1: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para BLX y k=1

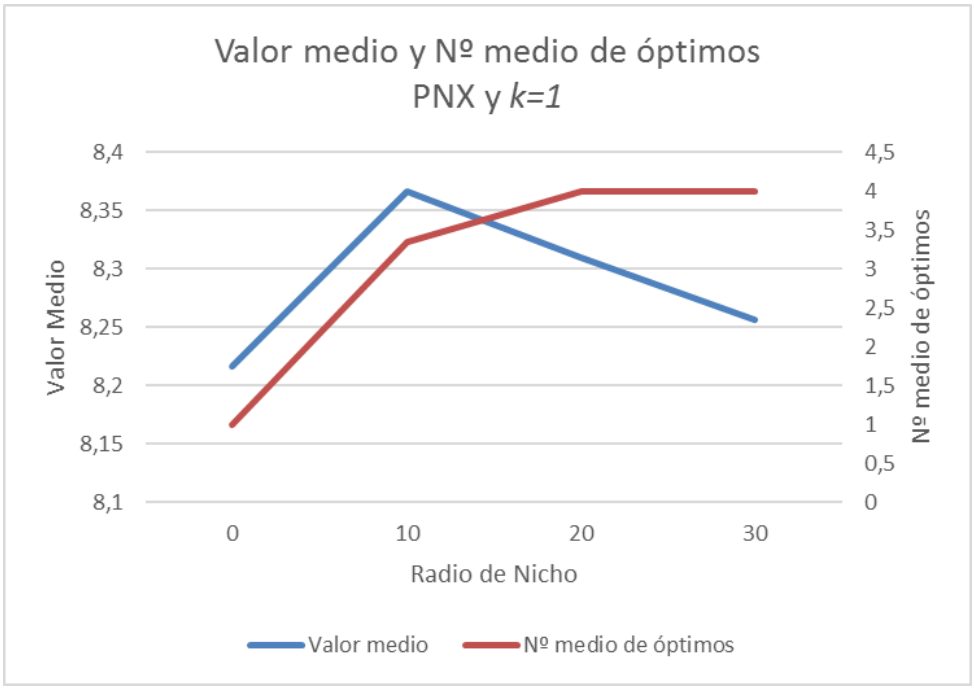

Figura AIV.2: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para PNX y k=1

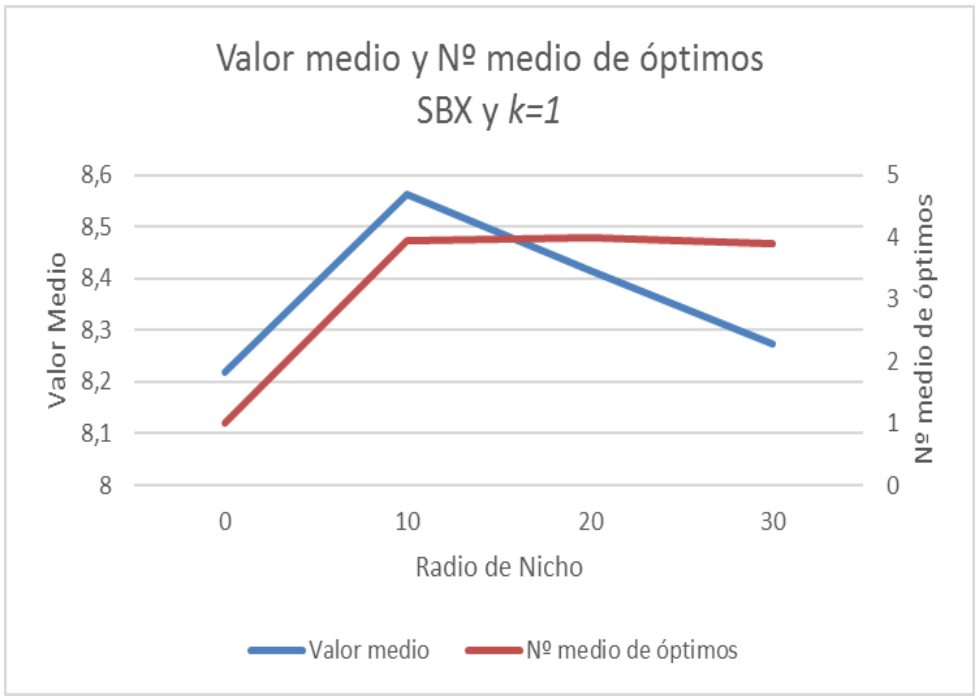

Figura AIV.3: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para SBX y k=1

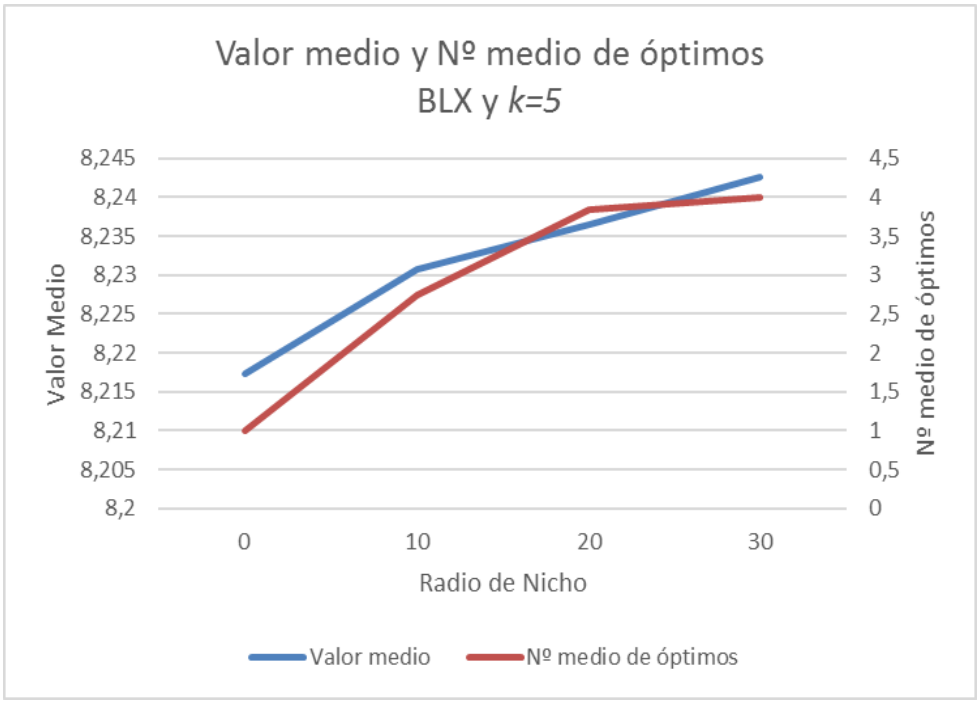

Figura AIV.4: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para BLX y k=5

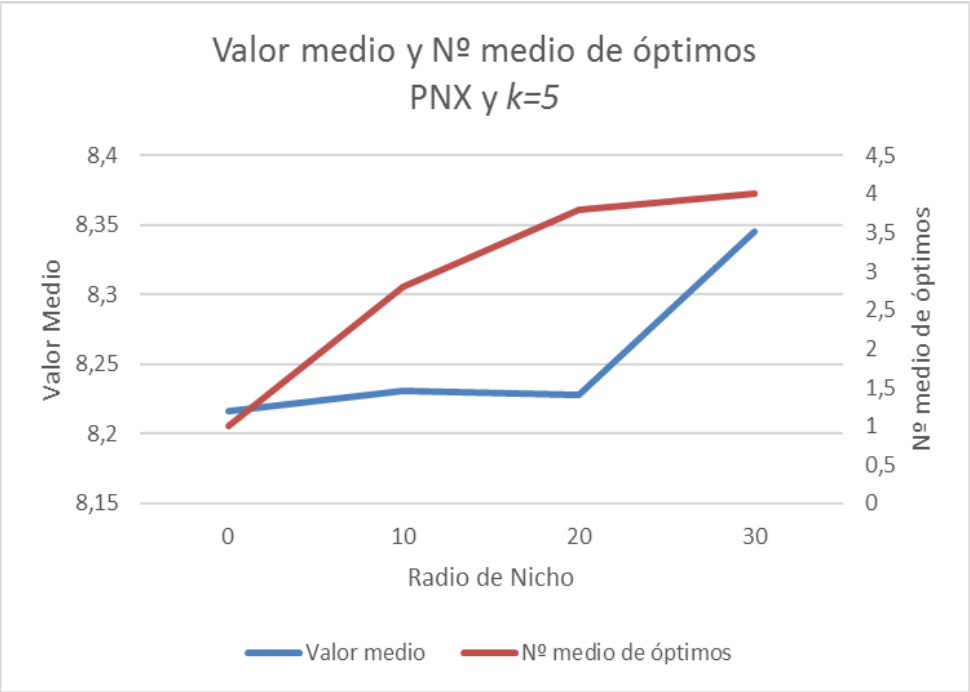

Figura AIV.5: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para PNX y k=5

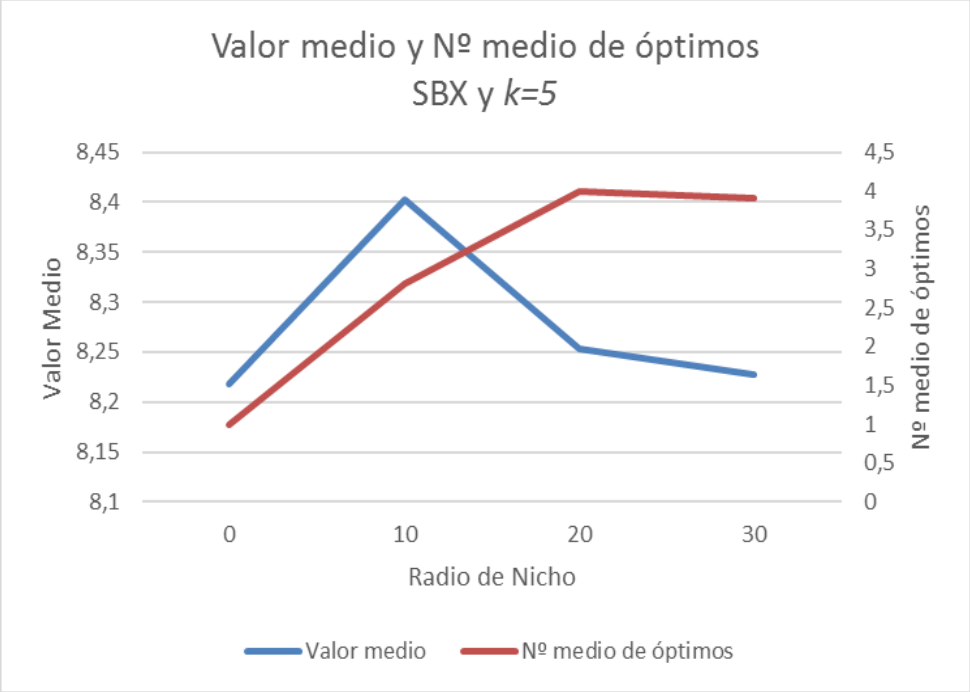

Figura AIV.6: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para SBX y k=5

ANEXO V:
GRÁFICAS RCS

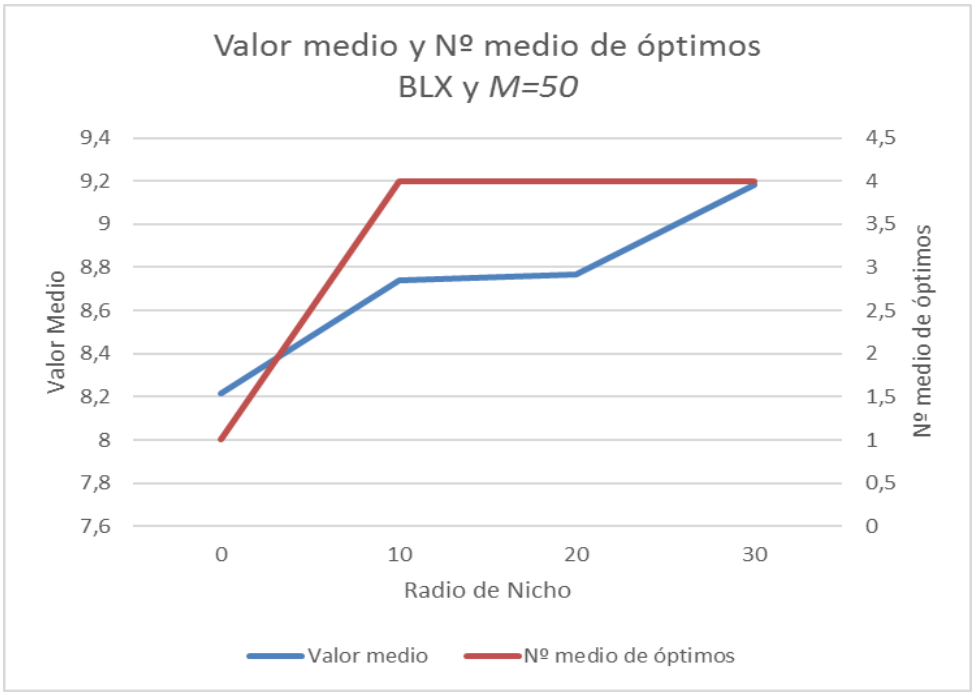

Figura AV.7: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para BLX y M=50

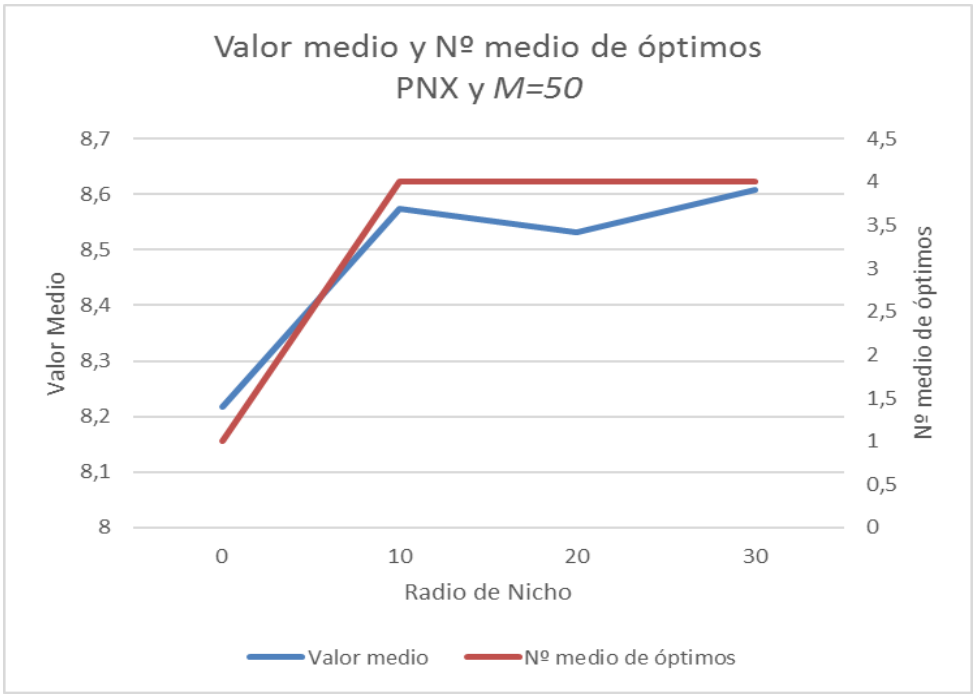

Figura AV.8: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para PNX y M=50

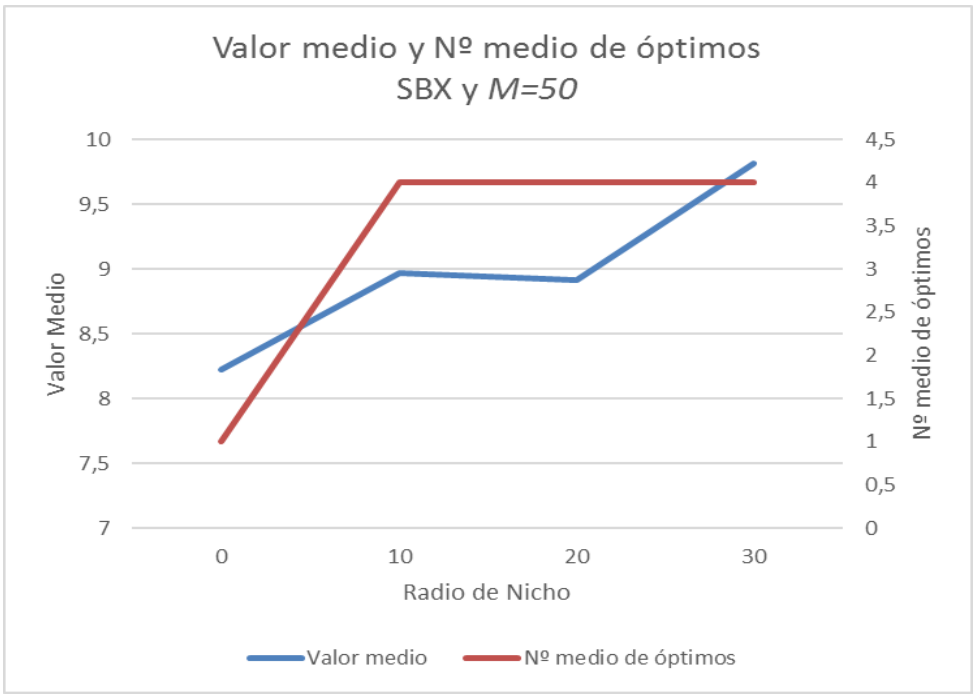

Figura AV.9: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para SBX y M=50

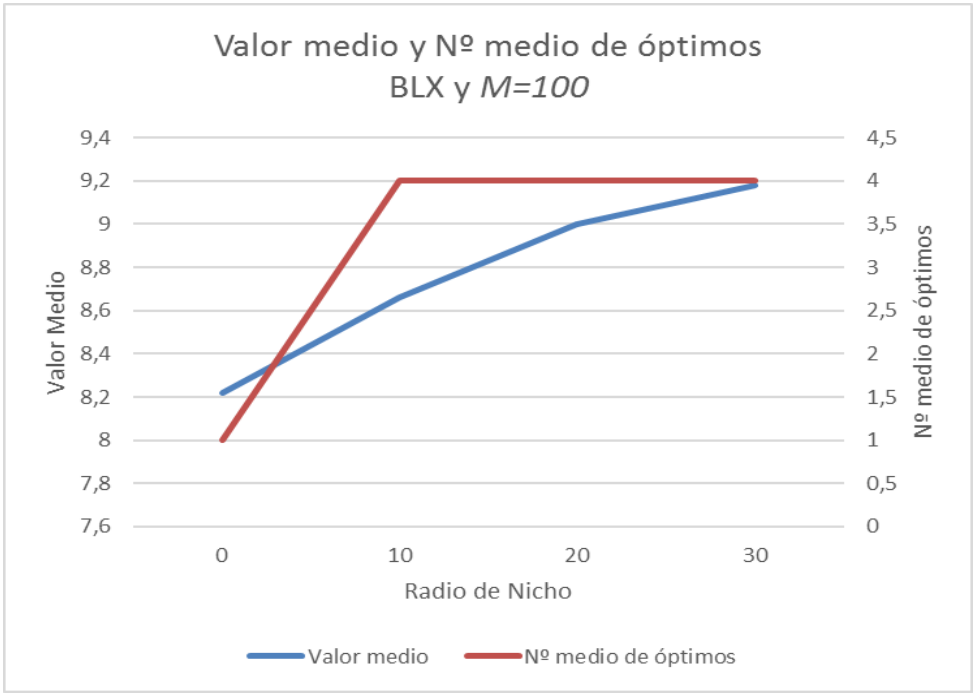

Figura AV.10: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para BLX y M=100

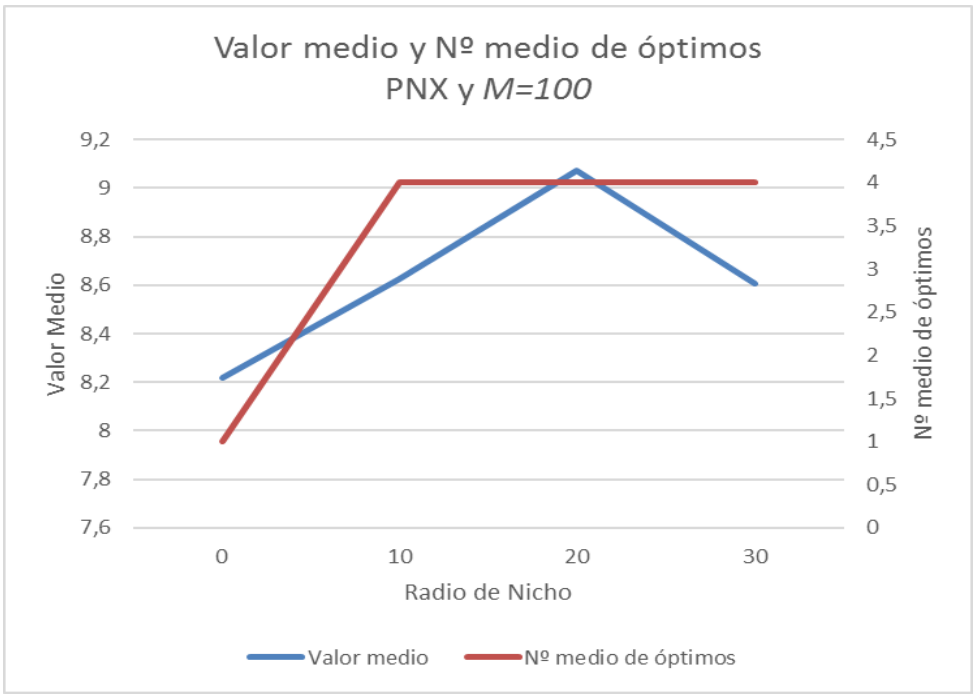

Figura AV.11: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para PNX y M=100

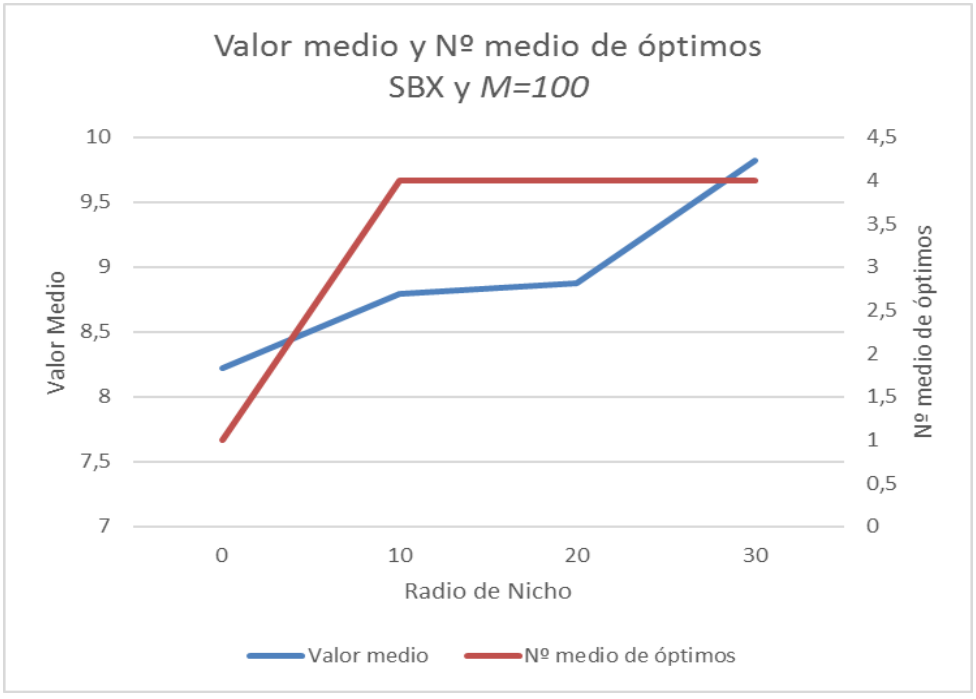

Figura AV.12: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para SBX y M=100

ANEXO VI:
GRÁFICAS SCGA

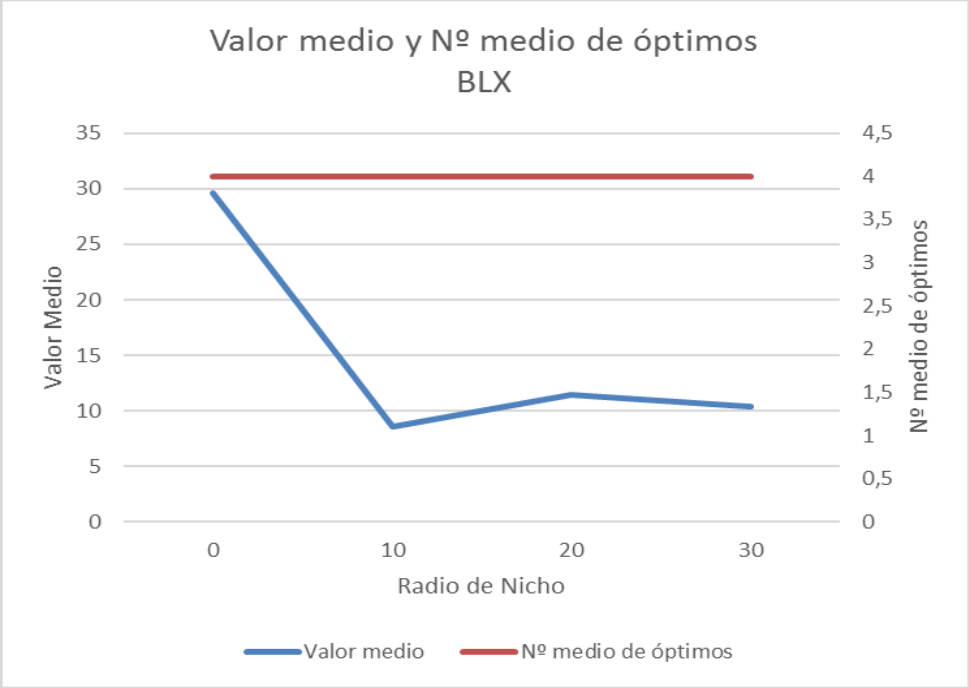

Figura AVI.13: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para BLX

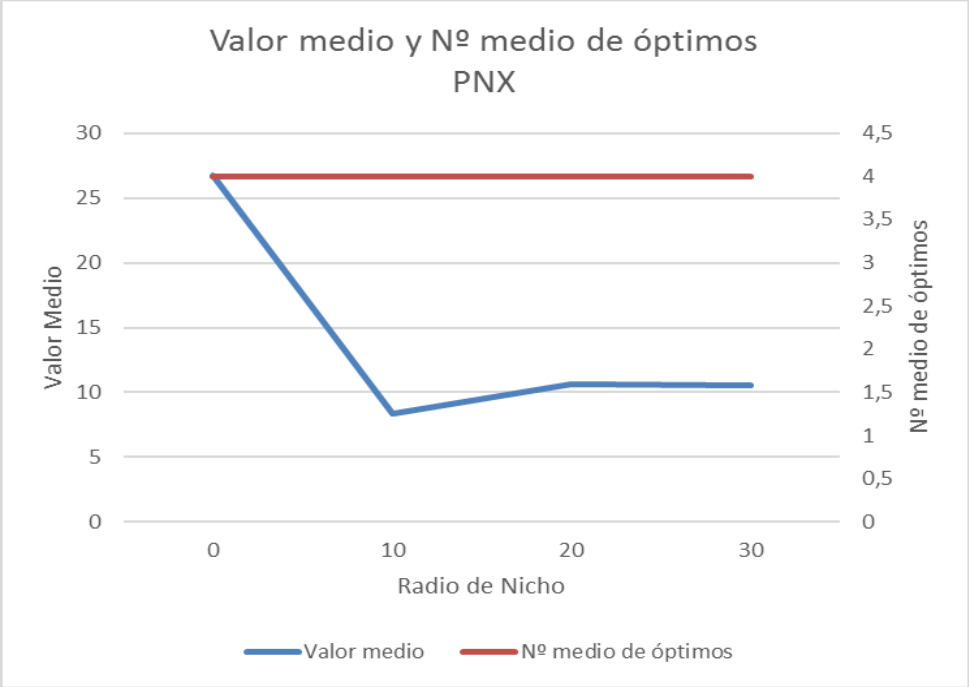

Figura AVI.14: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para PNX

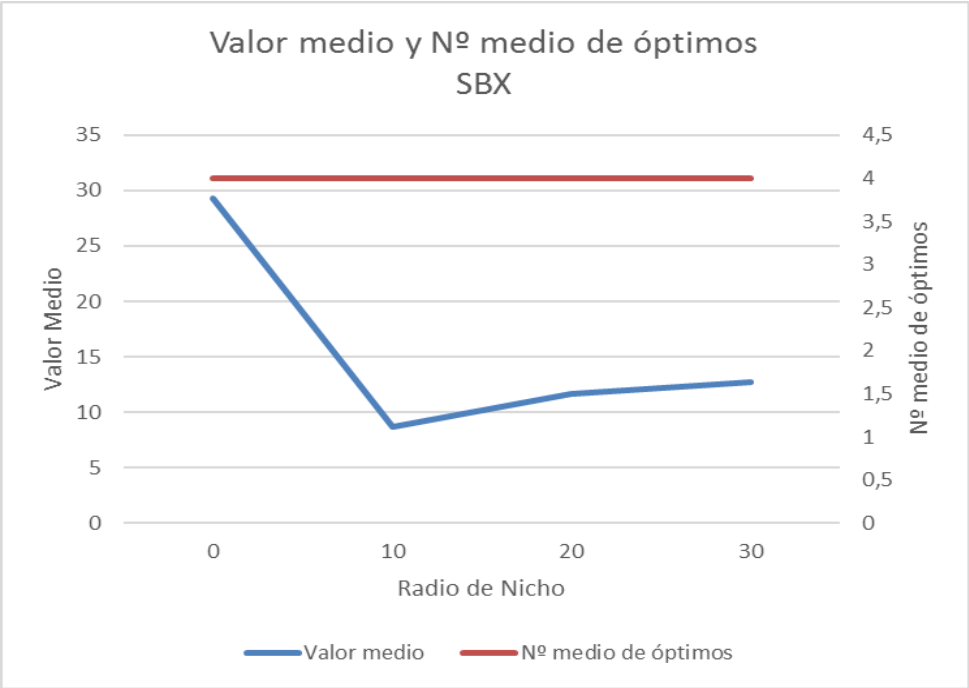

Figura AVI.15: Evolución del valor medio y el Nº medio de óptimos en función del radio de nicho para SBX