UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

Trabajo de Fin de Grado

Ingeniería de Tecnologías Especificas de Telecomunicación
Mención de Sistemas de Telecomunicación

Autor: DANIEL JIMÉNEZ GALINDO

Tutor: LUIS MIGUEL SAN JOSÉ REVUELTA

Título: "Diseño de filtros digitales IIR mediante algoritmos meméticos híbridos tipo SFLA."

Tutor: Luis M. San José Revuelta

Fecha defensa: 05 / 09/ 2018 12:30

Tribunal:

Presidente:	Luis M. San José Revuelta
Vocal:	M. Jesús González Morales
Secretario:	Juan C. García Escartín

Índice

0. Resumen

Este Trabajo Fin de Grado desarrolla la aplicación de un algoritmo memético (MA) a la resolución de un importante problema de optimización, concretamente el diseño de filtros digitales de respuesta al impulso de longitud infinita, o IIR. Para ello se propone un método de optimización basado en un algoritmo memético del tipo Shuffled Frog Leaping Algorithm (SFLA). SFLA es un algoritmo memético basado en una población formada por ranas que saltan en búsqueda de comida, de forma que aquellas en una peor situación serán capaces de imitar a las más aventajadas saltando en su misma dirección. El conjunto de ranas o población se divide en subgrupos de forma que estos evolucionen de forma independiente para luego ser barajadas, intercambiando la información sobre posiciones beneficiosas que han sido obtenidas al saltar. Aunque la base del MA propuesto es el algoritmo SFLA desarrollado por Eusuff.M, Lansey.K y Pasha.F. en [13], se han introducido una serie de modificaciones que lo hacen aplicable a problemas multiobjetivo además de mejorar la resistencia ante la tendencia a converger hacia mínimos locales, algo frecuente en algoritmos genéticos (AG) mal parametrizados. En este aspecto se ha introducido un mecanismo novedoso para asegurar la diversidad aplicando la entropía de Shannon a la distancia entre individuos en el plano de funciones objetivo. Los resultados obtenidos en el diseño de filtros IIR se han comparado con los correspondientes al método de diseño clásico basado en la Transformada Z Bilineal, además de los presentados en el trabajo de Upadhyay.P, Kar.R, Mandal.D y Ghoshal.S.P [24]. Los resultados obtenidos son prometedores desde el punto de vista del rizado en la banda de paso y eliminada además de la respuesta en fase del filtro en la banda de paso, presentando una mayor carga computacional que el método clásico de la BZT, compensado con una mejora en sus características, reduciendo el MSE obtenido con respecto al filtro ideal entre un 117% y un 342%. Adicionalmente, para el caso de diseño de filtros FIR también se encontró una mejora, en concreto el MSE quedó reducido entre un 147% y un 192%.

This proyect develops the application of memetic algorithms (MAs) to the resolution of optimization problems, concretely applied to Infinite Impulse Response digital filters (IIR) design. To tackle this objetive, a new optimization method based on a Shuffled Frog Leap Algortihm (SFLA) is proposed. SFLA is a memetic algorithm based on a population made up of frogs that leap in search of food, so those in the worst positions would be able to imitate the ones located in better positions by leaping in their same direction. The total population is divided into memeplexes, allowing them to evolve separately so they would be then shuffled, sharing information about the best positions discovered in the leaping process. Even tho the proposed MA is based on the SFLA developed by Luo.J, Yang.T, Liu.Q, Li.X, Chen.M, Gao.K in [23], many new modifications had been added in orden to make it suitable for multiple objective problems (MOPs) and resistent against the tendency of them getting trapped in local mínima, something that it´s frequently found in genetic algortihms (GA) that are not suitably parametrized. A new method has been introduced in order to ensure the diversity, applying the Shannon entropy to the distances between frogs in the objective function space. The obtained results refering to IIR design had been compared to the classic design method of the Bilinear

Z Transform, in addition to the ones presented in [24] by Upadhyay.P, Kar.R, Mandal.D, Ghoshal.S.P. The results are promising in regards of ripple in the passband and stopband and phase response in the passband, resulting in a higher computational load than the classic method of the BZT, compensated by an improvement in it's characteristics, reducing the MSE obtained between 117% and 342%. Additionally, applied to FIR design, the proposed MA reduced the MSE between 147% and 192%, proving to be suitable for that task.

1. Introducción

Actualmente los filtros digitales son un componente esencial en cualquier sistema electrónico y por tanto una pieza clave en el procesado de señal. La finalidad es extraer cierta información de la misma, bien seleccionando la parte que es de interés separándola de otras señales no deseadas o eliminando el ruido que la contamina el cual dificulta la extracción de ciertas características útiles.

Inicialmente se han tratado las bases de los filtros digitales IIR con el fin de establecer las características que los definen, así como los distintos métodos clásicos de diseño junto con las ventajas e inconvenientes de cada uno de ellos. La motivación de estudiar el diseño de filtros digitales en lugar de analógicos es que casi todo el procesado actual se realiza de forma digital, siendo el tratamiento de señal mediante software el futuro de las telecomunicaciones y la electrónica.

En referencia a los filtros digitales se distinguen dos tipos, los filtros FIR (Finite Impulse Response), cuya respuesta al impulso es finita; y los filtros IIR (Infinite Impulse Response), cuya respuesta al impulso es infinita. Los primeros también se les conoce como filtros MA (Moving Average) o de media móvil, y a los segundos también se les conoce como filtros ARMA (Autorecursive Moving Average). Estos últimos no solo dependen del estado de entradas anteriores como en el caso de los filtros FIR si no que también dependen de la salida en instantes anteriores, de ahí el calificativo de ªautorrecursivos".

Este trabajo se centrará en el diseño de filtros IIR ya que poseen ciertas ventajas con respecto a los filtros FIR, principalmente el menor uso de recursos ya que para un mismo orden los primeros ofrecen mejores características de filtrado. No obstante, existen ciertos inconvenientes en el uso de filtros IIR ya que al poseer polos no siempre son estable, característica indispensable de cualquier sistema de procesado.

1.1 Motivación

Como se explica en el capítulo 3, existen diversos métodos clásicos para el diseño de filtros IIR como pueden ser la BZT, método de la invarianza al impulso o el método de aproximación por derivadas. No obstante, estos métodos poseen ciertas desventajas principalmente relacionadas con el tipo de filtros que se pueden diseñar o el error que se produce al transformar la respuesta analógica del filtro al dominio digital, esto hace que no sean útiles en muchos sistemas cuya complejidad a aumentado drásticamente en los últimos años.

A pesar de que existen diversos tipos de filtros a diseñar, como Butterworth, Chebychev tipo I y II y elípticos, donde los primeros son los más simples y los últimos los más complejos, todos ellos carecen de una característica muy deseable que es la adaptabilidad. Proporcionalmente a

su complejidad, poseen ciertos parámetros que permiten establecer las restricciones del filtro obtenido, no obstante, en muchos casos no es posible adaptarlos por completo al problema que se desea abordar, por tanto, se proponen otros métodos de diseño más configurables basados en la naturaleza: los métodos bioinspirados.

Estos métodos bioinspirados basan su funcionamiento en ciertos ámbitos de la naturaleza como pueden ser el comportamiento de agrupaciones de animales o en leyes físicas relacionadas con la atracción gravitatoria o la formación de cristales en un sólido. Estos algoritmos bioinspirados tuvieron sus inicios con los algoritmos genéticos, los cuales se basan en la ley de la evolución formulada por Darwin, donde son los individuos que mejores características poseen con respecto al entorno los que mayor probabilidad tiene de preservar su información genética. En adición, estos individuos son siempre una combinación genética de sus progenitores, sufriendo en ciertas ocasiones modificaciones o mutaciones, que pueden ser ventajosas o no para su supervivencia.

Como sucesores de los algoritmos genéticos se encuentran los algoritmos meméticos que se centran en la capacidad de ciertos individuos de identificar y seguir a aquellos que se comportan de forma más ventajosa, lo cual es una diferencia esencial con respecto a los AG, donde los individuos no son capaces de comunicarse entre sí. Estos algoritmos meméticos poseen características aprovechables para el diseño de algoritmos de optimización, ya que los individuos exploran el espacio de soluciones y si alguno de ellos encuentra un buen “lugar” o solución, los demás serán capaces de seguirlo para moverse hacia un conjunto de soluciones mejores.

La motivación que impulsa el uso de algoritmos meméticos para el diseño de filtros IIR es por tanto la capacidad que tienen de mejora evitando óptimos locales en los que caen los métodos clásicos. Adicionalmente, estos poseen un mayor número de parámetros que permiten ajustar las soluciones para adaptarlas al problema, bien centrándose en la solución final o en la carga computacional del algoritmo.

1.2 Objetivos

Los objetivos de esta memoria son:

- Mostrar las características básicas que poseen los filtros IIR así como las especificaciones que los definen.
- Tratar distintos métodos clásicos para el diseño de filtros IIR así como las ventajas e inconvenientes de cada uno de ellos.
- Hacer una exposición del estado del arte actual con respecto a los algoritmos meméticos y genéticos.
- Enumerar y explicar distintos algoritmos meméticos reconocidos en la literatura científica, así como su aplicación al diseño de filtros IIR.

- Proponer un algoritmo memético capaz de abordar el problema del diseño de filtros IIR empleando dos funciones objetivo y tratando de mejorar los resultados mediante el método clásico de la BZT.
- Realizar una comparación cuantitativa de los resultados obtenidos mediante el algoritmo memético propuesto y los presentados en uno de los artículos.

1.3 Organización de la memoria

Esta memoria se organiza de la siguiente manera: el capítulo 2 es una introducción a los filtros IIR donde se presentan las bases necesarias para poder determinar sus requisitos y su función de transferencia. En el capítulo 3 se trata la correspondencia entre el plano analógico y digital así como los distintos métodos clásicos de diseño de filtros IIR partiendo de un filtro analógico con las restricciones deseadas. En el capítulo 4 se explica el funcionamiento de distintos algoritmos meméticos junto con las características que diferencian a cada uno de los demás, además de los resultados que se han obtenido en cada caso con respecto al diseño de filtros IIR. En el capítulo 5 se describe el algoritmo propuesto, presentando los distintos mecanismos para la optimización de las soluciones, así como el fundamento de cada uno de ellos, también se proponen algunas características de diseño que hacen su implementación posible. Seguidamente en el capítulo 6 se presentan los resultados obtenidos empleando el MA propuesto, contemplando los distintos valores de algunos de sus parámetros para obtener una respuesta óptima. Por otro lado, el MA propuesto se ha empleado para generar filtros de varios tipos comparando el resultado con el obtenido empleando filtros elípticos y con los resultados obtenidos en el artículo [24]. Finalmente, en el capítulo 7 se presentan las conclusiones a las que se ha llegado con el capítulo 6, incluyendo además futuras líneas de trabajo.

2. Introducción a los filtros digitales IIR

2.1 Filtros IIR

2.1.1 Descripción

El nombre IIR viene de las siglas en inglés Infinite Impulse Response, de forma que se trata de filtros digitales que por sus características producen una salida de longitud infinita ante un impulso a su entrada. La salida de este tipo de filtros tiene una longitud infinita debido a que esta, en un instante concreto no solo depende de los valores de la entrada en instantes temporales anteriores sino también de los de valores anteriores de dicha salida [17,18].

2.1.2 Función de transferencia y respuesta al impulso

Los filtros IIR más importantes vienen descritos por la siguiente ecuación en diferencias:

$$y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_M x(n-M) - a_1 y(n-1) - a_2 y(n-2) - \cdots - a_N y(n-N) \quad (2.1)$$

donde x(n) representa la señal de entrada, *y(n)* es la salida del filtro y los $\{a_1, a_2, \ldots, a_N\}$ y $\{b_0, b_1, \ldots, b_M\}$ son los coeficientes del filtro. Se asume que a_N es no nulo, de forma que la respuesta al impulso es la salida del sistema cuando la entrada es un impulso en $n=0$, con el sistema partiendo del reposo (salida nula para instantes anteriores a $n=0$).

Si se le aplica la transformada Z a la ecuación (2.1) se obtiene la función de transferencia del filtro, la cual es de la forma:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} \quad (2.2)$$

donde N es el orden del filtro. Se asume que M es siempre menor o igual que N, ya que si esto no se cumple la ecuación podría descomponerse en una suma de un polinomio en la variable z más un cociente, lo que representa un filtro FIR en paralelo con un filtro IIR [17].

2.3.3 Polos y ceros

Si reescribimos la respuesta al impulso descrita por la ecuación (2.2) de la forma:

$$H(z) = z^{N-M} \frac{b_0 z^M + b_1 z^{M-1} + \cdots + b_{M-1} z + b_M}{z^N + a_1 z^{N-1} + \cdots + a_{N-1} z^{-N}} \quad (2.3)$$

Asumiendo b_0 y b_M no nulos, podemos extraer que la respuesta al impulso posee N polos y M ceros, cuyos valores vienen dados por las raices del denominador y numerador respectivamente. Por otro lado, habrá *N-M* ceros localizados en el origen del plano complejo.

Si se obtienen las raices del numerador y denominador, y se reescribe ambos como producto de polinomios de grado 1, se llega a la siguiente fórmula:

$$H(z) = b_0 z^{N-M} \frac{(z - q_1)(z - q_2) \cdots (z - q_M)}{(z - p_1)(z - p_2) \cdots (z - p_N)} \quad (2.4)$$

donde las constantes $\{q_1, q_2, \ldots, q_M\}$ representan los ceros y las constantes $\{p_1, p_2, \ldots, p_N\}$ representan los polos de la respuesta al impulso. Asumiendo que los coeficientes representados en la ecuación (2.3) generan polos y ceros complejos, estos se encontrarán como pares de polos y ceros complejos conjugados, por tanto, se podrían reescribir dichos polinomios como un producto de polinomios de segundo grado con coeficientes reales.

Finalmente, de la ecuación (2.4) se deduce que estableciendo un valor para la constante b0, se podría representar la respuesta del sistema con tan solo observar las posiciones de los ceros y los polos, los cuales representan mínimos y máximos locales respectivamente [17].

2.3.4 Estabilidad

La estabilidad es una de las características más deseables de un filtro de respuesta infinita ya que al poseer una realimentación, una entrada acotada podría no producir una salida limitada, haciendo que este fuera instable. Para verificar la estabilidad de un filtro IIR todos los polos del mismo deben estar situados en el interior de la circunferencia de radio unidad en el dominio z, no obstante, en casos particulares es posibles que algunos de estos polos estén situados justo en la circunferencia unidad, entonces se dice que el filtro es marginalmente estable [17] (mantendría un estado oscilatorio).

2.3.5 Implementación.

A la hora de implementar un filtro IIR existen varios procedimientos para determinar el esquema del mismo, las formas más inmediatas son las directas, que se obtienen de la ecuación (3).

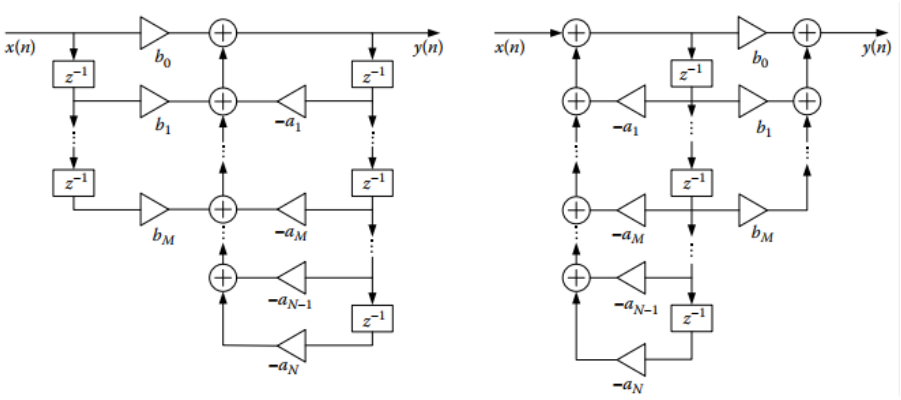

Figura 2.1. *Implementaciones directa I a la izquierda y directa II a la derecha.*

Mediante la transposición de los esquemas anteriores se pueden obtener las formas directas traspuestas I y II, como se muestran en la siguiente figura:

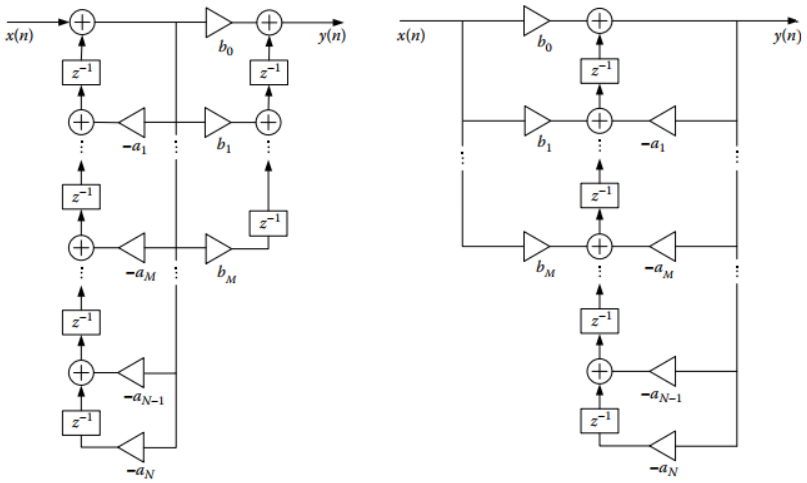

Figura 2.2*. Implementaciones indirecta I a la izquierda e indirecta II a la derecha.*

Por otro lado, existen las formas en cascada y en paralelo. Para la forma en paralelo se deberá descomponer la ecuación de respuesta al impulso como se ilustra en la ecuación (2.5), de forma que *K* módulos de orden dos irán conectados de forma paralela además de la constante.

$$H(z) = \frac{b_N}{a_N} + \sum_{i=1}^{K} \frac{b_{i0} + b_{i1} z^{-1}}{1 + a_{i1} z^{-1} + a_{i2} z^{-2}} \tag{2.5}$$

donde K tomará el valor N/2 si este es par o por el contrario valdrá *(N+1)/2* si es impar, adicionalmente, si se da el segundo caso, una de las fracciones estará formada por un polinomio de grado cero en el numerador y por un polinomio de grado uno en el denominador.

Para la forma en cascada la descomposición también se realiza mediante cocientes de segundo grado, no obstante, estos se encontrarán dentro de un productorio además de que no existirá ningún término independiente de grado cero (todos los módulos tendrán una entrada común y

una salida común hacia un sumador). La ecuación tendría la forma que se indica en la ecuación (2.6).

$$H(z) = \prod_{i=1}^{K} \frac{b_{i0} + b_{i1}z^{-1} + b_{i2}z^{-2}}{1 + a_{i1}z^{-1} + a_{i2}z^{-2}} \tag{2.6}$$

donde K tomará los mismos valores que en el caso de la configuración en paralelo, sin embargo, en este caso el esquema estará formado por K módulos de segundo grado colocados en cascada [17] (uno a continuación del otro).

2.3.6 Fase mínima

Se dice que un filtro IIR es de fase mínima cuando introduce el menor retardo de grupo de entre todos los filtros con esa misma respuesta, esto se consigue configurando sus polos y ceros para que todos ellos se encuentren dentro de la circunferencia unidad. Por otro lado si se tiene un filtro que no es de fase mínima este puede ser modificado transformando los polos y ceros que se encuentran fuera de la circunferencia unidad realizando el recíproco del complejo conjugado de cada uno de ellos, de forma que se consigue fase mínima y solo cambiará su amplitud (la cual se puede ajustar mediante un amplificador) [17].

2.4 Diseño de filtros IIR

2.4.1 Objetivos

Cuando se trata de diseñar un filtro casi siempre se hace de cara a cumplir una serie de especificaciones relacionadas con la amplitud de su respuesta en frecuencia, las cuales pueden ser más o menos estrictas dependiendo del caso concreto. No obstante hay ciertos ámbitos en lo que la respuesta en fase del filtro también juega un papel muy importante, como puede ser en el caso de un sistema de comunicaciones donde la fase de la señal puede llevar parte de la información, de forma que si esta se distorsiona puede resultar inservible. [17,18].

Por otro lado, el diseño de filtros IIR se puede abordar desde tres puntos de vista distintos:

Puede diseñarse a partir de un filtro analógico que cumpla las especificaciones y después realizar una serie de transformaciones para hallar un equivalente digital.
El filtro digital puede obtenerse a partir de un filtro digital paso bajo predefinido aplicando una serie de transformaciones en frecuencia.

Puede emplearse software para calcular los coeficientes del filtro de forma que este se aproxime todo lo posible al filtro deseado con una cierta tolerancia [17].

2.4.2 Especificaciones

A la hora de determinar las especificaciones de un filtro se asume en la mayoría de los casos que la amplitud del mismo tiene como máximo valor uno (amplitud normalizada) en la banda de paso. Aunque existen cuatro tipos de filtros (paso bajo, paso alto, paso banda y elimina banda) la notación para las especificaciones de cada uno de ellos es idéntica, basándose siempre en las del filtro paso bajo, como se muestra en la figura 2.3.

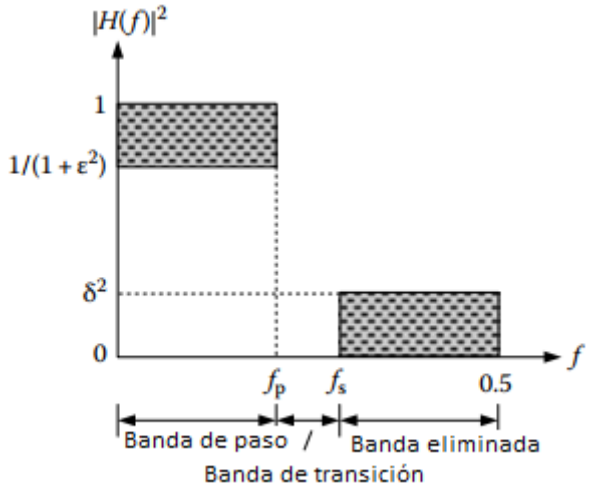

Figura 2.3. *Representación de las especificaciones de un filtro IIR paso bajo [17].*

donde 1/(1+ε2) representa la amplitud mínima del filtro en la banda de paso, δ2 representa la amplitud máxima en la banda eliminada, fp la frecuencia límite de la banda de paso (passband), fs la frecuencia a partir de la cual se considera la banda eliminada (stopband) y el intervalo de frecuencias entre fp y fs representa la banda de transición [17] (transition band).

2.4.3 La transformada z

Esta transformada es esencial para el estudio de filtros digitales, ya que nos permite representar sus polos y ceros así como su módulo y posición. La transformada z bilateral se obtiene mediante la ecuación (2.7).

$$X(z) = \mathcal{Z}\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n]z^{-n} \tag{2.7}$$

donde x[n] representa un señal discreta y z es la variable compleja. Junto con la transformada z a menudo es necesario indicar la región de convergencia o ROC ya que X(z) es una serie infinita de potencias que solo existe para los valores de z que la hagan converger. A pesar de que transformamos la señal x[n] a la variable z, es fácil identificar las características de la señal en tiempo ya que el coeficiente nos indica el valor de la señal y la potencia a la que está elevada la z nos indica su instante temporal, o de forma más genérica, el índice n al que pertenece la muestra [17].

2.4.4 Diseño mediante prototipo analógico

Este tipo de diseño se basa en la conversión de un filtro analógico a uno digital que cumpla las especificaciones necesarias, además es uno de los métodos más estudiados y empleados hasta el momento, en parte por su baja carga computacional en comparación con ciertos métodos software.

2.4.4.1 Conversión del plano *s* al plano *z*

Debido a que este tipo de métodos se basan en la conversión de un filtro analógico a uno digital es importante ilustrar la correspondencia que existe entre el plano analógico s (basado en la transformada de Laplace, apartado 2.2.1) y el plano digital z (dominio de la transformada z, apartado 2.4.3). La siguiente figura ilustra la correspondencia entre plano s y plano z:

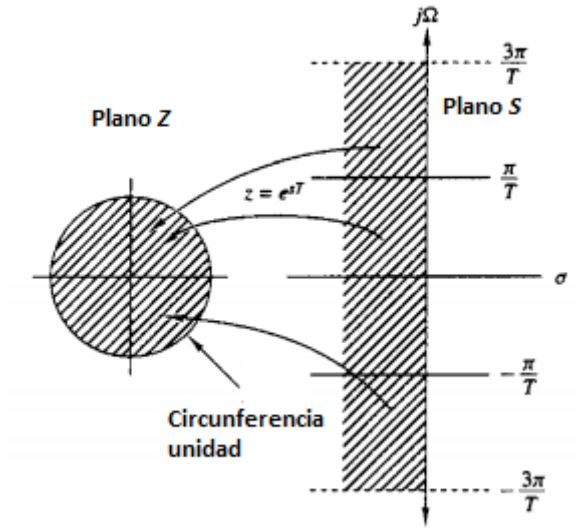

***Figura 2.4**. Representación de la correspondencia entre plano digital Z y plano analógico S [18].*

Como puede apreciarse, el eje *jΩ* pasa a ser la circunferencia unidad y todo el semiplano izquierdo en el dominio s pasa a ser el interior de la circunferencia unidad. Desde un punto de vista analítico, establecemos la relación entre ambos planos como: $z = sT$, y si tenemos en cuenta que las siguiente igualdades:

$$s = \sigma + j\Omega \quad (2.8)$$

$$z = re^{j\omega} \quad (2.9)$$

Se llega a la conclusión de que:

$$r = e^{\sigma T} \quad (2.10)$$

$$\omega = \Omega T \quad (2.11)$$

Las implicaciones de estas igualdades son:

- Siempre y cuando nos encontremos con $\sigma<0$, estaremos dentro de la circunferencia unidad en el plano z, lo cual es importante a la hora de colocar los polos de un filtro IIR.
- Debido a que todo el eje jΩ del plano s se mapea en la circunferencia unidad del plano z tenemos una correspondencia de mucho a uno, lo que refleja los efectos del aliasing cuando se muestrea. Un caso concreto de esto, es que tanto el intervalo $-\pi / T \leq \Omega \leq \pi / T$ como el intervalo $\pi / T \leq \Omega \leq 3\pi / T$ se mapean ambos en la semicircunferencia $-\pi \leq \omega \leq \pi$, por tanto la correspondencia no es de uno a uno, sino de muchos a uno.

2.4.4.2 Diseño mediante la invarianza al impulso (IIM)

En este método se basa en la aproximación del filtro analógico $h_a(t)$ mediante muestreo cada T segundos, por tanto, el filtro digital resultante $h(n)$ será equivalente a $h_a(nT)$. La ecuación que representa este muestreo en el dominio frecuencial viene dada por la ecuación (2.12).

$$H(f) = \frac{1}{T}\sum_{k=-\infty}^{\infty} H_a(\lambda)\Big|_{\lambda=2\pi(f+k)/T} = \frac{1}{T}\sum_{k=-\infty}^{\infty} H_a\left(2\pi\frac{f+k}{T}\right) \quad (2.12)$$

donde $H_a(\lambda)$ representa la transformada de Fourier del filtro analógico *y* $H(f)$ representa la transformada discreta del filtro digital correspondiente. De esta ecuación se extrae que el filtro digital obtenido es una versión periódica del filtro analógico que se repite cada 2π, por tanto, el aliasing es un factor a tener en cuenta ya que si T no es suficientemente pequeña estas versiones periódicas podrían solaparse.

Para evitar el aliasing deberá cumplirse que $H_a(\lambda)$ tenga valor cero para $|\lambda/2\pi| > 1/(2T)$, lo cual es bastante poco realista por el hecho de ningún filtro atenúa completamente en la banda eliminada, es decir, no es posible que el ancho de banda del filtro analógico esté completamente

acotado. Otra de las consecuencias de esta limitación es que no posible diseñar filtros paso alto o de banda eliminada mediante este método, ya que ambos tienen el máximo de su módulo para las frecuencias superiores a la corte, por tanto, siempre habrá aliasing.

Si representamos la función de transferencia del filtro analógico mediante la ecuación (2.13), entonces su respuesta al impulso vendrá dada por la ecuación (2.14). Finalmente, el filtro digital correspondiente quedará descrito mediante la ecuación (2.15).

$$H_a(s) = \sum_{i=1}^{N} \frac{b_i}{(s - p_i)} \tag{2.13}$$

$$h_a(t) = \sum_{i=1}^{N} b_i e^{p_i t} \tag{2.14}$$

$$\begin{aligned} H(z) &= \sum_{n=0}^{\infty} h(n) z^{-n} = \sum_{n=0}^{\infty} h(nT) z^{-n} \\ &= \sum_{i=1}^{N} b_i \sum_{n=0}^{\infty} (e^{p_i T} z^{-1})^n = \sum_{i=1}^{N} \frac{b_i}{1 - e^{p_i T} z^{-1}} \end{aligned} \tag{2.15}$$

De este modo la función de transferencia del filtro analógico pasa a ser la de un filtro digital, mediante el mapeo de las frecuencias analógicas hacia las digitales, pasando del plano s al plano z.

2.4.4.3 Diseño mediante aproximación de derivadas

Si expresamos la función de transferencia de un filtro analógico en forma diferencial obtenemos la ecuación (2.16).

$$\sum_{k=0}^{N} a_k \frac{d^k y(t)}{dt^k} = \sum_{l=0}^{M} b_l \frac{d^l x(t)}{dt^l} \tag{2.16}$$

donde *x(t)* representa la señal de entrada e *y(t)* representa la salida, podemos obtener su ecuación en diferencias equivalente. Como punto de partida establecemos el valor de la derivada de primer orden en función de *y(t)*, por tanto tendremos la ecuación (2.17).

$$\frac{dy}{dt} \approx \frac{y(n) - y(n-1)}{T} \tag{2.17}$$

donde T representa el intervalo de muestreo, asumiendo que *y(n) = y(Tn)*. Como se indicó en el apartado 2.2.1, la derivada de primer orden de y(t) equivale a una función del sistema de la

forma *H(s) = s*, la cual transformada al dominio Z digital tendrá como expresión *H(z) = (1 - z-1)/T*. Por tanto, se deduce que la expresión de s vendrá dada por la ecuación (2.19) y la de z por la ecuación (2.18).

$$z = \frac{1}{1 - sT} \tag{2.18}$$

$$s = \frac{1 - z^{-1}}{T} \tag{2.19}$$

Si consideramos s = jΩ y dividimos la expresión de z en parte real e imaginaria, a medida que Ω varía entre menos infinito e infinito, los puntos en la parte izquierda del plano s (único semiplano que consideramos ya que asumimos estabilidad, explicado en los apartados 2.3.4 y 2.4.4.1) pasarán a estar dentro de la circunferencia de radio ½ y centrada en z = ½.

Las consecuencias de esta transformación son:

- Todos los polos correspondientes a un filtro digital estable se encontrarán dentro de la circunferencia anteriormente mencionada, por tanto, se tiene la característica de que un filtro estable en el dominio s pasa a ser un filtro estable en el dominio z.
- Debido a que todos los polos deben estar dentro de esta circunferencia de tan poco tamaño en comparación con la circunferencia unidad, la fase de estos polos está muy limitada, lo que solo permite el diseño de filtros paso bajo y paso banda hasta ciertas frecuencias.

.2.4.4.4 Diseño mediante la transformada bilineal Z (BZT)

Los métodos de diseño de filtros IIR de los dos apartados anteriores tienen la severa limitación de que solo se pueden emplear para el diseño de filtros paso bajo o paso banda bastante restringidos. Con este método se mapea el plano s hacia el plano z de forma que no se limita en banda el diseño del filtro.

Con la BZT se transforma el eje jΩ del dominio s en la circunferencia unidad del dominio z, por tanto, los puntos en el semiplano izquierdo pasan a estar dentro de la circunferencia unidad y los puntos en el semiplano derecho pasan a estar fuera. Estas características implican que no se produce aliasing en las componentes frecuenciales [18].

Si consideremos la función del sistema en forma de ecuación diferencial y en lugar de extraer la expresión para la derivada de primer orden (como en el método de aproximación mediante derivadas del apartado 2.4.4.3) integramos mediante la fórmula trapezoidal, se obtiene otra expresión para la variable z expresada en la ecuación (2.20) y otra para la variable s representada en la ecuación (2.21).

$$z = \frac{(2/T) + s}{(2/T) - s} \tag{2.20}$$

$$s = \frac{1 - z^{-1}}{T} \tag{2.21}$$

donde T representa el periodo de muestreo. Por tanto, con tan solo sustituir s por la expresión en función de z en la función de transferencia del filtro analógico, se obtiene la transformada bilineal en el dominio Z del mismo.

Las consecuencias de esta transformación son las siguientes:

- Un filtro analógico estable conlleva un filtro digital estable, ya que todos los polos en el semiplano izquierdo quedan mapeados en el interior de la circunferencia de radio uno.
- Hay una correspondencia entre frecuencias analógicas y digitales no lineal, por tanto, una vez conocidas las especificaciones en frecuencia del filtro digital a diseñar, es preciso realizar un proceso de "pre-warping" del filtro analógico del que se parte.

2.4.5 Transformaciones en frecuencia digitales

Este método se basa en que se ha podido realizar el diseño del filtro paso bajo digital necesario, de forma que se realiza una transformación en frecuencia del filtro paso bajo para obtener el filtro con las características deseadas. Denotando como z' a la variable z para el filtro LPF de partida y como z para el filtro objetivo, y de la misma forma f' para las frecuencias del filtro LPF y f para las frecuencias del filtro digital objetivo.

Al contrario que en los métodos descritos en los tres apartados previos, la transformación en frecuencia en este caso se realiza en el dominio digital mientras que en los demás métodos se partía de un filtro analógico ya transformado. Esta diferencia hace que podamos emplear cualquiera de los métodos descritos en los apartados 2.4.4.2, 2.4.4.3 y 2.4.4.4, ya que todos ellos pueden ser empleados para el diseño de filtro paso bajo digitales.

Para la transformación desde un filtro paso a bajo a otro filtro paso bajo con especificaciones de ancho de banda distintas, se emplean las correspondencias entre variables z mediante la ecuación (2.22) y entre frecuencias f y f', mediante la ecuación (2.23).

$$z' = \frac{z + \alpha}{1 + \alpha z} \quad |\alpha| < 1 \tag{2.22}$$

$$f' = \frac{1}{2\pi} \arctan\left[\frac{(1 - \alpha^2)\sin 2\pi f}{2\alpha + (1 + \alpha^2)\cos 2\pi f}\right] \tag{2.23}$$

de forma que podemos elegir α para desplazar las frecuencias del filtro LPF original hacia las frecuencias deseadas. Análogamente existen transformaciones para todos los tipos de filtros. Las ecuaciones (2.234) y (2.25) representan las correspondencias para filtro paso bajo.

$$z' = -\frac{z + \alpha}{1 + \alpha z} \quad |\alpha| < 1 \tag{2.24}$$

$$f' = \frac{1}{2\pi} \arctan\left[\frac{-(1 - \alpha^2)\sin 2\pi f}{-2\alpha - (1 + \alpha^2)\cos 2\pi f}\right] \tag{2.25}$$

Por otro lado, para el caso de filtros paso banda tenemos las ecuaciones (2.26) y (2.27).

$$z' = -\frac{1 + \frac{2\alpha k}{k+1} z + \frac{k-1}{k+1} z^2}{\frac{k-1}{k+1} + \frac{2\alpha k}{k+1} z + z^2} \quad |\alpha| < 1,\ k > 0 \tag{2.26}$$

$$f' = \frac{1}{2\pi} \arctan\left[\frac{(1 - b)\{2a \sin 2\pi f + (1 + b)\sin 4\pi f\}}{-a^2 - 2b - 2a(1 + b)\cos 2\pi f - (b^2 + 1)\cos 4\pi f}\right] \tag{2.27}$$

donde a = 2αk/(k + 1) y b = (k - 1)/(k + 1).

Y finalmente, las ecuaciones (2.28) y (2.29) corresponderían a las transformaciones para filtros de banda eliminada.

$$z' = \frac{1 + \frac{2\alpha k}{k+1} z + \frac{k-1}{k+1} z^2}{\frac{k-1}{k+1} + \frac{2\alpha k}{k+1} z + z^2} \quad |\alpha| < 1,\ k > 0 \tag{2.28}$$

$$f' = \frac{1}{2\pi} \arctan\left[-\frac{(1 - b)\{2a \sin 2\pi f + (1 + b)\sin 4\pi f\}}{a^2 + 2b + 2a(1 + b)\cos 2\pi f + (b^2 + 1)\cos 4\pi f}\right] \tag{2.29}$$

donde las variables a y b se calculan de la misma forma que en la transformación de paso bajo a paso banda [18].

Como observación final, es importante remarcar la fuerte relación que hay entre las transformaciones a paso banda y banda eliminada, ya que la variable z sufre las mismas variaciones y solo cambia la correspondencia de frecuencias.

3. Algoritmos genéticos y algoritmos meméticos

3.1 Introducción

3.1.1 Breve historia de los Algoritmos genéticos

La computación evolutiva ha tenido una especial relevancia en la última década, no obstante, lo orígenes se remontan a la década de los 50 con los trabajos de investigadores Bremermann, Friedberg o Box, de forma que han pasado algo más de tres décadas sin captar la atención de la comunidad científica. Esto se debió principalmente a la falta de ordenadores que ofrecieran la capacidad de computación necesaria para la convergencia de este tipo de algoritmos.
Durante la década de los 80, las mejoras en el rendimiento de los ordenadores permitieron la aplicación de los algoritmos genéticos a la resolución de problemas de optimización del mundo real, obteniendo resultados bastante prometedores que impulsaron la investigación en esta área [1].

Aunque la investigación acerca de los métodos genéticos cobró importancia, no fue hasta la década de los 90 cuando los distintos expertos en la materia empezaron a compartir sus avances con el fin de aportar perspectivas distintas a la variedad de problemas que surgían.

3.1.2 Motivación de los algoritmos genéticos

Debido al rápido desarrollo de la tecnología que nos ofrece una capacidad de computación extraordinaria, se ha retomado la tarea de tratar de resolver problemas que hace tan solo un par de décadas se consideraban irresolvibles, principalmente por las limitaciones en memoria disponible y velocidad de ejecución de algoritmos sobre fragmentos grandes de datos.

Muchos de estos problemas se basan en la optimización de parámetros, es decir, escoger el valor del parámetro que aplicado al problema nos permita ajustarnos con la máxima precisión posible a la solución que queremos obtener. Con respecto al proceso de optimización, los métodos convencionales nos ofrecen soluciones válidas al problema pero habitualmente quedan estancados en mínimos locales, es decir, soluciones considerablemente buenas y que en muchos casos de acercan a la óptima pero que pueden mejorarse.
En el área que nos ocupa, que es el desarrollo de filtros, optimizar las soluciones supone la disminución de la amplitud del rizado (valores entre los que oscila la ganancia en una banda) y reducir la ganancia en la banda eliminada.

3.1.3 Introducción a los algoritmos meméticos

Los algoritmos meméticos comienzan su aparición en los años ochenta, aunque es importante mencionar que ya se habían realizado diversos estudios relacionados con la computación evolutiva, un ejemplo de esto son los algoritmos genéticos. Los MA´s son una familia de metaheurísticas que pretenden recoger conceptos de distintas técnicas de resolución anteriores como pueden ser SA (recocido simulado) y TS (búsqueda tabú).

La palabra memético surge del término inglés “meme”, que no es más que la forma de denominar a la unidad teórica de información cultural transmisible de un individuo a otro, por otro lado en nuestro caso tiene un significado más enfocado a la teoría de la genérica, aludiendo a las características genéticas que se comparten y transmiten de una generación a la siguiente. Podemos entender entonces los algoritmos meméticos como un tipo de algoritmo donde existe una población, donde cada individuo presenta unas características especiales (fenotipo) que lo harán mas o menos apto y que podrán ser transmitidas a la siguiente generación en función de los beneficiosas que sean estas características [2].

Este tipo de algoritmos han captado la atención de investigadores en las últimas décadas debido a sus ventajas con respecto a los algoritmos evolutivos tradicionales ya que, al contrario que estos últimos, los MA están basados en explotar el conocimiento disponible hasta el momento sobre las soluciones al problema.

3.2 Términos empleados en MA.

Este tipo de algoritmos tienen una sintáctica bastante concreta para determinar a cada uno de los elementos que los forman, así como a los distintos procesos que tienen lugar en cada una de las iteraciones de los mismos. Por ello es importante presentar los conceptos que más se van a emplear, así como una descripción de sobre cuál es su papel en este tipo de algoritmos.

Agente: Es cada uno de los elementos que componen la población, de forma que cada uno de ellos representa una o varias soluciones al problema expuesto, los cuales deberán competir y cooperar de forma muy similar a lo que ocurre en la naturaleza con individuos de la misma especie o que ocupan un mismo ecosistema.

Fenotipo: Representa el conjunto de características que un agente tiene, las cuales podrán ser beneficiosas en distinto grado y que será transmitidas en parte a su descendencia en combinación con las de otro agente siguiendo criterios que están estrechamente relacionados con el problema que se está estudiando.

Fitness: Palabra inglesa cuyo significado es “aptitud”, que representa en qué grado un agente posee unas características beneficiosas frente a la resolución del problema, es decir, si su fenotipo es una solución potencial y cuánto se aproxima este al valor óptimo que se está buscando. El fitness vendrá dado por una función de fitness que evalúa el fenotipo de cada uno de los agentes, favoreciendo la supervivencia de las características beneficiosas de cara a la siguiente población.

Población: Se refiere al conjunto de agentes que existen en un momento concreto en el que desarrolla el algoritmo, es una extensión del concepto de individuo que tanto se emplea en el contexto de los algoritmos evolutivos.

3.3 Desde los EA a los MA

Los algoritmos meméticos pueden entenderse como la evolución de los algoritmos evolutivos tradicionales mediante la introducción de metaheurísticas estocásticas de búsqueda global, o dicho de otra forma: algoritmos evolutivos híbridos. La hibridación tiene como fin, bien acelerar la convergencia del algoritmo (los algoritmos evolutivos clásicos tardaban largos periodos de tiempo en encontrar soluciones apropiadas) o bien mejorar la calidad de estas soluciones (alcanzar máximos o mínimos globales en vez de estancarse en extremos locales).

La mayoría de los algoritmos meméticos emplean la búsqueda local como método de hibridación de los EA, de forma que no solo se tiene en cuenta el gran espacio de soluciones posibles si no que de algún modo se puede hacer hincapié en las soluciones más prometedoras de entre todas ellas.

Para ilustrar mejor estas diferencias podemos emplear un diagrama de flujo representado en la figura 3.1 en el cual se muestran las etapas de un algoritmo evolutivo clásico donde se han marcado con puntos negros los pasos donde se puede aplicar la anteriormente mencionada búsqueda local para llegar a un algoritmo memético.

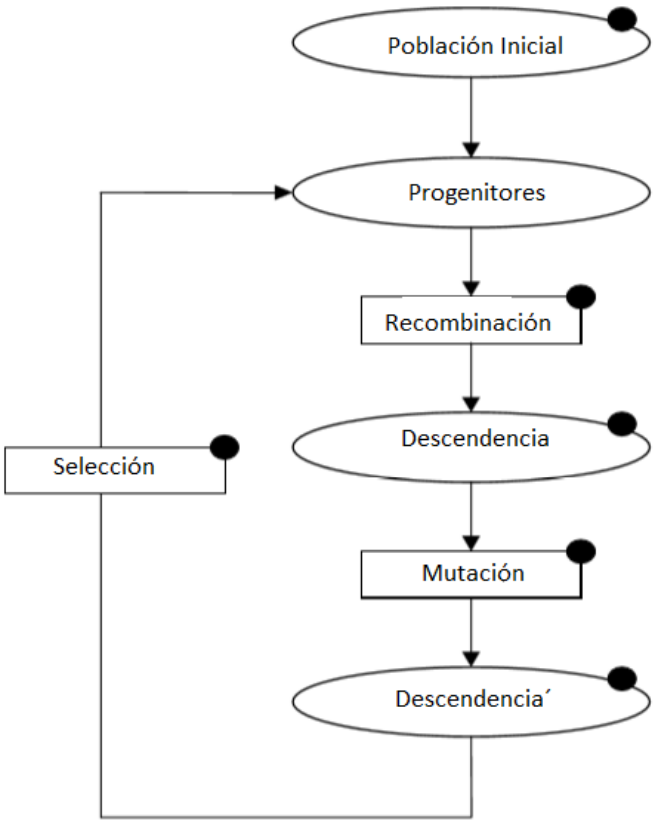

***Figura 3.1**. Diagrama de flujo de un algoritmo genético genérico.*

Fijándonos en el gráfico podemos advertir multitud de posibilidades de hibridación, como por ejemplo en la población inicial, la cual podemos iniciar con agentes con propiedades orientadas al problema en lugar de establecer una población completamente aleatoria.

Aunque el proceso de hibridación se puede aplicar a casi cualquiera de las etapas, se ha observado de forma empírica que el rendimiento del algoritmo no depende tanto del nivel de hibridación sino del conocimiento que tiene el algoritmo del problema específico. Las formas más habituales de hibridación son en la introducción de una o más fases de búsqueda local basadas en la selección de individuos en particular. Con esta fase surgen una serie de dilemas con respecto a la variación de la población como pueden ser el efecto Lamarckiano en contraposición con el Baldwiniano o la preservación de la diversidad enfrentada a la mejora de la velocidad en la búsqueda de soluciones.

3.3.1 Efecto Lamarckiano vs. efecto Baldwiniano.

Cuando ejecutamos un proceso de búsqueda local surge la decisión de qué hacer con las mejores soluciones que se han obtenido en cada etapa. Uno de los enfoques es el efecto Lamarckiano que consiste en sustituir el agente mejorado tras la búsqueda local por el agente previo con menor fitness (representa una peor solución al problema que su versión mejorada), en ese caso la información genética del segundo se pierde y queda sustituida. El otro enfoque

es el efecto Baldwiniano que se basa en mantener la información genética de la versión sin mejorar e igualar su fitness al de la versión mejorada tras la búsqueda local. [8]

Comparando ambos enfoques, en base a los resultados teóricos más recientes se ha llegado a la conclusión de que el método Lamarckiano acelera sustancialmente la convergencia del algoritmo, pero en muchos casos se llega a una convergencia prematura del mismo. Por otro lado, el enfoque Baldwiniano mantiene una mayor diversidad en la población evitando convergencias prematuras pero ralentizando considerablemente el proceso de convergencia.

3.3.2 Preservación de la diversidad.

Uno de los problemas que surge a la hora de elaborar un método de búsqueda local es la diversidad, esto se debe a que si introducimos métodos de mejora local muy agresivos podemos llegar a una población donde una gran porción de los agentes son localmente óptimos pero muy similares en cuanto a información genética dificultando la salida de este óptimo local para llegar a óptimos globales. Por lo tanto, se contemplan algunas de las soluciones para mantener la diversidad.

Cuando se inicializa la población, aunque a priori generar individuos orientados al problema puede parecer atractivo, presenta un problema ya que todos ellos mejorarán hacia un mismo punto perdiendo la diversidad del conjunto. Por ello es importante considerar una población inicial con una parte formada por agentes completamente aleatorios.

Con respecto a la búsqueda local, se ha comprobado empíricamente que una forma de mantener la diversidad es la introducción de varias etapas de búsqueda local donde cada una de ellas posee un espacio de búsqueda un mínimo local distinto.

Los operadores pueden ejecutar procesos de recombinación específicos del problema manteniendo siempre la diversidad por encima de un umbral y en caso de no llegar al mismo, ejecutar un proceso de re-inicialización, sustituyendo parte de la población de esa iteración por agentes nuevos con información aleatoria.

Finalmente se pueden realizar modificaciones sobre el operador de selección de forma que según el método de reconocimiento simulado podemos asignar valores de probabilidad de reproducción distintos de cero a agentes con un fitness muy bajo para así mantener una población con diversidad, ya que, aunque estas soluciones no sean aptas para el problema en cuestión en la iteración actual, si son óptimos en potencia.

3.4 Esqueleto de un MA

Existen infinidad de algoritmos meméticos para problemas concretos pero todos ellos poseen una serie de cualidades con respecto a las etapas que lo componen. Las etapas propias de un MA guardan una gran similitud con las que forma un GA, no obstante, los MA introducen algunas variaciones que los hacen más eficientes.

Todo MA debe partir de una población inicial, de forma que los individuos sean generados de forma aleatoria (o pseudoaleatoria en los casos en los que conocemos algunas de las características que deben presentar las soluciones al problema) cuyo número variará ampliamente en parte debido a la capacidad computacional de la que se disponga.

Seguidamente en cada una de las iteraciones del algoritmo, se aplicará la función de fitness evaluando a cada uno de los individuos y seleccionando aquellos cuyo fenotipo es más beneficioso (los agentes serán evaluados y se les asignará una probabilidad de "reproducción" conforme a su fitness de forma que aquellos que más se acercan a la solución deseada tienen una mayor probabilidad de transmitir su fenotipo a la siguiente población).

En cuanto a la cooperación, esta se consigue a través de la reproducción, de forma que se crean nuevos agentes a partir de los existentes, no obstante, existirán varias poblaciones intermedias, tantas como operadores de reproducción se estén empleando. Este proceso consiste en la creación de cada una de las poblaciones intermedias a partir de aplicarle el operador a la población intermedia anterior, creando una sola población final. [1, 2, 6]. Típicamente se emplean tres operadores: recombinación, mutación y búsqueda local.

3.4.1 Pseudocódigo

Un ejemplo típico de MA básico sería el de la figura 3.2, cuyo desarrollo se describe mediante pseudocódigo.

Algoritmo Memético

ENTRADA: una instancia I de un problema P.
SALIDA: una solución *sol*.

```
    // generar población inicial
1:  para j ← 1:popsize hacer
2:    sea ind ← GenerarSoluciónHeuristica (I)
3:    sea pop[j] ← MejoraLocal (ind, I)
4:  finpara
5:  repetir // bucle generacional
      // Selección
6:    sea criadores ← SeleccionarDePoblación (pop)
      // Reproducción segmentada
7:    sea auxpop[0] ← pop
8:    para j ← 1:#op hacer
9:      sea auxpop[j] ← AplicarOperador (op[j], auxpop[j − 1], I)
10:   finpara
11:   sea newpop ← auxpop[#op]
      // Reemplazo
12:   sea pop ← ActualizarProblación (pop, newpop)
      // Comprobar convergencia
13:   si Convergencia (pop) entonces
14:     sea pop ← RefrescarPoblación (pop, I)
15:   finsi
16: hasta CriterioTerminación (pop, I)
17: devolver Mejor (pop, I)
```

Figura 3.2. *Pseudocódigo que representa un algoritmo memético típico.*

Primero se genera la población inicial entre los pasos 1 y 4, de forma que se crea un individuo *ind* en base a una función denominada *GenerarSoluciónHeurística(I*) (función que posee información sobre el problema que se aborda con el fin de generar agentes que a priori son potenciales soluciones al problema), en el siguiente paso se aplica la función *MejoraLocal()* al agente y este queda añadido a la población.

En el paso 6 se seleccionan los futuros progenitores en base a la función *SeleccionarDePoblación(),* escogiendo aquellos con las mejores características para luego ser heredadas, seguidamente se emplea una variable auxiliar llamada *auxpop[]* que almacena la población tras cada uno de los operadores para aplicar el siguiente operador sobre la misma.

Del paso 8 al 10 se emplea un bucle para la ejecución de cada uno de los operadores, de forma que se actualiza cada valor de *auxpop[]* en base al operador de esa iteración y al valor de *auxpop[]* anterior. Tras aplicar todos lo operadores la población *newpop* se actualiza con el último elemento de *auxpop[]* (población a la que ya se le han aplicado todos los operadores).

Finalmente en los pasos 13, 14 y 15 se comprueba la convergencia con la función *Convergencia()* que recibe como parámetro la población actualizada, si esta ha convergido se

le aplica la función *RefrescarPoblación()* de forma que algunos agentes son sustituidos, con el fin de mantener la diversidad y evitar la monotonía de los fenotipos.

Para establecer un punto de terminación se introducen los pasos 16 y 17, que comprueban la condición de terminación con *CriterioTerminación()* y devuelven la mejor instancia de la población respectivamente.

3.4.2 Recombinación.

Mediante este proceso se obtienen agentes nuevos en base a los de la generación anterior extrayendo las cualidades relevantes contenidas en los agentes cooperantes. Esto es una ventaja con respecto a los EA convencionales los cuales empleaban la recombinación siguiendo unas pautas preestablecidas que en ocasiones podrían hacer que cualidades relevantes de ciertos individuos se perdieran debido a la uniformidad del método.

Los descendientes se generan a partir de la información contenida en un cierto número de soluciones o padres (habitualmente se emplean dos padres, pero este número puede verse modificado según el problema). Si se diera el caso de que los descendientes están formados enteramente por la información obtenida de los padres se dice que la recombinación está basada en la transmisión.
No obstante, esta situación en particular se da en escasas ocasiones ya que depende en gran medida del ámbito del problema [3] que se esté tratando, como puede ser el TSP (Traveling Salesman Problem). Si ese fuera el caso se pueden considerar otras características de interés como pueden ser la conservación o la reorganización.

Se dice que es el método es conservativo cuando el operador de recombinación genera descendencia que mantiene todas las características comunes de sus progenitores. Por otro lado se dice que un operador incorpora recombinación cuando la descendencia que genera posee cualquier combinación de las características de sus progenitores. No obstante, en cualquiera de los casos anteriores es importante tener en cuenta que la recombinación de la información debe llevarse a cabo teniendo en cuenta el problema concreto [4], nunca “a ciegas”.

3.4.3 Mutación.

Este es un operador clásico de los GAs, el cual tiene el papel de introducir información nueva en la siguiente población, ya que se sustituye parte de la información menos valiosa del fenotipo del agente por otra nueva dotada de cierta aleatoriedad. Esta introducción de nueva información debe realizarse paulatinamente ya que realizar grandes cambios de forma drástica llevaría el problema a la búsqueda entre soluciones prácticamente aleatorias [3].

Este es uno de los aspectos que diferencian a los MA de los GA, ya que los primeros tratan a toda costa de mantener las cualidades valiosas de los agentes, mientras que los segundos realizaban cambios aleatorios que podían hacer que soluciones potenciales se perdieran [2,3].

3.4.4 Búsqueda local.

Es una de las características más distintivas de los algoritmos meméticos, de forma que los individuos tras haberse recombinado y mutado colaboran entre sí para realizar mejoras autónomas de sus cualidades.

Finalmente, muchos MA introducen una fase de comprobación de la convergencia de las soluciones. En esta fase se comprueba en qué medida las soluciones disponibles en la población actual están correladas, es decir, qué similitud guardan entre ellas. Una de las medidas que se emplean es la entropía de Shannon, fijando un umbral bajo el cual se considera que la población ha convergido hacia una solución local, consecuentemente se aplica una función de refresco [3] que realiza cambios en la población mediante la sustitución de agentes existentes por otros nuevos.

3.5. Representación

La representación es una de las primeras características que debemos tener en cuenta a la hora de elaborar un MA ya que se refiere a la forma en que se expresan las posibles soluciones del problema. No tiene que ver necesariamente con su codificación, ya que este término está más orientado al almacenamiento y uso de memoria, sino que es una representación abstracta de las soluciones lo más adecuada posible al problema que se está atacando.
Cada una de las soluciones se puede entender como una combinación de determinadas unidades de información donde cada una de ellas tendrá sus propias características, ventajosas o no.

Si consideramos un problema cuyas soluciones son el conjunto de permutaciones de m elementos, estas estarán compuestas por diferentes tipos de información:

- Posicional: posición en la que se encuentra un elemento dentro de la solución.
- Precedencia: si un elemento aparece antes o después de otro elemento determinado.
- Adyacencia: si un elemento aparece junto a otro elemento determinado.

Cada tipo de información tiene un valor distinto dependiendo del problema que se esté tratando ya que habrá ocasiones en las que, por ejemplo, la adyacencia carezca de importancia como puede ser en la planificación de cadenas de montaje, mientras que en el problema del viajante de comercio la adyacencia es crucial. El caso contrario sería el de la información posicional, la cual es de vital importancia en la planificación de cadenas de montaje.

Tiendo en cuenta estos tipos de información, deberemos considerar una gran variedad de operadores de recombinación de forma que podamos conservar las características beneficiosas en base al problema que tratemos de resolver.

3.6 Operadores de recombinación.

Como se ha comentado anteriormente el tipo de información que es relevante para el problema debe ser considerado a la hora de escoger el operador de recombinación adecuado con el fin de no perder las características beneficiosas de los progenitores en el proceso. Este tipo de información relevante se debe en gran medida al problema que se trata de abordar pero principalmente a la representación que se haya escogido de las soluciones (agentes).

Algunos de los operadores de recombinación más conocidos son: edge-recombination(ER) partially mapped crossover (PMX) y uniform cycle crossover (UCX), los cuales se denominan operadores heurísticos constructivos, lo que implica que no son operadores “ciegos” sino que poseen cierto conocimiento del problema[una introducción a los AM Moscato].

3.6.1 Recombinación de bordes.

El operador de bordes (nombre traducido al español) emplea una tabla de extremo para llevar a cabo la recombinación. Esta tabla es una tabla de adyacencia que almacena los elementos de información adyacentes a cada uno de los tipos de elementos de información presentes en ambos padres. Esta tabla se emplea posteriormente para la creación de la descendencia de forma que se introduzcan el menor número de bordes ajenos a los padres, es decir, se hace énfasis en los bordes comunes entre ambos progenitores

Un ejemplo que ilustra esto de una forma bastante intuitiva sería el caso del problema del navegante. Suponiendo que existieran 12 ciudades y cada una de ellas estuviera representada por una letra, dos posibles padres serían:

Padre1:[g d m h b j f i a k e c] **Padre2**:[c e k a g b h i j f m d]

Las tabla de adyacencia correspondiente a estos progenitores concretos sería la tabla 3.1.

Ciudad	Lista de bordes
a	-k, g, i
b	-h, g, j
c	-e, d, g
d	-m, g, c
e	-k, -c
f	-j, m, i
g	a, b, c, d
h	-b, i, m
i	h, j, a, f
j	-f, i, b
k	-e, -a
m	-d, f, h

Tabla 3.1. *Tabla de adyacencia entre padre 1 y padre 2*

En la tabla 3.1 se representan cada una de las ciudades y las ciudades adyacentes a cada una de ellas, en caso de que una de ellas tenga una ciudad adyacente común a ambos progenitores esta se representa con el signo menos seguido de la letra que la identifica.

En la creación del descendiente se elige inicialmente una de las ciudades de forma aleatoria, eliminándola de la tabla allá donde aparezca (esto es crucial si consideramos como solución al problema una ruta donde no es posible pasar dos veces por la misma ciudad). La siguiente ciudad para la secuencia será elegida con ciertas prioridades, tendrán mayor prioridad las ciudades marcadas con el signo menos (ciudades adyacentes comunes a ambos progenitores), y después en el orden de prioridad irán las ciudades que tengan una lista de bordes con menos elementos.

Aunque el proceso de elección de cada uno de los elementos es aleatorio se asignan pesos según la prioridad, consecuentemente, si la primera ciudad fuera a, la siguiente ciudad más probable en la secuencia sería k, ya que es un borde común a ambos progenitores, seguida en probabilidad por i, ya que tiene menos elementos en su lista de bordes, y finalmente la ciudad g (no es un borde común y su lista de bordes está formada por cuatro elementos).

Siguiendo estos criterios llegaríamos a la secuencia [a k e c], de forma que tendríamos como opciones d y g para la siguiente ciudad, como ambas son candidatas con prioridad equivalente, se elige una de forma aleatoria. Finalmente llegaríamos a la secuencia [a k e c d m h b g], produciéndose un error ya que no existen más bordes en la lista de g, por tanto se elige una ciudad aleatoriamente entre las restantes.

Tras el proceso de recombinación, el descendiente estará formado por la secuencia:
[a k e c d m h b g i f j], de forma que han aparecido dos bordes que antes no existían; [g-i] y [j-a]. Estos borden aparecen como fruto de la recombinación, pero no se han heredado de ninguno de los progenitores [5].

3.6.2 Edge-3.

El sistema explicado anteriormente se denomina Edge-2 el cual incorpora ciertos problemas cuando encontramos un elemento en la secuencia que ya no tiene bordes en su lista, en ese caso decimos que es un terminal muerto, entendiendo como terminal a un elemento en cualquiera de ambos extremos de la secuencia. Contrariamente, un terminal vivo es aquel que aún posee bordes en su lista.

El método Edge 3 soluciona esta selección aleatoria cuando encontramos un terminal muerto que lleva a la aparición de bordes externos ajenos a los progenitores mediante el uso de terminales. Al igual que en el método anterior, el primer elemento se elige de forma aleatoria entre todas las posibilidades, el cual siempre será un terminal vivo, y se irán añadiendo elementos creando una subsecuencia hasta llegar a un terminal muerto.

Con el primer terminal muerto la subsecuencia se traspone y se siguen concatenando elementos mientras los terminales estén vivos, hasta llegar a un terminal muerto, entonces se separa esta subsecuencia y se empieza una nueva subsecuencia con la elección de un terminal vivo nuevo elegido aleatoriamente [5]. Este método se aplica recursivamente hasta tener la secuencia completa.

Si lo comparamos con el ejemplo empleado en Edge 2, al llegar a la secuencia [a k e c d m h b g], esta se traspone transformándose en la secuencia [g b h m d c e k a]. Continuando con la secuencia, el único borde disponible para el elemento a, es el borde i (en caso de existir más bordes disponibles estos se elegirían con el criterio de prioridad de Edge 2). Hasta el final de la secuencia, los elementos se irán añadiendo de la forma habitual hasta llegar a la secuencia [g b h m d c e k a i f j] la cual solo presenta un borde ajeno a los progenitores que sería el borde [j-g].

3.6.3 Partially mapped crossover (PMX).

Este tipo de operador de recombinación se basa en información posicional, es decir, al contrario que la operación de recombinación por bordes, se emplea en situaciones en las que la información de adyacencia no es relevante, esto lleva a la conclusión de que el PMX no es útil en problema análogos al del viajante de comercio (TSP) [5,6].

3.6.3.1 Mecánica.

Este operador se emplea para el intercambio de orden y valor de determinados elementos de información entre progenitores para la generación de descendencia. El nombre de cruce parcialmente mapeado se debe a que se intercambia un fragmento de ambos progenitores y se realiza una sustitución del resto de elementos de información.

Para ilustrar la mecánica de este operador vamos a exponer un ejemplo en particular. Supongamos dos cadenas que representan dos permutaciones de los números del uno al diez, creando los progenitores A y B que vienen definidos por la tabla 3.2

	Posición									
Cadena	1	2	3	4	5	6	7	8	9	10
A	9	8	4	5	6	7	1	3	2	10
B	8	7	1	2	3	10	9	5	4	6

Tabla 3.2. *Representación de los progenitores A y B, estableciendo el valor de cada una de sus 10 posiciones.*

Inicialmente se eligen dos posiciones de forma aleatoria, separadas siempre una longitud menor a la longitud de la cadena, entonces se extraen los elementos comprendidos entre ambas posiciones en cada una de las cadenas y se intercambian. Seguidamente se sustituyen los valores de los elementos restantes por los de sección original que se ha sustituido con el fin de evitar repeticiones en los valores de los elementos de información.

En este caso particular podemos emplear la los índices 4 y 6, consecuentemente los segmentos que se intercambiarán serán [5 6 7] del progenitor A y [2 3 10] del progenitor B. Finalmente los elementos restantes en cada uno de los progenitores serán sustituidos por los de la sección original que poseían, por ejemplo: en el progenitor A el 10 de la posición 10 será sustituido por

un 7, el 3 de la posición 8 será sustituido por un 6 y el 2 en la posición 9 será sustituido por un 5. El proceso se ejecuta de forma análoga para el progenitor B.

Las cadenas resultantes serán:

Cadena A: 9 8 4 2 3 10 1 6 5 7
Cadena B: 8 10 1 5 6 7 9 2 4 3

3.6.3.2 Rendimiento.

Como en cualquier operador de recombinación tratamos de crear una descendencia a partir de la población anterior con el fin de mejorar las características de la población anterior, pero para ello es crucial que las características bondadosas se mantengan tras la recombinación por tanto es interesante analizar la probabilidad de estos empleando el PMX. Para ello se va a comprobar la probabilidad de supervivencia de un esquema beneficioso, es decir, una secuencia concreta de elementos de información dentro del agente los cuales no son necesariamente adyacentes.

Suponemos inicialmente un agente con l elementos de información, seguidamente contemplamos los casos de que el esquema no tenga ninguna posición fija, una posición fija, dos posiciones fijas, etc… Por tanto, llegamos a la conclusión de que el número de esquemas con j posiciones fijas en un agente de longitud l (l sobre j), multiplicadas por las posibles permutaciones de esquemas de longitud j en un agente de longitud l da como resultado:
Introduciendo el sumatorio desde 0 a l (posibles longitudes de un agente):

Por otro lado, consideramos la probabilidad de supervivencia de un esquema tras ejecutar el PMX, la cual se puede expresar mediante la probabilidad condicionada de tres sucesos mutuamente excluyentes: suceso en el que el esquema está completamente contenido en el fragmento a intercambiar, suceso en el que el esquema está completamente fuera del segmento a intercambiar y por último el suceso en el que el esquema queda cortado por el segmento a intercambiar. Entonces las posibilidad de supervivencia del esquema vendrá dada por la ecuación 3.1.

$$P(S) = P(S|D)P(D) + P(S|F)P(F) + P(S|C)P(C) \quad (3.1)$$

Donde D expresa la situación en la que el esquema queda dentro del segmento, F expresa la situación en la que queda fuera y C en la que queda fragmentado (parte queda dentro y parte queda fuera). Si analizamos la expresión podemos advertir que la probabilidad de que el esquema sobreviva habiendo sido fragmentado (P(S|C)) es prácticamente nula por ello la consideramos 0.

Asumiendo una longitud de segmento K, una longitud del esquema L(s) y un número de posiciones fijas dado por O(s), podemos expresar la probabilidad de supervivencia en un agente de longitud l mediante la ecuación 3.2.

$$\mathbf{P(S) = K - L + 1l+l - K -L + 1l(1-K + 1l)} \quad (3.2)$$

Si analizamos la expresión anterior podemos afirmar que la probabilidad de supervivencia de un esquema es inversamente proporcional a la longitud del mismo (expresada mediante la variable L) por tanto el esquema con más probabilidades de sobrevivir será aquel cuya longitud sea pequeña comparada con la anchura K del intervalo de selección.

Como conclusión obtenemos que los esquemas más propensos a sobrevivir son aquellos de longitud reducida comparada con la del segmento seleccionado para la sustitución y cuyo número de posiciones fijas es mucho menor que la longitud del agente. Esto es debido principalmente a que esquemas con menor número de elementos tienen menores probabilidades de que alguno de sus elementos sea sustituido o quede sesgado.

3.6.4 Variant of Partially Mapped Crossover (VPMX).

Este tipo de operador como el propio nombre indica es una variante del operador PMX, con el fin de buscar un intercambio entre progenitores algo más complejo ya que en este caso los segmentos que se emplean no afectan a las mismas posiciones de elementos de información.

3.6.4.1 Mecánica.

Este operador difiere con respecto al operador PMX en las posiciones de los elementos de información que se ven afectadas ya que la longitud de los segmentos elegidos debe ser igual (la descendencia deberá tener una longitud constante entre iteraciones e igual a la de sus progenitores) [6]. Por tanto para cada progenitor se elige dos posiciones aleatorias separadas la longitud del los segmentos que se intercambiarán.

Suponemos dos progenitores idénticos a los del apartado anterior sobre el operador PMX del apartado 3.6.3.

	Posición									
Cadena	1	2	3	4	5	6	7	8	9	10
A	9	8	4	5	6	7	1	3	2	10
B	8	7	1	2	3	10	9	5	4	6

Tabla 3.2. *Representación de los progenitores A y B, estableciendo el valor de cada una de sus 10 posiciones.*

Como primer paso elegimos segmentos de longitud 3, seguidamente establecemos que posiciones se verán afectadas en cada uno de los progenitores. Vamos a tomar en el caso del progenitor A el segmento [4 5 6] y en el progenitor B tomamos [9 5 4], lo cuales afectan a posiciones distintas.

Después se intercambian dichas posiciones llevando a los progenitores:

Progenitor A´: 9 8 4 **9 5 4** 1 3 2 10

Progenitor B´: 8 7 1 2 3 10 **4 5 6** 6

Donde las posiciones que se han visto afectadas por el intercambio se han resaltado en negrita. Finalmente se sustituyen los elementos de información repetidos ajenos a los segmentos intercambiados, llevando a dos posibles descendientes:

Descendiente A: 6 8 7 9 5 4 1 3 2 10

Descendiente B: 8 7 1 2 3 10 4 5 6 9

3.6.4.2 Comparativa PMX vs. VPMX

Este par de parámetros de recombinación se han empleado mayoritariamente para alcanzar soluciones óptimas a problemas contenidos en el TSPLIB (Traveling Salesman Problem Library), el cual se puede considerar como una “librería” sobre problemas estrechamente relacionados con el problema del navegante. Entre los problemas que TSPLIB contiene están: gr24, swiss4r2, eil51, eil76 y kroA1000 [6, 7].

En la mayoría de los experimentos llevados a cabo por Kusum Deep y Hadush Mebrahtu, el operador VPMX con operador de mutación de desplazamiento invertido ha obtenido los mejores resultados para este tipo de problemas. Se entiende como mejores resultados a aquellos

que han ofrecido un porcentaje de error menor, calculado como el cociente entre la diferencia entre el valor obtenido y el valor óptimo entre el valor óptimo. La fórmula correspondiente vendrá dada por la ecuación (3.2).

$$Porcentage\ de\ error\ (\%) = \frac{Valor\ obtenido - Valor\ óptimo}{Valor\ óptimo} * 100 \qquad (3.2)$$

Para visualizar mejor esta comparación podemos observar los resultados que se obtuvieron en la solución al problema swiss42, ya que es el caso donde más se aprecia la efectividad del operador VPMXID (VPMX con mutación por inversión desplazada).

	Tipo de operador	Cost and time /sec	Generaciones					Mejor y %
			1000	5000	10000	20000	30000	
PMX	Inversión	min	1425	1345	1345	1345	1345	1345 (5.7)
		max	2900	2900	2900	2900	2900	
		time	3	18	34	68	103	
	Inversión desplazada	min	1384	1336	1335	1335	1335	1335 (4.8)
		max	2238	2331	2331	2331	2331	
		time	4	20	36	70	107	
VPMX	Inversión	min	1432	1383	1383	1383	1383	1383 (8.6)
		max	2702	2702	2702	2702	2788	
		time	4	23	43	84	136	
	Inversión desplazada	min	1334	1295	1286	1273	1273	1273 (0)
		max	2600	2824	3066	3066	3066	
		time	5	24	45	89	134	

Tabla 3.3. *Comparativa entre los operadores PMX y VPMX.*

Como podemos ver, este último presenta un porcentaje de error del cero por ciento, es decir, se ha alcanzado la solución óptima y con una carga computacional muy similar a la del resto de operadores ya que estos resultados fueron obtenidos mediante la misma máquina con las mismas especificaciones técnicas descritas en [7].

3.6.5 Cyrcle crossover

Este tipo de recombinación se emplea en problemas con codificación por permutación. Consiste en combinar información de ambos progenitores para crear descendientes de forma que se transmiten las unidades de información junto con su posición.

3.6.5.1 Mecánica

Para seleccionar qué segmentos se intercambian de cada uno de los progenitores se establecen ciclos de forma que no se repitan elementos de información en los descendientes. Primero se escoge el primer elemento del padre A, se observa el valor del elemento de la primera posición en el padre B, seguidamente se va hasta el elemento con el mismo valor en el padre A, después se mira el valor del elemento en el padre B con la misma posición, y así hasta llegar al primer elemento del padre A. Finalmente, cada uno de los elementos por los que se ha pasado se intercambian entre los progenitores [12] para formar dos descendientes, este proceso se muestra en la figura 3.3.

Para ilustrar mejor este operador hemos contemplado un ejemplo concreto donde los progenitores A y B y el proceso de recombinación se muestran en la figura 3.3.

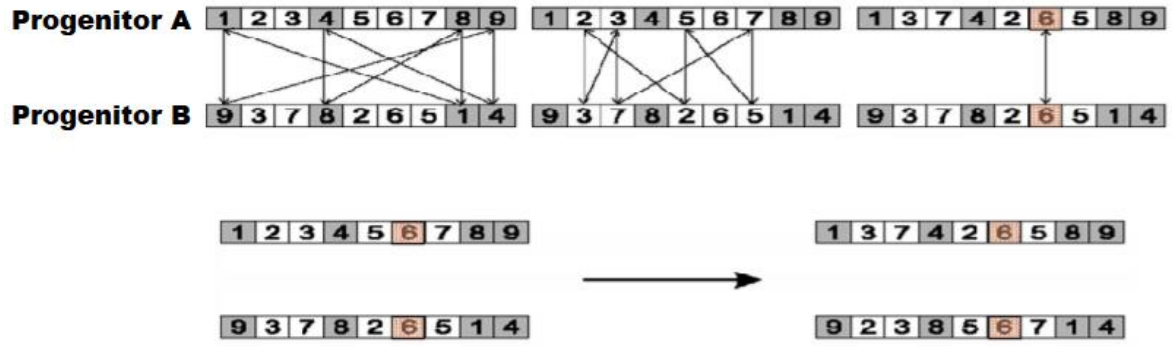

Figura 3.3. *Representación del funcionamiento del operador cíclico [12].*

En este ejemplo concreto el proceso es el siguiente:

2 Se escoge el elemento en la primera posición del progenitor A con valor 1
3 El primer elemento del progenitor B tiene el valor 9.
4 El valor nueve se encuentra en la posición 9 del progenitor A.
5 En esta posición en el progenitor B se encuentra el valor 4.
6 El valor 4 en el progenitor A se encuentra en la posición 4.
7 La posición 4 en el progenitor B se corresponde con el valor 8.
8 El valor 8 en el progenitor A se corresponde con la posición 8.
9 La posición 8 en el progenitor B tiene el valor 1 (primer elemento del progenitor A).

4. Algoritmos bioinspirados

El diseño de filtros tanto de respuesta finita al impulso (FIR) como de respuesta infinita al impulso (IIR) se presenta como un problema que puede involucrar una gran cantidad de parámetros, cuya optimización puede ser compleja y en la mayoría de las ocasiones imposible de abordar mediante algoritmos inteligentes clásicos. [9, 10]

Hasta el momento se han empleado una gran diversidad de métodos para el diseño de filtros, tales como: alineación simulada (SA), optimización de enjambre de partículas (PSO), algoritmos genéticos (GA) y optimización de enjambre de partículas cuánticas (QPSO). Sin embargo, cada uno de ellos posee ciertas limitaciones o inconvenientes. En el caso de los algoritmos genéticos, presentan una implementación bastante compleja en la mayoría de las situaciones además de una lenta convergencia. Por otro lado, la optimización mediante enjambre de partículas es un algoritmo de búsqueda aleatoria que se ha aplicado a multitud de problemas del mundo real pero como demostró Van de Bergh [9] no es un algoritmo de convergencia global.

4.1 Optimización de enjambre de partículas caóticas.

El CPSO es una técnica de búsqueda estocástica global la cual es capaz de encontrar un óptimo global mejorando el tiempo de búsqueda del clásico PSO (Particle Swarm Optimization) e incluso con un rendimiento superior al del método QPSO (Quantum Particle Swarm Optimizacion).

Este algoritmo está estrechamente relacionado con el algoritmo PSO, ya que comparte sus características básicas, pero ha introducido la característica del movimiento caótico. Por otro lado, también puede verse como una mejora del algoritmo de enjambre de partículas cuánticas con la introducción de la ya mencionada ecuación del caos con lo que al diseño de filtros FIR se refiere.

4.1.1 Algoritmo PSO

El PSO es un sistema de optimización multi agente que se basa en la metáfora del comportamiento social ya que las partículas o agentes tenderán a moverse hacia mejores posiciones basadas en lo movimientos del resto de partículas.

Cada agente llamado partícula “vuela” (empleamos este término debido al calificativo de enjambre refiriéndonos al conjunto de partículas) en un espacio S de D dimensiones basándose en su comportamiento previo y en el del resto de partículas del enjambre.

En el enjambre habrá m partículas que vuelan por un espacio con un número de dimensiones igual al número de parámetros que tiene el problema, por tanto la partícula i-ésima tendrá una posición $x=(x_{i1}, x_{i2},...,x_{iD})$ con i=1,2,...,m que será una posible solución al problema. La partícula i-ésima posee una velocidad $v=(v_{i1},v_{i2},...,v_{iD})$ y la mejor posición hasta el momento de esta partícula es $Pi=(p_{i1},p_{i2},...,p_{iD})$, de forma adicional la mejor posición descubierta por el sistema hasta el momento viene dada por $Pg=(p_{g1},p_{g2},...,p_{gD})$.

En cada iteración cada partícula actualiza su posición mediante la ecuación (4.1), como la posición anterior más la velocidad en la iteración actual dada por la ecuación (4.2).

$$V_i^{t+1}= wv_i^t + c_1r_1(p_i^t-x_i^t) + c_2r_2(p_g^t-x_i^t) \quad (4.1)$$

$$X_i^{t+1}=v_i^{t+1}+x_i^t \quad (4.2)$$

Donde r_1 y r_2 son valores aleatorios entre 0 y 1, el superíndice *t+1* y t representan el número de iteraciones (de forma que t+1 es la iteración siguiente a t). Los valores de c_1 y c_2 determinan el grado de influencia relativo de los puntos P_g y P_i con lo que al vuelo se refiere. Finalmente, w representa el peso inercial que ajusta dinámicamente la velocidad de las partículas.

4.1.2 Algoritmo CRPSO

Con el fin de evitar las limitaciones que el algoritmo PSO presenta, se introduce una componente de "locura" o aleatoriedad (las siglas corresponden con el nombre en inglés "craziness particle swarm optimization"), de esta forma la velocidad de los individuos puede cambiar drásticamente y de forma imprevisible para evitar mínimos locales. Esta nueva velocidad vendrá dada por la ecuación (4.3), donde r_1, r_2 y r_3 son valores aleatorios en el intervalo [0,1] y $sign(r_3)$ representa una función cuyo valor es -1 si $r_3<0.05$ y 1 en caso contrario [25].

$$V_i^{(k+1)} = r_2 * sign(r_3) * V_i^{(k)} + (1-r_2) * C_1 * r_1 * \left\{ pbest_i^{(k)} - S_i^{(k)} \right\} + (1-r_2) * C_2 * (1-r_1) * \left\{ gbest^{(k)} - S_i^{(k)} \right\} \quad (4.3)$$

Esta función sign() introduce aún más aleatoriedad a la velocidad ya que es capaz de cambiar el sentido del movimiento, introduciendo la posibilidad de que el individuo se mueva en dirección contraria a pbest (mejor individuo en la iteración actual) y gbest (mejor individuo hasta el momento), aumentando el espacio de búsqueda [25]. Esto a su vez implica que el algoritmo salga de un mínimo local si tanto pbest como gbest se encuentran en dicho mínimo. De nuevo, como se puede ver en la ecuación (4.4) se introduce otra componente aleatoria

dada por sign(r4) de forma análoga a la ecuación (4.3), pero en este caso el valor de decisión es de 0.5.

$$V_i^{(k+1)} = V_i^{(k+1)} + P(r_4) * sign(r_4) * v_i^{craziness} \tag{4.4}$$

4.1.3 Aplicación a diseño de filtros IIR

Como ejemplo para comparar el rendimiento del algoritmo CRPSO se ha tratado de aproximar la respuesta de un filtro de orden 5 dado por la ecuación (4.5) mediante un filtro de orden inferior, en este caso uno de orden 4. El resultado obtenido mediante el algoritmo CRPSO se ha comparado con el de otros algoritmos para el diseño de filtros IIR como: RGA, PSO y DE. Como medida del error entre el filtro de orden 5 preestablecido y el filtro de orden 4 obtenido se ha empleado el error cuadrático medio o MSE entre ambos. La gráfica obtenida en [25] que representa el MSE a lo largo de las iteraciones para los algoritmos comparados viene dada por la figura 4.1.

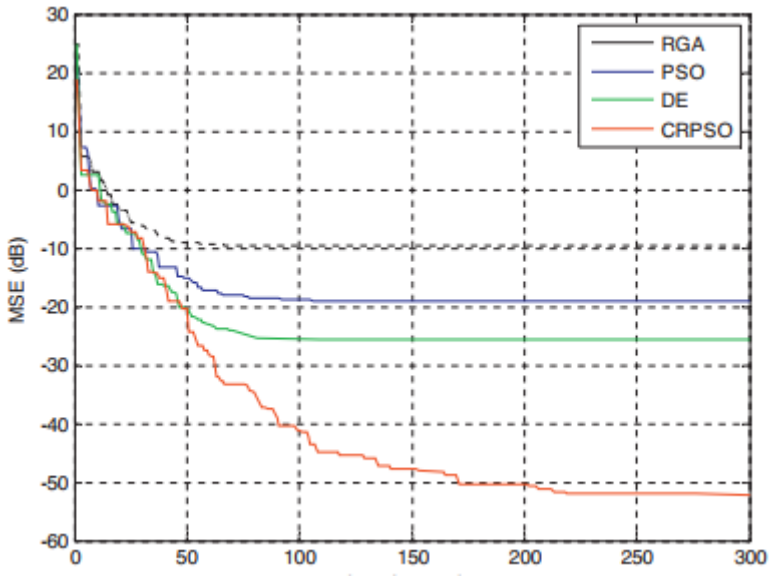

Figura 4.1. *MSE frente a número de iteraciones para algoritmos RGA, PSO, DE y CRPSO.*

Como se puede ver en la figura 4.1 el algoritmo CRPSO es el que mejores resultados obtiene con una MSE final entorno a -52dB, ínfima en comparación con la obtenida con los algoritmos RGA (-10dB), PSO (-19db) y DE (-26db). Esta mejora con respecto al resto de algoritmos viene dad principalmente por la capacidad del algoritmo CRPSO de evitar mínimos locales, al contrario que el resto de algoritmos, cuya convergencia quedó “estancada” entre las 50 y 100 iteraciones.

Como conclusión se obtiene que el algoritmo CRPSO es bastante eficiente para la elaboración de filtros IIR ya que es capaz de escapar de mínimos locales por su comportamiento caótico.

4.2 Algoritmo de evolución diferencial.

4.2.1 Mecánica

Inicialmente suponemos un espacio de *N* dimensiones, tantas como parámetros tenga el problema y un número de soluciones *S*, que formarán la población que siempre se mantendrá constante en número. Cada una de las soluciones al problema representa un vector con N componentes: x_{iG} donde $i=0, 1, ..., S-1$ y *G* representa la generación a la que pertenece el agente.

La población inicial se basa en agentes cuyas componentes son elegidas en base a una distribución aleatoria normal, siempre y cuando no se tenga conocimiento de una posible solución al problema. En ese caso las soluciones se formarán añadiendo cantidades aleatorias a las componentes de la solución inicial. Asumiremos de ahora en adelante que todos los valores aleatorios se obtienen mediante una distribución de probabilidad uniforme.

En cada iteración se evalúa a los agentes donde el mejor de todos ellos se le denomina por x_{bestG} con el fin de llevar un registro del desarrollo del algoritmo. Una característica remarcable de este algoritmo es que en cada iteración se extraen características de la población actual para la elección de las desviaciones aleatorias que se llevan a cabo en el proceso de reproducción y recombinación.

Existen varios esquemas de este tipo de algoritmos en los cuales varía la función que establece el valor del vector v. Este valor siempre depende de la diferencia de las componentes de dos o más agentes de la población escogidos aleatoriamente, de ahí el calificativo de "diferencial". En este caso vamos a tratar el funcionamiento del algoritmo diferencial con la ecuación (4.5):

$$\mathbf{V=x_{r1,G} + F*(x_{r2,G} – x_{r3,G})} \qquad (4.5)$$

Los valores de r_1, r_2 y r_3 son escogidos aleatoriamente del intervalo [0, S-1], es decir, cogemos tres vectores (agentes) de la población actual. Por otro lado F es número entero y positivo que controla la amplificación del factor diferencial ($x_{r2,G} – x_{r3,G}$). Para ilustrar esto vamos a incluir un ejemplo gráfico de un sistema con 2 dimensiones (consecuentemente dos parámetros a optimizar) donde se ha calculado el vector v.

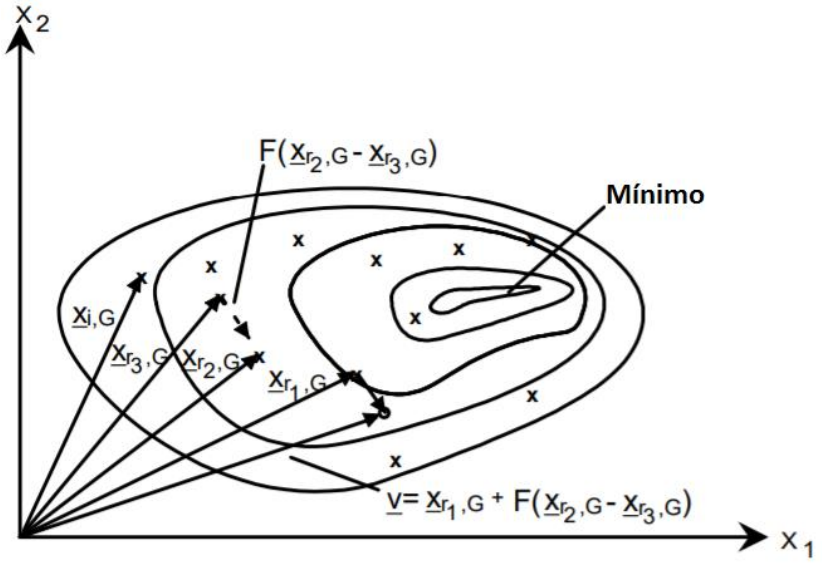

Figura 4.2. *Representación gráfica del cálculo de los vectores diferenciales.*

En el gráfico de la figura 4.2 se muestran tres vectores aleatorios $xr_{1,G}$ $xr_{2,G}$ $xr_{2,G}$ y el vector v representado como un círculo.

Tras definir el vector v se procede a la recombinación entre este vector y otro agente aleatorio de entre la población. Se elige un valor para el parámetro n de entre el intervalo [0,N-1] que representará el índice en el que comienza el segmento a recombinar, además se elige otro parámetro L de entre el mismo intervalo, el cual representa la longitud del intervalo a recombinar, con probabilidad $P_r(L=t) = (CR)t$, siendo CR la probabilidad de crossover (siempre entre 0 y 1). A continuación, se ilustra un ejemplo en la figura 4.3 con n=2, L=3 y D=7.

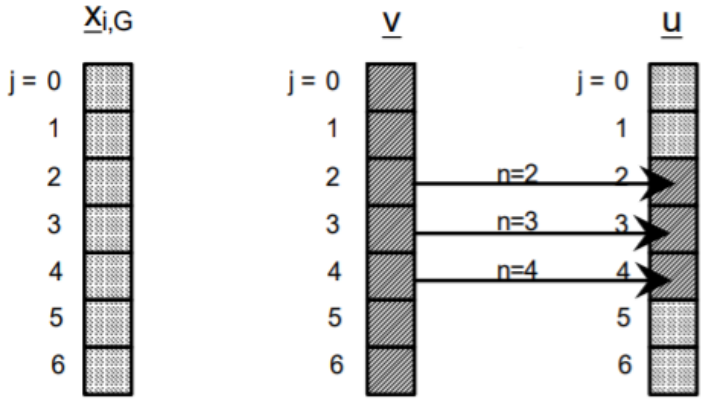

Figura 4.3. *Representación gráfica de la recombinación empleada.*

Como podemos observar se extrae el segmento de v delimitado por los parámetros anteriormente seleccionados y se sustituye por los elementos con los mismo índices en el agente x_{iG} para formar el agente u. Finalmente se les aplica a ambos las función de fitness y se

determina si u es ventajoso con respecto a x_{iG}, si esto es así este queda sustituido por el agente mejorado, en caso contrario se mantiene el agente x_{iG}.

Este proceso se repite hasta comprar cada uno de los agentes de la población de una iteración, es decir, en cada iteración se calculan tantos vectores v como agentes haya.

4.2.2 Esquema adicional

En el caso anterior se ha mostrado un ejemplo para una función generadora del vector v concreta, no obstante esta puede escogerse en base al problema que se esté tratando de abordar. Por tanto se hablará de evolución diferencial siempre y cuando el vector v sea función de una diferencia de agentes de la población. Otra forma de determinar este vector es la ilustrada en la ecuación (4.6).

$$\mathbf{V=x_{i,G} + \lambda*(x_{best,G} - x_{ri,G})+ F*(x_{r2,G} - x_{r3,G})} \quad (4.6)$$

De forma que se introduce un parámetro lamda adicional para determinar en qué medida cada nuevo elemento se acercará hacia el mejor valor hasta el momento, estableciendo una mayor finura en la configuración del algoritmo. El proceso de recombinación se lleva a cabo de la misma forma que en el caso anterior de por tanto lo único que cambia es la expresión para determinar v.

4.2.3 Aplicación a diseño de filtros IIR

En [10] este algoritmo se denomina DC o de diferencia cultural, no obstante, las bases de su funcionamiento son exactamente las mismas que para el algoritmo denominado DE. Este se ha empleado en [10] tanto para el diseño de filtros FIR como de filtros IIR, sin embargo, los resultados que se van a presentar son los obtenidos para filtros IIR que es el problema que nos ocupa.

Al igual que en el resto de algoritmos meméticos el DE se compara con otros algoritmos genéticos y meméticos clásicos y bien conocidos, en este caso con QPSO y AQPSO. Para medir la calidad de los resultados obtenidos en [10] se ha tratado de generar un filtro paso bajo con banda de paso hasta 0.4 y banda eliminada a partir de 0.47, todas ellas frecuencias normalizadas. En la figura 4.4 se muestran los resultados obtenidos para los tres algoritmos representando el valor de la función de fitness a minimizar frente al número de iteraciones.

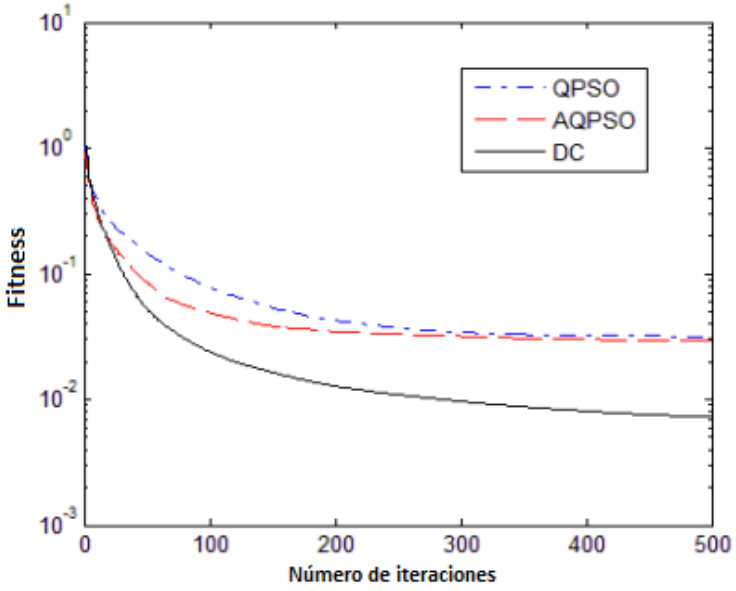

Figura 4.4. *Fitness obtenido para los distintos algoritmos meméticos comparados.*

Observando el gráfico de la figura 4.4 podemos afirmar que los mejores resultados se obtienen para el algoritmo DC propuesto ya que este alcanza un fitness final mucho menor en comparación con el resto de algoritmos que quedan atascados en mínimos locales en valores muy similares entorno a 2e-2. Por otro lado, el algoritmo DC alcanza valores entorno a 7e-3.

4.3 Algoritmo de colonia de hormigas (ACO)

Este es un algoritmo introducido por Dorigo y que tiene sus bases en el Ant System (AS) ambos inspirados en el comportamiento natural de las hormigas, las cuales buscan el camino más corto hacia una fuente de alimento. Se considera una metaheurística ya que las hormigas se comunican entre ellas para compartir sus experiencias sobre los mejores caminos dejando feromonas a su paso, de forma que las hormigas tenderán a guiarse por los caminos más transitados. [14,15,16, 25]

En cada iteración cada hormiga elabora una solución propia de forma que deja feromonas por el camino que recorre, dejando un menor número de las mismas en el inicio y una mayor cantidad a medida que se acerca a una buena solución para que otras hormigas puedan detectar que esa es una buena trayectoria. El diagrama de flujo que ilustra esto es el de la figura 4.5 [14].

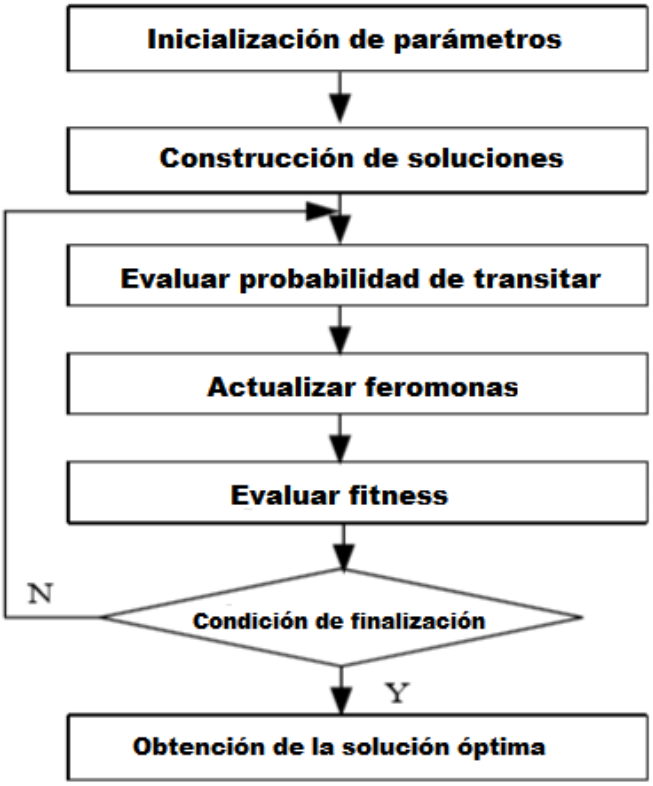

Figura 4.5. *Diagrama de flujo del algoritmo de colonia de hormigas (ACO).*

4.4.1 Decisiones.

Cada una de las hormigas representa una de las posibles soluciones al problema, caracterizada por una secuencia de bits, de forma que cada vez que una hormiga realiza un movimiento en su posición deberá decidir si cada uno se sus bits cambia o mantiene su valor anterior. Esta decisión se lleva a cabo mediante la evaluación de la cantidad de feromona que existe en cada uno de los caminos posibles, de forma que un camino con mayor cantidad de feromonas tendrá mayor probabilidad de hacer que el resto de las hormigas cambien su valor para adaptarlo al de este camino.

Supongamos un camino aleatorio ilustrado en la figura 4.6, donde la hormiga elige entre 0 y 1 en cada uno de los pasos de forma que la cadena resultante es la que se muestra. Establecemos la notación para los pasos como a-b de forma que se indica que la hormiga da un paso desde a hasta b.

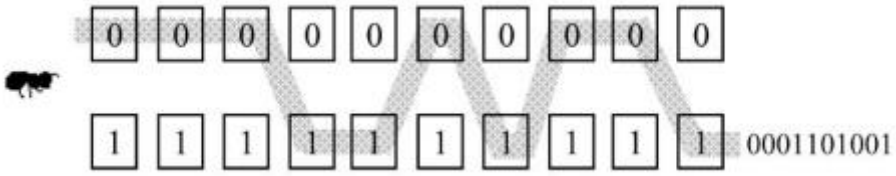

Figura 4.6. *Camino hipotético escogido por una hormiga cualquiera.*

Para cada decisión se emplea la fórmula dada en la ecuación (4.7), donde p_{01} representa la probabilidad de tomar el paso 0-1, τ_{00} y τ_{01} representan las feromonas de los pasos 0-0 y 0-1.

$$P_{01}(t) = \frac{\tau_{01}}{\tau_{01} + \tau_{00}} \tag{4.7}$$

Cada vez que una hormiga recorre un camino esta actualiza las feromonas del mismo siguiendo la ecuación (4.8), por tanto, si el camino es recorrido las feromonas aumentan, no obstante, si la hormiga no pasa por ese camino no se depositan feromonas.

$$\Delta\tau_{01}^{k}(t, t+1) = \begin{cases} \frac{Q}{F_k} & \text{si la hormiga k recorre el paso 0->1} \\ 0 & \text{en caso contrario} \end{cases} \tag{4.8}$$

donde k es la hormiga que recorre el paso, Q es una constante y F_k es la función de fitness calculada a partir de la hormiga *k*. Cuando todas las hormigas se han movido y han actualizado las feromonas de cada camino, la cantidad total de feromonas de cada camino es la suma de las feromonas depositadas por todas las hormigas que han pasado por el mismo.

4.4.2 Actualización de feromonas

Con el fin de mejorar la solución, los rastros de feromonas deben ser actualizados tanto de forma local como global según la fórmula dada por la ecuación (4.9).

$$\tau_{01}(t+1) = \rho\tau_{01}(t) + \Delta\tau_{01}(t, t+1), \tag{4.9}$$

En esta fórmula se incluye el ratio de evaporación de las feromonas ρ (escogido entre 0 y 1) y la cantidad de feromona añadida por la hormiga k en el paso 0-1 en el intervalo temporal entre t y t+1 dada por $\Delta\tau_{01}$(t,t+1) [14,15, 25], cuya expresión viene descrita por la ecuación (4.8).

4.4.3 ACO modificado

En [25] se propone una mejora del algoritmo ACO básico que se describe en los apartados anterior. Uno de los problemas que parecen en el ACO básico es que cuando una hormiga encuentra rápidamente un buen camino que le lleva a una solución aceptable, todas las hormigas comienzan a seguirla sin tener en cuenta la longitud del mismo.

Esto hace que se alcancen mínimos locales de forma rápida ya que el movimiento de las hormigas se rige únicamente por la cantidad de feromonas depositadas en los distintos caminos, por tanto, en [25] se propone la introducción de una memoria que almacena los movimientos anteriores de las hormigas, todo ello inspirado por el algoritmo de búsqueda tabú tratado en el apartado 4.7.

Para la mejora del ACO se introduce una limitación de frecuencia, es decir, se analiza con qué frecuencia un camino es escogido, por tanto, si muchas hormigas recorren un determinado camino, la probabilidad de que las siguientes hormigas lo hagan se reduce. Empleando la ecuación (4.10) se manipula la probabilidad de que cada hormiga vuelva a visitar un determinado camino.

$$p_{01}(t) = \begin{cases} 1 & \text{Si } (f * f_{01} < f_{00}) \\ \dfrac{\tau_{01}}{\tau_{01} + \tau_{00}} & \text{en caso contrario} \end{cases} \tag{4.10}$$

donde f es el factor de frecuencia (siempre mayor o igual que uno [25]). Si la condición $f*f_{01}<f00$ se cumple, entonces el paso 0-1 se escoge con probabilidad 1, en caso contrario se emplea el método clásico de decisión por feromonas de la ecuación (4.15), esto implica que el paso 0-0 se encuentra en la lista tabú y no puede ser elegido.

4.4.4 Aplicación a diseño de filtros IIR

En [25] se evalúa tanto el funcionamiento del ACO básico como el del ACO modificado comparándolos con los resultados obtenidos para un algoritmo de búsqueda tabú. Para cuantificar el rendimiento de cada uno de ellos se establece un filtro deseado de orden 2 dado por la ecuación (4.11), y se trata de obtener la misma respuesta, pero mediante un filtro de orden 1. Por tanto, lo que se pretende es ajustar los coeficientes del filtro de orden 1 para ajustar su respuesta todo lo posible a la del filtro deseado.

$$H_p[z^{-1}] = \frac{0.05 - 0.4z^{-1}}{1.0 - 1.1314z^{-1} + 0.25z^{-2}} \tag{4.11}$$

En la figura 4.7 se muestran los resultados obtenidos en forma de histogramas tanto para el ACO básico como para el modificado, así como para el algoritmo de búsqueda tabú clásico.

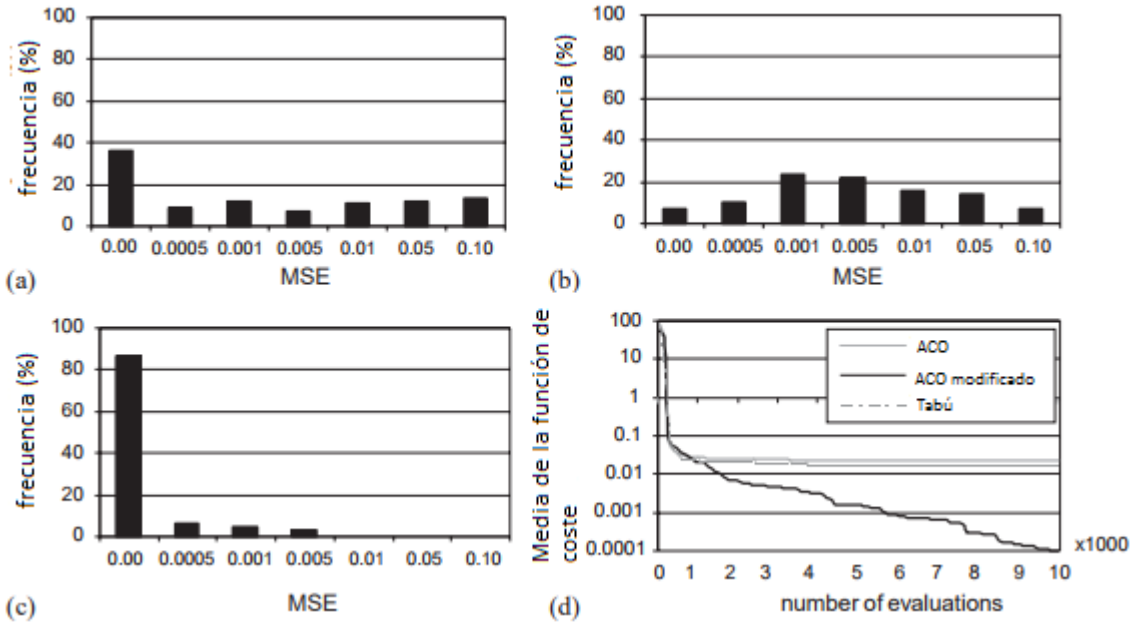

Figura 4.7. *a) ACO básico, b) TS, c) ACO modificado, d) Valor de la función de coste media para 100 ejecuciones aleatorias.*

Como conclusión se obtiene que el ACO modificado no solo obtiene mejores resultados para la mayoría de las ejecuciones, sino que tiene una probabilidad mucho menor que el resto de algoritmos de quedarse en mínimos locales.

4.5 Algoritmo de luciérnagas (FFA)

Este algoritmo está inspirado en el comportamiento de las luciérnagas, de forma que se tiene en cuenta el patrón luminiscente de las mismas, así como su movimiento. En este algoritmo el sexo de las luciérnagas no es relevante, ya que solo se tendrá en cuenta cuanto luce cada una de ellas, por tanto, aquellas luciérnagas que más lucen atraerán a las demás hacia su posición.

4.5.1 Luminiscencia

Debido a que aquellas luciérnagas que lucen mucho representan un referente para el resto, se establece que cuanto mejor es la solución que esta representa mayor brillo se le asigna. Como en la mayoría de los algoritmos evolutivos aplicados a diseño de filtros IIR, se emplea la medida del error cuadrático medio entre el filtro deseado y el obtenido para cada individuo, de forma que el brillo es inversamente proporcional al MSE.

No obstante, la atracción entre luciérnagas es relativa ya que entra en juego la distancia entre ambas además del coeficiente de absorción del medio, esto es, la atracción es inversamente proporcional a la distancia que las separa.

4.5.2 Atracción

Como la intensidad de la luz que una luciérnaga percibe es función de la distancia y del coeficiente de absorción del medio, se propone la ecuación (4.12) para modelar dicha intensidad. En ella r representa la distancia entre ambas luciérnagas, β_0 es una constante de escalado que representa el brillo de la luciérnaga para r = 0 (brillo inversamente proporcional al MSE calculado anteriormente) y γ representa el coeficiente de absorción del medio.

$$\beta = \beta_0 e^{-\gamma r^2} \tag{4.12}$$

La distancia entre dos luciérnagas se calcula como la distancia euclídea entre las mismas, es decir, como la distancia entre dos vectores con *N* dimensiones, donde *N* representa la longitud del genotipo de una luciérnaga, formado por los coeficientes del numerador y denominador del filtro concatenados.

4.5.3 Movimiento

El movimiento de la luciérnaga i-ésima se ve afectada por la atracción hacia otra con mayor intensidad luminosa según la ecuación (4.13).

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha\left(rand - \frac{1}{2}\right); \tag{4.13}$$

donde x_i y x_j son las posiciones de las dos luciérnagas, α es una constante de escalado que regula la componente aleatoria (típicamente entre 0 y 1). En la ecuación (4.13) se plantea la situación en la que la luciérnaga i-ésima atrae a la j-ésima, es decir, la primera tiene mayor intensidad luminosa, si el caso fuera el contrario, los índices i y j quedarían intercambiados en la ecuación.

4.5.4 Absorción del medio

Como se ha comentado en el apartado 4.5.1, la atracción entre dos luciérnagas se ve afectada por la absorción del medio, representado mediante la variable γ, de forma que si esta tiende a infinito la atracción entre luciérnagas prácticamente se anularía haciendo que estas se movieran de forma aleatoria. En el caso contrario, si $\gamma = 0$ se estaría en la situación de visibilidad

completa, es decir, vacío completamente despejado. De esta característica se deduce que la correcta elección del parámetro de absorción es crucial para establecer un equilibrio entre búsqueda local y global.

4.5.5 Esqueleto del algoritmo

Paso 1. Se inicializa la población de n luciérnagas escogiendo los b_k (coeficientes del numerador) y a_k (coeficientes del denominador) de forma aleatoria.

Paso 2. Se evalúan todas las luciérnagas de forma que se almacena la mejor de todas ellas, es decir, la que menos MSE haya producido como h_{gbest}.

Paso 3. Se consideran todos los pares posibles de luciérnagas, se calcula la distancia euclídea entre cada una de ellas, se calcula la intensidad luminosa de todas ellas y se aplica la ecuación (4.13) para determinar las nuevas posiciones de las luciérnagas que se han visto atraídas. Tras este paso, las luciérnagas más brillantes habrán atraído a las no tan brillantes, es decir, aquellas que representen un mayor MSE tenderán a moverse hacia aquellas con un menor MSE.

Paso 4. Se vuelve al paso 2 y se repite el ciclo hasta completar el número máximo de iteraciones, o hasta que se cumpla una condición de convergencia determinada.

Paso 5. Finalmente, el algoritmo devolverá el último valor obtenido para h_{gbest} será el filtro que menor error produce con respecto al deseado hasta el momento.

4.5.6 Aplicación a diseño de filtros IIR

Con el fin de cuantificar cuán de bueno es el algoritmo que [24] propone, se comparó con otros algoritmos evolutivos clásicos. Estos algoritmos son: PSO (Particle Swarm Optimization), DE (Diferential Evolution) y RGA (Regularized Global Aproximation).

Uno de los resultados más representativos consiste en la aproximación hacia un filtro de orden 5 mediante uno de orden 4. Es decir, el algoritmo trata de generar una respuesta lo más similar posible a la del filtro dado por la ecuación (4.14) mediante un filtro de orden 4 de la forma dada por la ecuación (4.15).

$$H_s(z) = \frac{0.1084 + 0.5419z^{-1} + 1.0837z^{-2} + 1.0837z^{-3} + 0.5419z^{-4} + 0.1084z^{-5}}{1 + 0.9853z^{-1} + 0.9738z^{-2} + 0.3864z^{-3} + 0.1112z^{-4} + 0.0113z^{-5}} \quad (4.14)$$

$$H_{af}(z) = \frac{b'_0 + b'_1 z^{-1} + b'_2 z^{-2} + b'_3 z^{-3} + b'_4 z^{-4}}{1 - a'_1 z^{-1} - a'_2 z^{-2} - a'_3 z^{-3} - a'_4 z^{-4}} \quad (4.15)$$

Por tanto, lo que se trata de conseguir es conseguir la respuesta que daría un filtro de orden 5 mediante uno de orden 4 (más simple ya que necesita de dos coeficientes menos). Los resultados numéricos del error cuadrático medio (MSE) y del tiempo de ejecución se ilustran en la tabla 4.1, tanto para el algoritmo FFA en cuestión como para el resto comentados al inicio de este apartado.

Nº Ejecución	RGA		PSO		DE		FFA	
	MSE	Tiempo	MSE	Tiempo	MSE	Tiempo	MSE	Tiempo
1	0.0356	8.907633	0.0277	4.953836	6.8820e−004	6.661808	5.6061e−006	2.907472
2	0.0507	8.166661	0.0103	4.053498	0.0014	6.529037	1.8737e−006	2.869395
3	0.0991	8.082030	0.0068	4.098620	0.0022	6.485849	5.7630e−006	2.887492
4	0.0307	8.042549	0.0177	4.115985	9.7176e−004	6.699796	4.5938e−006	2.871119
5	0.0556	8.983041	0.0035	4.987767	0.0016	6.491667	4.7569e−006	2.872375

Tabla 4.1. *Tabla comparativa entre FFA y el resto de algoritmos clásicos.*

Como se puede observar, el algoritmo que P. Upadhyay, R. Kar , D. Mandal y S.P. Ghoshal proponen en [24] es superior tanto en MSE como en tiempo de ejecución al resto de algoritmos evolutivos clásicos.

4.6 Modelo mediante membranas celulares

Este algoritmo se basa en el comportamiento de las células que componen a los seres vivos, las cuales cooperan y se comunican entre ellas mediante las membranas que las contiene y separan. Este algoritmo forma parte de los algoritmos de computación paralela o distribuida cuya principal ventaja es el equilibrio entre búsqueda global y explotación de la búsqueda local.

4.6.1 Sistemas basados en membranas

Dentro del grupo de algoritmos basados en membranas existen dos subdivisiones: los sistemas celulares y de tejidos. El primero de ellos se basa en la distribución de las células de un ser vivo separadas por membranas, las cuales están organizadas de forma jerárquica. El segundo tipo hace referencia a la colaboración entre células pertenecientes a distintos tejidos, los cuales se acercan bastante al funcionamiento de las redes neuronales artificiales, donde existen distintos niveles y donde a cada elemento se le asigna un estado [19, 20].

4.6.2 Descripción del sistema

Formalmente, un sistema de tejidos por membranas con grado q>1 viene descrito por reglas antiport/symport[19] de la forma indicada por la ecuación (4.16).

$$\Pi = (O, w_1, \ldots, w_q, R_1, \ldots, R_q, R', i_0) \tag{4.16}$$

donde:

- es un alfabeto finito con símbolos limitados llamados objetos.
- W_1, w_1,... w_q son las características iniciales de las células 1,2,...q respectivamente.
- R_1, R_2,...R_q es un conjunto finito de reglas evolutivas en las células q respectivamente.
- R' es un conjunto finito de reglas de comunicación de la forma (i,u/v,j) con $i \neq j$, i,j = 1,2,...q.
- I_0 representa la región de salida del sistema.

Un sistema de tejidos está formado por q células, cada una de ellas rodeada por una membrana celular donde todas ellas están en contacto con la región exterior llamada medio. típicamente cada célula contiene una serie de objetos w_1, w_2,... w_q .

Cada sistema de tejidos tiene dos tipos de reglas: regla de evolución y regla de comunicación. La regla de evolución tiene la forma $u \rightarrow v$, lo cual significa que *u* evoluciona hacia *v*.

Adicionalmente lado la regla de comunicación tiene la forma *(i,u/v,j)* llamada regla *antiport*, de forma que u en la célula i y v en la célula j son intercambiadas, además si *j* o *i* son 0, significa que esta comunicación se hace con el medio. Por otro lado si u o v son un objeto vacío λ, entonces las regla se llama *symport*, por ejemplo, $(i,u/\lambda,j)$, esto significa que u se comunica desde la célula i a la célula j (solo se transmite, no se realiza ningún intercambio).

4.6.3 Sistema de tejidos en forma de anillo.

Esta distribución concreta de *q* células fue introducida por [19] para el diseño de filtros IIR óptimos. El sistema consta de *q* células denominadas por 1,2,...,*q* donde el elemento 0 representa el medio que está en contacto con las membranas de todas las células del sistema, el cual además hace la función del medio de salida, es decir, el resultado final del proceso se encontrará en el medio, de forma que el medio contiene la mejor solución obtenida [19,20]. Esta configuración concreta de las células se basa en el anillo, de forma que cada célula solo se encuentra en contacto con otras dos adyacentes y el medio, y solo puede comunicarse en un solo sentido. Esta estructura se ilustra en la figura 4.8.

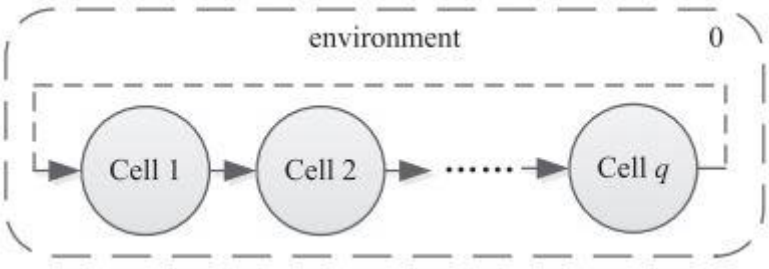

Figura 4.8. *Esquema de la distribución circular del sistema celular.*

4.6.4 Pasos del algoritmo

4.6.4.1 Objetos

Los objetos o fenotipo que determina cada una de las células viene dada por el vector $Z = (z_1, z_2, \ldots z_D)$ siendo D el número de dimensiones o variables del problema. El número de objetos de cada célula puede variar en algunos problemas concretos, pero en este caso se asume que todas ellas tienen el mismo número de objetos.

Adicionalmente cada célula memoriza la mejor solución local dada por Z_{lbest} que representa la mejor solución que cada célula tiene en cada iteración de entre el conjunto de objetos w, por otro lado, el medio almacena la mejor solución global denotada por Z_{gbest}, de forma que esta pasará a ser el resultado final cuando se llegue a un número concreto de iteraciones o se alcance un valor umbral.

4.6.4.2 Inicialización

Inicialmente se generan n objetos para cada una de las q células, los cuales forman los conjuntos de objetos $w_1, w_2, \ldots w_q$. Lo más habitual es la generación aleatoria de dichos objetos iniciales, estableciendo un umbral inferior y otro superior.

4.6.4.3 Reglas de comunicación

Estas son las reglas que establecen de qué forma se comparten o transfieren los objetos entre células adyacentes o entre las células y el medio. Cada célula transmite su mejor objeto local Z_{lbest} a la célula siguiente en el anillo, además también se contempla actualizar el valor de Z_{gbest} del medio si este mejor objeto local supera al fitness del mejor objeto global.

En este caso existen dos tipos de reglas de comunicación:

- *Regla* $(i,Z_{lbest}/\lambda,j)$ donde $j = i+1$, siempre y cuando j sea distinta de q, en ese caso se le transmite a la primera célula i=1.
- Regla $(i,Z_{libest}/\lambda,0)$, de forma que cada célula transmite su mejor objeto al medio, pero la mejor solución global solo es actualizada si $f(Z_{libest}) < f(Z_{gbest})$, es decir, si hay una mejora en la función de fitness f.

4.6.4.4 Reglas de evolución

En cuanto a las reglas de evolución, existen una gran variedad de métodos, no obstante, el más empleado es el de evolución diferencial, donde el valor del nuevo objeto depende de la diferencia entre el valor del objeto actual y el valor del mejor objeto local o global o ambos. En la ecuación (4.17) se ilustra un ejemplo de regla de evolución diferencial.

$$Y_i = Z_i + F \cdot (Z_{lbest} - Z_i) + F \cdot (Z_{ebest} - Z_i) + F \cdot \left(Z_{r_1} - Z^i_{r_2}\right) \tag{4.17}$$

donde Z_i es el objeto original de la célula *i* e Y_i es el objeto que se ha creado como donante, Z_{r1} y Z_{r2} son dos objetos elegidos aleatoriamente de entre el conjunto de la célula i, y finalmente *F* es un factor de escalado aleatorio cuyo valor se determina mediante la ecuación (4.18).

$$F = 0.5 \times (1 + rand(0, 1)) \tag{4.18}$$

4.6.4.5 Esqueleto general

Inicialmente se generan un número q de células, cada una de ellas con un conjunto de objetos aleatorios. Seguidamente se evalúan todos los objetos de cada una de ellas para determinar cuál de todos ellos es el mejor objeto local y si este actualiza al mejor objeto global (generado aleatoriamente en el inicio) que se encuentra en el medio. Después cada una de ellas se comunica mediante las reglas de comunicación, realiza las mutaciones establecidas mediante las reglas de mutación y de nuevo evalúa sus objetos como al inicio.

Este proceso se repite durante un número concreto de iteraciones preestablecidas o hasta que se cumple la condición de parada (habitualmente si se supera el umbral de fitness deseado en el mejor objeto global). Finalmente se extrae el valor del mejor objeto global hasta el momento que se encontrará en el medio.

4.6.5 Aplicación a diseño de filtros IIR

Con el fin de comparar la eficiencia del algoritmo basado en membranas (TMS), los resultados se compararon con los obtenidos para un algoritmo genético básico (GA), un algoritmo de nube

de partículas (PSO), uno basado en evolución diferencial (DE), uno basado en el enjambre de abejas (ABC) y finalmente uno de enjambre de partículas caóticas (CSO).

En [19] se realizaron 50 ejecuciones con el fin de aproximar un filtro mediante uno de un orden inferior, de forma que los resultados quedan plasmados en la tabla 4.2, dando el valor del RMSE (raíz del error cuadrático medio) entre el filtro obtenido y el de orden superior.

Método	Ejemplo 1	Ejemplo 2	Ejemplo 3	Ejemplo 4
TMS	**1.534e-2** (±**2.281e-20**)	**1.284e-3** (±**4.526e-21**)	**6.398e-3** (±**8.251e-12**)	**7.853e-5** (±**3.824e-8**)
GA	6.337e-2 (±3.294e-2)	1.295e-2 (±6.731e-2)	6.132e-2 (±3.281e-2)	1.873e-1 (±3.528e-1)
PSO	3.881e-2 (±2.019e-2)	2.521e-3 (±2.378e-19)	8.581e-3 (±3.755e-3)	5.166e-4 (±1.246e-3)
DE	2.829e-2 (±1.983e-2)	1.394e-3 (±2.516e-19)	7.725e-3 (±4.619e-4)	8.935e-5 (±2.285e-4)
ABC	2.517e-2 (±2.142e-2)	1.413e-3 (±2.615e-19)	7.832e-3 (±5.328e-4)	9.137e-5 (±2.331e-4)
CSO	1.815e-2 (±5.16e-18)	1.387e-3 (±2.428e-19)	6.716e-3 (±3.025e-11)	7.898e-5 (±2.613e-5)

Tabla 4.2. *RMSE obtenido para cada uno de los métodos bajo estudio.*

En la tabla 4.2 se ilustran 4 ejemplos distintos, donde en el ejemplo 1 se trataba de aproximar un filtro de orden uno a otro de orden dos desconocido, en el ejemplo 2 se trataba de aproximar un filtro de orden tres mediante uno de orden 2, en el ejemplo 3 se trataba de aproximar un filtro de orden tres a uno de orden 4 y finalmente en el ejemplo 4 se trataba de aproximar un filtro de orden cuatro a uno de orden cinco.

Interpretando los resultados se observa que el algoritmo TMS obtiene el menor valor de RMSE para todos los ejemplos. No obstante, tanto en el ejemplo 1 como en el ejemplo 4 los resultados son muy similares a los obtenidos con el algoritmo CSO, aunque es importante remarcar que siguen siendo mejores, por tanto, el algoritmo FFA tiene mejor convergencia que la mayoría de los algoritmos meméticos clásicos sin modificación.

4.7 Algoritmo basado en búsqueda tabú (TSA)

Como todos los algoritmos meméticos comentados a lo largo de este documento este trata de minimizar la función de coste f(x) la cual puede ser lineal o no. No obstante, lo que caracteriza a este algoritmo es la capacidad de converger rápidamente evitando soluciones exploradas anteriormente las cuales no mejoraban a las soluciones anteriores, es decir, reduce paulatinamente el área de búsqueda en función de lo ya explorado.

4.7.1 Lista tabú

En cada una de las iteraciones se almacena el valor actual de cada uno de los individuos denominados por x_{inow} donde i va desde 1 hasta q, donde q representa el número máximos de individuos de la población. Tomemos un individuo concreto x_{now}, entonces este sufrirá un pequeño cambio o movimiento, con el fin de salir del mínimo local en el que se encuentra, hacia uno de sus vecino x^* (incluso si este representa una solución peor que x_{now}).

La consecuencia de realizar este proceso repetidamente sería la entrada en un bucle de soluciones muy lejos del mínimo global que se busca, por tanto, se introduce el concepto de lista tabú, representada por *T*. Esta lista almacena todos los movimientos tabú para el individuo x_{now}, de forma que se conocen todos los movimientos que no pueden ser aplicados al objeto x_{now}. Estos movimientos almacenados en T son aquellos que se han realizado de forma más frecuente en las últimas iteraciones, todo ello en base a una serie de restricciones tabú.

El uso de la lista tabú evita entrar en un ciclo de movimientos similares ya que elimina la posibilidad de visitar de nuevo soluciones anteriores durante un número concreto de iteraciones [21].

4.7.2 Proceso de selección

Tras realizarse una serie de movimiento de acuerdo con la lista tabú, se ha generado un subgrupo de posibles soluciones denominado como Q^*, de entre las cuales se escoge aquella que disminuya la función objetivo en mayor medida para poder actualizar el valor de x_{now} con dicha solución [21].

4.7.3 Restricciones tabú

Estas restricciones establecen cuando un movimiento no debe volver a repetirse. Estas restricciones pueden generarse en función a las iteraciones, es decir, si una solución almacenada en la lista tabú ha sido visitada un cierto número de veces al largo de un número

predeterminado de iteraciones anteriores o bien a partir de la frecuencia. En este segundo caso se toma un rango más amplio de iteraciones, no obstante, no se cuenta el número de veces que se ha visitado sino la frecuencia con la que se ha hecho.

Como consecuencia de las restricciones tabú es posible que a lo largo de la ejecución se hayan restringido movimientos hacia la solución óptima o que todos los movimientos posibles se hayan clasificado como tabú, por tanto, se introduce un factor de "olvido" el cual permite al algoritmo refrescar o actualizar estas restricciones sólo en base a una serie de movimientos anteriores [21], es decir, no lista tabú no es constante y tampoco abarca desde el inicio de la ejecución.

4.7.4 Esqueleto del algoritmo de búsqueda tabú.

En la figura 4.9 se muestra el esqueleto del algoritmo de búsqueda tabú mediante un diagrama de flujo.

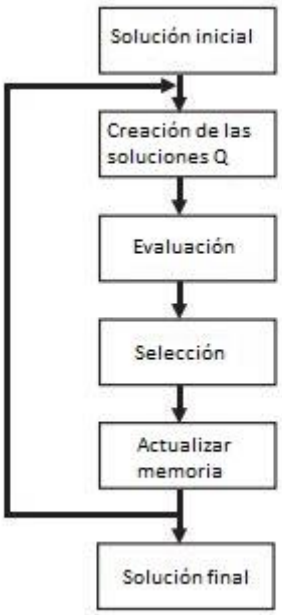

Figura 4.9. *Diagrama de flujo del algoritmo de búsqueda tabú.*

Como se muestra en la figura anterior, en el inicio se crean una serie de soluciones aleatorias, después todas ellas realizan una serie de movimientos que generan un subconjunto Q^* de soluciones, de las cuales solo la mejor la mejor solución visitada por cada uno de los individuos x_{inow} pasará a sustituir el valor actual de cada una de ellas. Como consecuencia de todos los movimientos realizados se actualiza la lista tabú y las restricciones en el paso *memory modification* [21]. Este proceso se repite hasta que se cumpla la condición de parada, en ese caso se presenta la solución final como la mejor de todos los individuos de la última iteración.

4.7.5 Aplicación a diseño de filtros IIR

En [21] se trata de evaluar la capacidad de convergencia hacia el mínimo de la función de coste (dada por el error cuadrático medio la salida de ambos filtros) del algoritmo TS mediante la aproximación de un filtro de orden dos a uno de orden tres. El filtro de tercer orden viene dado por la ecuación (4.19), de forma que se trata de conseguir la misma salida para una entrada de ruido blanco con un filtro de orden dos ajustando los coeficientes de este último.

$$H_p(z) = \frac{-0.3 + 0.4z^{-1} - 0.5z^{-2}}{1 - 1.2z^{-1} + 0.5z^{-2} - 0.1z^{-3}} \tag{4.19}$$

Por otro lado el algoritmo de búsqueda tabú propuesto en [21] se ha comparado con el algoritmo ASA (*Adaptative Simulated Annealing*), de forma que los resultados se han plasmado en una gráfica dada por la figura 4.10, que ilustra la convergencia hacia el mínimo global del MSE con un valor de 0.059.

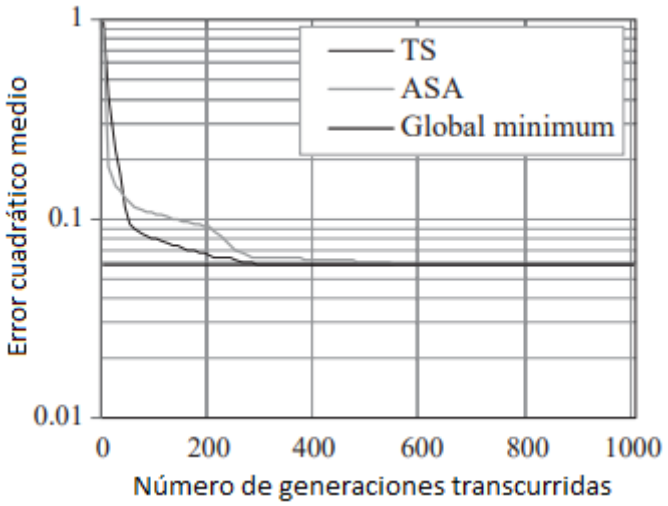

***Figura 4.10**. Evolución del MSE de los algoritmos TS y ASA a lo largo de las generaciones.*

La conclusión que se extrae es que ambos algoritmos, tanto el de búsqueda tabú como el algoritmo ASA alcanzan el mismo mínimo global, sin embargo, en base a la gráfica de la figura 4.18 se observa que el algoritmo TS converge más rápidamente (unas 200 generaciones antes que el algoritmo ASA) por tanto es más eficiente.

4.8 Algoritmo de búsqueda gravitacional híbrido (HGSA)

Este algoritmo memético para la optimización está basado en la ley de la gravedad de Newton de forma que cada uno de los objetos de la población es una masa, siendo este método híbrido una mejora del clásico GSA al cual se le ha incorporado BSA (*Backtracking Search Algortihm*). Cada objeto posee cuatro cualidades: masa gravitacional activa, masa gravitacional pasiva, inercia y posición. De entre las variables que conforman cada uno de los objetos, las tres primeras vienen determinadas por la función de fitness, mientras que la posición representa por sí misma una posible solución al problema.

4.8.1 Ley gravitacional

Cada uno de los objetos atrae a todos los demás mediante una fuerza gravitacional, la cual es directamente proporcional al producto de sus masas e inversamente proporcional a la distancia que las separa, al contrario que en la ley de gravitación, donde se emplea el cuadrado de las distancias [22]. Esta fuerza que existe entre los objetos i-ésimo y j-ésimo en la iteración t viene dada por la fórmula de la ecuación (4.20).

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N_P} \text{rand}(j)\, F_{ij}^d(t), \tag{4.20}$$

donde M_i y M_j son las masas de ambos objetos, $R_{ij}(t)$ es la distancia euclídea entre ambos en la iteración *t*, $G(t)$ es la constante de gravitación en la iteración *t* y ε es una constante de valor reducido para evitar picos excesivos en la fuerza gravitacional cuando dos objetos están muy juntos.

La constante de gravitación mencionada en el párrafo anterior tiene un valor dependiente de las iteraciones con el fin de centrar el algoritmo en la búsqueda global al inicio e ir moviéndose hacia una búsqueda más local a lo largo de la ejecución [22]. $G(t)$ viene dada por la ecuación (4.21), formada por una exponencial decreciente, cuyo exponente viene multiplicado por la constante α y dividida por *T*, que representa el número máximo de iteraciones.

$$G(t) = G_o e^{-\alpha t/T}, \tag{4.21}$$

No obstante, como en muchos algoritmos meméticos es interesante introducir un comportamiento estocástico al movimiento, por tanto, cada una de las componentes de la fuerza que actúa sobre un objeto va multiplicada por una variable aleatoria en el intervalo [0,1].

4.8.2 Ley del movimiento

La velocidad de cada uno de los objetos en cada iteración es igual a una fracción de la velocidad y aceleración que poseía en la iteración anterior. Esta aceleración es igual a la fuerza aplicada al objeto dividida entre la masa inercial que posee, por tanto, la fórmula resultante es la de la ecuación (4.22).

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)}. \tag{4.22}$$

En función de esta aceleración y a la velocidad que poseía el objeto en la iteración anterior t se calcula la posición en la iteración actual $t + 1$ según la ecuación (4.24) donde la velocidad empleada se ha obtenido mediante la ecuación (4.23).

$$v_i^d(t+1) = \text{rand}(i) \times v_i^d(t) + a_i^d(t), \tag{4.23}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1), \tag{4.24}$$

donde *rand(i)* es una variable aleatoria en el intervalo [0,1] correspondiente al objeto i-ésimo y *d* representa a la componente que se modifica.

4.8.3 Actualización de las masas

La masa de cada uno de los objetos depende tanto del fitness del propio objeto como del fitness los objetos peor y mejor de cada una de las iteraciones, además esta se normaliza dividiéndola entre suma de todas las masas de todos los objetos [22]. El mejor objeto de entre todos los existentes en una iteración es aquel de menor fitness (en caso de que se trate de un problema de minimización del mismo), por el contrario, el peor objeto será aquel con mayor fitness. Es la ecuación (4.26) la que se encarga de la actualizan de la masa del objeto i-ésimo, a partir de la ecuación (4.25).

$$m_i(t) = \frac{\text{fit}_i(t) - \text{worst}(t)}{\text{best}(t) - \text{worst}(t)}, \tag{4.25}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \quad \text{con} \quad (i = 1, 2, \ldots, N) \tag{4.26}$$

donde $fit_i(t)$ es el fitness del objeto i en la iteración t y *worst(t)* y *best(t)* son el fitness de los objetos peor y mejor respectivamente.

Con el fin de obtener un buen equilibrio entre búsqueda global y local, el número de objetos que se considera que ejercen una fuerza gravitatoria disminuye con las iteraciones [22], de

forma que al inicio se consideran todos los objetos y después solo los k mejores hasta que al final de la ejecución sólo quedará uno de ellos que será el que se tome como solución.

4.8.4 Exploración local

En cada iteración se modifica cada una de las componentes de la posición y se evalúa de nuevo para ver si se ha realizado una mejora, en ese caso se guarda la nueva posición. Si tras variar todas las componentes la nueva posición representa una mejora, se considera un éxito, en caso contrario se vuelve a la mejor posición de entre todas las obtenidas [22].

4.8.5 Esqueleto del algoritmo

Inicialmente se generan un número de objetos con posiciones aleatorias y se calcula el fitness de cada uno de ellos para asignarles una masa. En cada iteración primero se evalúan todos los objetos para obtener el mejor y peor de todos ellos y seguidamente se actualiza *G(t)* según la ecuación (4.21). Después se calcula la fuerza que actúa sobre cada uno de ellos según la ecuación (4.20) para establecer la aceleración de cada uno de ellos y sus nuevas posiciones según la ecuación (4.24). Finalmente se calcula el fitness de todos los objetos y se realiza el proceso de exploración en cada uno de ellos para comenzar la iteración siguiente.

4.8.6 Aplicación a diseño de filtros IIR

Para la evaluación de la efectividad del algoritmo HGSA, D.S.Sidhu, J.S.Dhillon y Dalvir Kaur [22] se generaron los cuatro tipos de filtros: paso bajo, paso alto, paso banda y banda eliminada. El orden de cada uno de ellos es además una de las variables de optimización del problema escogiendo un orden máximo de 12. Las frecuencias de corte y bandas de paso para cada uno de ellos quedan reflejadas en la tabla 4.3.

Filtro	Orden máximo	Banda de paso	Banda eliminada
LP	12	$0 \leq \omega \leq 0.2\pi$	$0.3\pi \leq \omega \leq \pi$
HP	12	$0.8\pi \leq \omega \leq \pi$	$0 \leq \omega \leq 0.7\pi$
BP	12	$0.4\pi \leq \omega \leq 0.6\pi$	$0 \leq \omega \leq 0.25\pi$ $0.75\pi \leq \omega \leq \pi$
BS	12	$0 \leq \omega \leq 0.25\pi$ $0.75\pi \leq \omega \leq \pi$	$0.4\pi \leq \omega \leq 0.6\pi$

Tabla 4.3. *Bandas de paso y de corte para los cuatro tipos de filtros.*

Los resultados obtenidos para el algoritmo GSA se ha comparado con los de otros algoritmos meméticos populares para el diseño de filtros IIR. En [22] se han realizado pruebas con los algoritmos: CCGA, NSGA-II, LS-MOEA, RCGA y GSA.

Técnica	Orden del filtro	Rizado en banda de paso	Rizado en banda eliminada	Error de fase
LP filter				
CCGA	3	$0.9034 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1699$	1.4749×10^{-4}
NSGA-II	3	$0.9117 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1719$	1.2662×10^{-4}
LS-MOEA	3	$0.9083 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1586$	1.0959×10^{-4}
RCGA	3	$0.9141 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1556$	1.1788×10^{-4}
GSA	3	$0.9201 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1471$	1.0391×10^{-4}
HGSA	**3**	$\mathbf{0.9208 \leq \lvert H(e^{j\omega})\rvert \leq 1.0}$	$\mathbf{\lvert H(e^{j\omega})\rvert \leq 0.1475}$	$\mathbf{1.0375 \times 10^{-4}}$
HP filter				
CCGA	3	$0.9044 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1749$	9.7746×10^{-5}
NSGA-II	3	$0.8960 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1769$	9.1419×10^{-5}
LS-MOEA	3	$0.9004 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1746$	9.6251×10^{-5}
RCGA	3	$0.9004 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1742$	9.5757×10^{-5}
GSA	3	$0.9052 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1674$	7.5431×10^{-5}
HGSA	**3**	$\mathbf{0.9051 \leq \lvert H(e^{j\omega})\rvert \leq 1.0}$	$\mathbf{\lvert H(e^{j\omega})\rvert \leq 0.1698}$	$\mathbf{7.3734 \times 10^{-5}}$
BP filter				
CCGA	4	$0.8920 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1654$	8.1751×10^{-5}
NSGA-II	4	$0.9100 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1771$	3.6503×10^{-4}
LS-MOEA	4	$0.9285 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1734$	6.0371×10^{-5}
RCGA	4	$0.9333 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1641$	5.9070×10^{-5}
GSA	4	$0.9354 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1587$	6.0121×10^{-5}
HGSA	**4**	$\mathbf{0.9398 \leq \lvert H(e^{j\omega})\rvert \leq 1.0}$	$\mathbf{\lvert H(e^{j\omega})\rvert \leq 0.1634}$	$\mathbf{5.6440 \times 10^{-5}}$
BS filter				
CCGA	4	$0.8966 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1733$	1.6198×10^{-4}
NSGA-II	4	$0.8917 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1770$	1.5190×10^{-4}
LS-MOEA	4	$0.8967 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1725$	1.5084×10^{-4}
RCGA	4	$0.8975 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1708$	1.5144×10^{-4}
GSA	4	$0.9004 \leq \lvert H(e^{j\omega})\rvert \leq 1.0$	$\lvert H(e^{j\omega})\rvert \leq 0.1481$	1.4103×10^{-4}
HGSA	**4**	$\mathbf{0.9009 \leq \lvert H(e^{j\omega})\rvert \leq 1.0}$	$\mathbf{\lvert H(e^{j\omega})\rvert \leq 0.1459}$	$\mathbf{1.3827 \times 10^{-4}}$

Tabla 4.4. *Resultados obtenidos en el diseño de los cuatro tipos de filtros para los seis algoritmos.*

Como se puede observar en la tabla 4.4, los mejores resultados obtenidos se corresponden con el algoritmo HGSA, superando incluso al GSA clásico, por tanto, la introducción de BSA es un acierto. Por otro lado, esta mejora también reduce el número de parámetros (el algoritmo BSA se controla con un solo parámetro), facilitando la configuración.

Finalmente se llega a la conclusión de que HGSA es el más eficiente tanto en los rizados en la banda de paso y eliminada como en el error de fase obtenido, además es mucho más robusto ya que según [22] la desviación estándar es menor que para el resto de los algoritmos, disminuyendo la probabilidad de obtener un resultado no óptimo.

5. Método de diseño propuesto

5.1 Introducción

El método que se ha implementado para el diseño de filtros IIR está basado en el algoritmo de salto de ranas barajadas o SFLA, no obstante, se han introducido bastantes modificaciones con el fin de mejorar su rendimiento además de adaptarlo al problema que nos ocupa. Adicionalmente se han introducido bastantes mecanismos a mayores que añaden bastantes más parámetros configurables que permiten ajustar los resultados y velocidad del algoritmo.

5.2 Generación inicial de la población

Al igual que en todos los algoritmos bioinspirados se necesita de una población inicial, la cual se genera de forma aleatoria. No obstante, existen problemas en los que a priori se conocen ciertas cotas o características de las soluciones, así los individuos iniciales pueden ser generados con ciertas restricciones para acelerar la convergencia y dirigir la búsqueda global inicial.

En el caso del diseño de filtros IIR inicialmente no se consideraron restricciones con respecto al módulo o fase de los ceros y polos, por tanto, cada vez que se genera una rana aleatoria se ha de comprobar si el filtro es estable, es decir, si todos sus polos están dentro de la circunferencia unidad. Estos coeficientes además siempre tomarás un valor dentro del intervalo [-1,1] mediante una distribución aleatoria uniforme, con el fin de mantener la normalización.

La generación completamente aleatoria de polos y ceros para filtros de orden bajo no resulta un problema ya que, al haber un número reducido de polos, la probabilidad de que todos ellos estén dentro de la circunferencia unidad es reducida pero no lo suficiente como para que el algoritmo quede ralentizado en exceso. No obstante, de forma empírica se ha comprobado que para filtros de orden superior a 15, el tiempo que se tarda en generar cada uno de los filtros aleatorios estables es excesivamente grande, ya que la probabilidad de que todos los polos tengan un módulo menor que uno es ínfima.

Con el fin de evitar este obstáculo se ha optado por una generación “guiada”, de forma que en lugar de generar los coeficientes directamente como un valor aleatorio en el intervalo [-1,1], se generan tantos polos como orden tenga el filtro. Para asegurar la estabilidad se genera inicialmente el módulo de los mismo siempre en el intervalo [0,1), por tanto, siempre estarán en el interior de la circunferencia unidad. Seguidamente se determina la fase de cada uno de ellos en el intervalo $[0,2\pi]$ también mediante una distribución aleatoria uniforme.

Finalmente se ha considerado también establecer cotas a la fase de los polos y ceros con el fin de generar filtros aleatorios con un fitness mayor en lugar de emplear el intervalo [0,2π]. En la respuesta en frecuencia de un filtro IIR los polos representan máximos, la magnitud de este máximo viene dada por el módulo del mismo, cuanto más cercano a uno mayor será la ganancia, por otro lado, la posición del mismo vendrá dada por su fase. Contrariamente, los ceros representan mínimos cuyo módulo determina la atenuación, cuanto más cercano está este módulo a 1, mayor es la atenuación a la frecuencia especificada por su fase.

Consecuentemente, si se piensa en la respuesta en frecuencia de un filtro, los polos estarán localizados en la banda de paso y los ceros en la banda eliminada, por tanto, se modifica el rango de la fase de polos y ceros acorde con la tabla 5.1.

Tipo de filtro	Fase de los polos	Fase de los ceros
Paso bajo	[0 , Wc]	[Wc , 2π]
Paso alto	[Wc, 2π]	[0 , Wc]
Paso banda	[Wc_1 , Wc_2]	[0 , Wc_1] U [Wc_2 , 0]
Banda eliminada	[0 , Wc_1] U [Wc_2 , 0]	[Wc_1 , Wc_2]

Tabla 5.1. *Intervalos de generación de la fase de polos y ceros en el algoritmo propuesto.*

Donde Wc_1 y Wc_2 representan las frecuencias de corte inferior y superior en el caso de los filtros paso banda y banda eliminada y U representa la unión de ambos intervalos.

En el diseño de filtro paso bajo se emplean las ecuaciones (5.1) y (5.2) para determinar la fase de los ceros y los polos respectivamente. De forma análoga se emplean las ecuaciones (5.3) y (5.4) para filtro paso alto.

$$fasecero = rand\,(Wc * \pi, \pi) \tag{5.1}$$

$$fasepolo = rand\,(0, Wc * \pi) \tag{5.2}$$

$$fasecero = rand\,(0, \pi * Wc) \tag{5.3}$$

$$fasepolo = rand\,(Wc * \pi, \pi) \tag{5.4}$$

Para el diseño de un filtro paso banda se emplean las ecuaciones (5.5) y (5.6), que determinan la fase de los ceros y polos respectivamente, donde Wc_1 representa la frecuencia de corte inferior y Wc_2 representa la frecuencia de corte superior. En el caso de diseñarse un filtro de banda eliminada se emplearían las ecuaciones (5.7) y (5.8) de forma análoga al caso de filtro paso banda

$$fasecero = rand\,(-\pi * Wc_2\,, \pi * Wc_1) \qquad (5.5)$$

$$fasepolo = rand\,(\pi * Wc_1\,, 2\pi * Wc_2) \qquad (5.6)$$

$$fasecero = rand\,(\pi * Wc_1\,, 2\pi * Wc_2) \qquad (5.7)$$

$$fasepolo = rand\,(-\pi * Wc_2\,, \pi * Wc_2) \qquad (5.8)$$

Tras determinar la fase de polos y ceros, denotadas por *fasepolo* y *fasecero* respectivamente, siendo *rand(a, b)* un número aleatorio según una distribución uniforme entre a y b, se emplean las ecuaciones (5.9) y (5.10) para calcular los polos y ceros con módulo igual o menor a uno.

$$Polo = rand(0{,}1) * [cos(\,fasepolo) + j * sen(\,fasepolo\,)] \qquad (5.9)$$

$$Cero = rand(0{,}1) * [cos(\,fasecero) + j * sen(\,fasepolo)] \qquad (5.10)$$

Con respecto a las ecuaciones desde la (5.1) a la (5.8), solo se ha considerado el intervalo [0, π], ya que al tratarse de filtros con coeficientes reales, todos los polos y ceros deberán estar agrupados en pares conjugados.

Establecer intervalos variables para la fase de polos y ceros dependiendo del tipo de filtro que se desea diseñar acelera la convergencia inicial ya que la convergencia de partida siempre será mayor que en el caso de generar ceros y polos con fase aleatoria. Se deduce de todo ello que un filtro con polos en la banda de paso y ceros en la banda eliminada siempre tendrá más aptitud que un filtro con los polos y ceros posicionados aleatoriamente.

5.3 Representación de las soluciones

Debido a la analogía con la charca de ranas, cada una de ellas se encuentra en una posición determinada por una serie de coordenadas, las cuales vienen dadas por los coeficientes del filtro IIR que representa cada una de ellas. el vector que determina la posición está formado por la concatenación de los coeficientes del numerador y los del denominador como se muestra en la ecuación (5.11).

$$v = \{b_0, b_1, \ldots, b_n, 1, a_1, a_2, \ldots, a_n\} \qquad (5.11)$$

5.4 Cálculo del fitness

Con el fin de poder evaluar cuán de apta es cada una de las ranas que conforman la población para luego poder establecer cuales guiarán a las demás, se ha evaluado cuanto se aproxima el filtro generado por los coeficientes que conforman la posición de la rana al filtro ideal objetivo.

El filtro objetivo en cada caso es un filtro ideal no realizable cuyo valor en la banda de paso es uno (expresado en decibelios su respuesta en la banda de paso sería de 0 dB, es decir, ganancia unitaria) y cuyo valor en la banda eliminada es cero. La respuesta tanto del filtro ideal como la del filtro que representa cada rana se encuentra en unidades naturales ya que al comparar la banda de paso eliminada de ambos filtros surge el problema de que el valor cero en la banda eliminada en el filtro ideal equivale a $-\infty$ dB.

La fórmula para calcular el fitness del filtro obtenido tanto en la banda de paso como en la banda eliminada es la ilustrada en la ecuación (5.12), donde *J(w)* representa el error cuadrático medio en la banda que se evalúa, la cual se calcula como el sumatorio del error al cuadrado entre el número de muestras (expresión indicada en la ecuación (5.13)).

$$fitness = \frac{1}{1+J(w)} \tag{5.12}$$

$$MSE = \frac{1}{n} * \sum_{w=0}^{n-1}[h_i(w) - h(w)]^2 \tag{5.13}$$

donde MSE representa el error cuadrático medio (mean square error en inglés), n es el número de muestras comparadas, *hi(w)* y *h(w)* representan la respuesta en frecuencia del filtro ideal y obtenido respectivamente. La fórmula (5.12) se ha usado en varios artículos académicos como en [19] y [21] .

5.5 Generación de los memeplexes

Una de las bases de SFLA es la división de la población en memeplexes que evolucionan de forma independiente, aunque cada ciertas iteraciones se realice un barajado o mezclado. Por tanto, una de las cuestiones que se plantean es el método de agrupamiento de dicha población con el fin de obtener la mayor independencia y variedad posibles. La idea de agrupamiento que se emplea en este método se basa en dos aspectos: que los centros de los agrupamientos estén distribuidos uniformemente en el plano de funciones objetivo y que aquellos individuos más cercanos en el plano de funciones objetivo se encuentren en el mismo grupo o memeplexe.

El primer motivo por el que se realiza un agrupamiento en memeplexes de esta forma concreta es generar grupos lo más dispares posibles con el fin de explorar todo el espacio de soluciones. El segundo motivo, es que como se comenta en el artículo [23], agrupar las ranas con un fitness similar hace que cada memeplexe evolucione independientemente de una forma más eficiente, ya que se evita la aleatoriedad del fitness de las ranas de cada memeplexe.

El proceso de conformado de los memeplexes se ilustra en el siguiente fragmento de pseudocódigo:

1 **Entrada:** parámetros *P, m, n, k, etc*
2 **A: Generar el plano acorde con los valores de funciones objetivo**
3 **Para** *i = 1, 2, ..., k* **hacer**
4 Determinar los valores máximo y mínimo de las funciones objetivo para calcular los rangos del plano.
5 Dividir el plano de funciones objetivo en 2**m* secciones
6 **Fin**
7 **B: Generar m centros para los memeplexes**
8 Se generan m puntos equidistantes a lo largo de la diagonal que va desde el valor máximo de la funcion de fitness para la banda de paso hasta el valor máximo de la función de fitness para la banda eliminada.
9 **Para** *j = 1, 2, ..., m-1, m* **hacer**
10 $C_{cluster\_j} = [\, L_2 + 2(j-1)L_2,\ L_3 + 2(j-1)L_3,\ \ldots,\ L_k + 2(j-1)L_k,\ (2m-1)L_1 - 2(j-1)L_1 \,]$
11 **Fin**
12 **C: Agrupamiento de todas las soluciones, generar los memeplexes.**
13 Para cada centro se analiza la distancia a cada una de los individuos
14 **Para** *i = 1, 2, ..., n-1, n* **hacer**
15 **Para** *j = 1, 2, ..., m-1, m* **hacer**
16 **s = argmin d($C_{cluster\_j}$, F(p))**
17 Añadir *s* al memeplexe *j*
18 Eliminar *s* de *P*
19 **Fin**
20 **Fin**
21 **Devolver:** *m* memeplexes

Figura 5.1*. Pseudocódigo que ilustra el proceso de generación de los memeplexes*

Inicialmente se establecen las variables, donde P representa la población de ranas, m es el número de memeplexes, n es el número de ranas en cada uno de los memeplexes, k es el número de objetivos, $C_{cluster\_j}$ representa las coordenadas del centro del agrupamiento j en el espacio de funciones objetivo, *F(p)* es el vector objetivo de la rana p y $d(C_{cluster\_j}, F(p))$ representa la distancia euclídea entre $C_{cluster\_j}$ y *F(p)*.

En la línea 10 del código $Li\ (\ i=1, 2, \ldots, k)$ es la longitud del objetivo *i* (en esta aplicación particular del algoritmo se emplea un valor de *k=2*, ya que solo tenemos dos funciones objetivo, la función de fitness correspondiente a la banda de paso y la correspondiente a la banda eliminada) de forma que los $C_{cluster\_j}$ *(j=1, 2, ..., m)* estarán distribuidos a lo largo de la diagonal del plano de funciones objetivo, esto conlleva que cada punto del espacio tiene igual probabilidad de ser explorado. Esta localización concreta de los centros de los memeplexes está basada en la premisa de generar memeplexes variados, ya que si los centros se encontrasen en la bisectriz del plano se generaría un memeplexe con las peores ranas (no evolucionarían hacia una buena solución ya que todas ellas son las peores de entre la población) y otro con las

mejores (estas evolucionarían rápidamente hacia un máximo local de ambas funciones de fitness ya que hay ausencia de diversidad).

En el proceso se establecen los valores máximos y mínimos de las funciones objetivo entonces los centros de los memeplexes quedan distribuidos uniformemente a lo largo de la recta que une estos valores (pasos del 7 al 11). La generación de los m memeplexes comienza en el paso 12, de forma que se escoge a cada una de las ranas de la población y se evalúa su distancia hasta cada uno de los centros de los memeplexes generados anteriormente, por tanto, esta rana queda asignada al memeplexe cuyo centro se halle más cerca de la misma en el plano de funciones objetivo, finalmente la rana se elimina del conjunto P. Este proceso se realiza de forma iterativa hasta que el conjunto P no contenga ningún elemento.

En la siguiente gráfica se ilustra un ejemplo donde se consideran dos funciones objetivo (k=2), 12 ranas (*P* contiene 12 individuos), 4 memeplexes (*m=4* y por tanto *n=3*). Además, se marca con una línea verde el frente Pareto, siendo el eje de ordenadas representa los valores para la función de fitness correspondiente a la banda de paso mientras que el eje de abscisas representa los valores para función de fitness para la banda eliminada. Estos ejes siempre tendrán como valor máximo uno, correspondiente al fitness del filtro ideal.

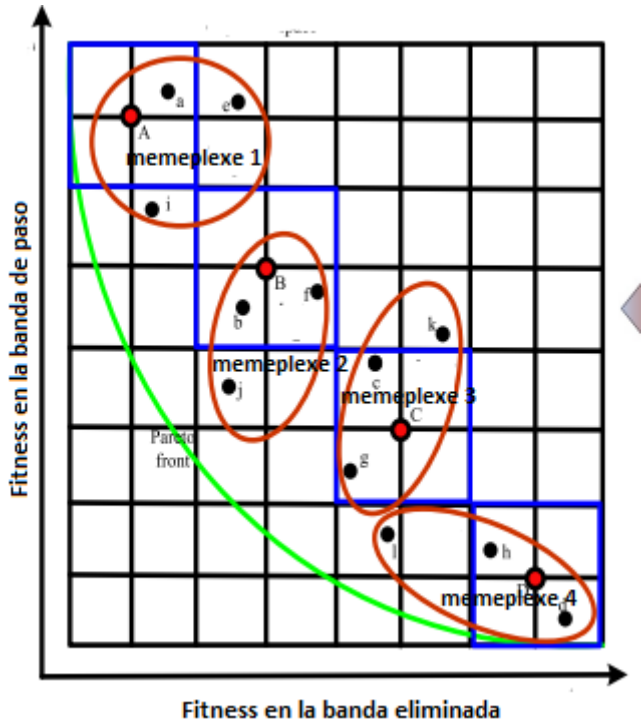

***Figura 5.2**. Representación de los clusters generados*

5.6 Adaptación del algoritmo SFLA a problemas multiobjetivo

Una de las características que hace más complejo a este algoritmo es la existencia de más de una función objetivo, en nuestro caso particular solo tenemos dos: la función de fitness para la banda de paso y a la función de fitness para la banda eliminada. La complejidad radica en la dificultad de comparar las ranas con el fin de determinar cuáles son mejores o peores ya que no basta con ordenar el fitness de cada una de ellas por el hecho de que todas ellas poseen dos funciones a optimizar.

Desde el punto de vista teórico. cómo se comenta en el artículo [23], la forma más adecuada de cuantificar cuán de buena es una rana es compararla con el frente Pareto en el plano de funciones objetivo. El frente Pareto representa los puntos del plano de funciones objetivo donde al mejorar una de ellas no empeora la otra. Debido a que la determinación del frente Pareto para este algoritmo aplicado a diseño de filtros IIR no es ni mucho menos trivial, se emplea otro método para evaluar las distintas ranas.

Como método de evaluación se emplea la distancia euclídea desde cada una de las ranas en el plano de funciones objetivo al centro de coordenadas (origen de coordenadas del plano desarrollado en el apartado 5.5), por tanto, cuanto más alejada esté una rana del origen de coordenadas (distancia euclídea mayor) mejores características poseerán. Esta distancia se calcula mediante la ecuación (5.14).

$$distancia = \sqrt{(origen_1 - fobjetivo1)^2 + (origen_2 - fobjetivo2)^2} \quad (5.14)$$

donde $origen_1$ y $origen_2$ representan el mínimo del eje de abscisas y ordenadas respectivamente y $fobjetivo_1$ y $fobjetivo_2$ representan las coordenadas en el eje de abscisas y ordenadas respectivamente de la rana que se está evaluando.

Uno de los problemas que surgen en la clasificación es que puede ocurrir que más de una rana se encuentre a la misma distancia del centro de coordenadas, en ese caso se emplea el mecanismo desarrollado en el apartado 5.7, de forma que se elige aquella que favorezca la diversidad.

5.7 Determinación del salto

En cada una de las iteraciones saltarán N ranas, de forma que cada una de ellas realizará un movimiento hacia una rana con mejor posición. El primer salto se realiza en función de la mejor rana del memeplexe, denotada por x_b en el que se encuentra la peor rana x_w siguiendo la ecuación 5.15, donde C es un factor de escalado que disminuye de forma lineal a lo largo de las iteraciones, desde el valor máximo *escalado_max* hasta el valor mínimo *escalado_max.*

Este sistema de variación de la constante de escalado hacia valores menos se menciona en [23] y tiene la finalidad de ajustar el rango de búsqueda de cada uno de los individuos. Por tanto, en el inicio los saltos realizados son grandes, centrando la búsqueda en el ámbito global, y a medida que avanza la ejecución los saltos se vuelven más pequeños basando la búsqueda en el ámbito local.

$$x'_w = x_w + C * rand(0,1)[x_b - x_w] \quad (5.15)$$

Esta nueva rana x'_w se evalúa y se compara con la rana antes del salto, si se ha producido alguna mejora esta queda sustituida por la nueva, en caso contrario, la nueva posición se calcula en función de la mejor rana global denotada por x_g según la ecuación (5.16).

$$x'_w = x_w + C * rand(0,1)[x_g - x_w] \quad (5.16)$$

5.8 Mutación

En cada iteración cabe la posibilidad de que la peor rana de cada memeplexe tenga que ser mutada con el fin de que esta experimente una pequeña mejora, en caso de que esta no haya saltado a una mejor posición siguiendo las ecuaciones (5.15) y (5.16).

Inicialmente se escoge una componente del vector que conforma la posición de la rana de forma aleatoria y se realiza un pequeña modificación aleatoria sobre la misma siguiendo la ecuación (5.17), seguidamente se evalúa la nueva rana, que denominaremos por x'_w. Si esta nueva rana modificada experimenta alguna mejora con respecto a la x_w original entonces queda sustituida, en caso contrario se escoge otra componente aleatoria y se muta de nuevo. Este ciclo se repite tantas veces como componentes tenga el vector, en nuestro caso el doble del orden del filtro más uno (ya que el denominador comienza con el valor uno y este no debe ser cambiado para mantener la normalización).

$$x'_w = x_w \pm rand(0,1) * \gamma \quad (5.17)$$

Donde γ representa un factor de escalado que disminuye linealmente a lo largo de las iteraciones desde su valor máximo *gamma_max* hasta su valor mínimo *gamma_min*, de esta forma las modificaciones son cada vez menores centrando la búsqueda en el ámbito local.

5.9 Mejora de la convergencia

Uno de los problemas más comunes de los algoritmos meméticos es el estancamiento en óptimos locales por falta de diversidad por tanto en el algoritmo propuesto se ha introducido un mecanismo para evitar la monotonía en la convergencia. Este sistema consiste en comprobar en cada una de las iteraciones si el valor actual del fitness lleva repitiéndose durante un número concreto de generaciones, en caso de que esto ocurra se activa el mecanismo de mutación empleado para las peores ranas con probabilidad P_{ab} pero en este caso para la mejor rana de cada memeplexe y con probabilidad 1. Esto hace que el algoritmo siga convergiendo hacia el óptimo global ya que se fuerza la mejora de las x_{best} haciendo que el algoritmo salga del óptimo local en el que ha quedado estancado.

Adicionalmente se ha introducido un sistema para evitar la falta de diversidad en las soluciones, dando prioridad a aquellos individuos mejor distribuidos en el plano de funciones objetivo, favoreciendo así un convergencia más estable y no prematura. Para medir la diversidad se han empleado las características de la entropía de Shannon, de forma que esta se calcula sobre el vector de distancias de cada rana al resto, para evaluar si se encuentran uniformemente distribuidas o por el contrario se encuentran la mayoría a distancias muy similares. La ecuación (5.18) es la empleada para calculas la entropía de Shannon. El fin de esta ecuación es evaluar que probabilidad de aparición tiene cada uno de los valores dentro del vector de distancias. Una entropía de reducida implica una gran variedad de valores para las distancias, y por tanto la rana sobre la que se ha calculado estará "esparcida" con respecto al resto, característica deseable si lo que se pretende es mantener la diversidad.

$$H(X) = -\sum_{i} p(x_i) \log_2 p(x_i) \tag{5.18}$$

Donde X representa el vector de distancias, $H(X)$ es el valor de la entropía del vector X y x_i es el elemento i-ésimo del vector X.

5.10 Pseudocódigo del algoritmo propuesto

El pseudocódigo que representa el esqueleto del algoritmo completo queda reflejado en la figura 5.3.

```
1   Entrada: Parámetros de configuración
2   Generar la población P con F ranas de forma aleatoria
3   Evaluar el fitness de cada una de las ranas
4   Para s < Numero máximo de iteraciones
5       Construir los m memeplexes a partir del plano de fitness
6       Para k = 1 hasta N hacer
7           Seleccionar x_b y x_w para cada memeplexe además de x_g
8           Para j = 1 hasta m hacer
9               Actualizar x_w y generar x'_w usando la ecucación 5.15 en el
                memeplexe j
10              Si x'_w mejor que x_w
11                  x_w = x'_w, continuar
12              Actualizar x_w y generar x'_w usando la ecucación 5.16 en el
                memeplexe j
13              Si x'_w mejor que x_w
14                  x_w = x'_w, continuar
15              r = random(0,1)
16              Si r < P_ab
17                  Proceso de mutación sobre x_w
18              si no
19                  x_w se sustituye por una rana aleatoria
20          Fin
21      Fin
22      Generar de nuevo los memeplexes
23  Fin
```

Figura 5.3. *Pseudocódigo del esqueleto del algoritmo completo*

Como se observa en el pseudocódigo de la figura 5.3, inicialmente se generan las ranas de forma aleatoria, pero con las restricciones ya comentadas en el apartado 5.2, seguidamente se calcula el valor de las dos funciones de fitness para cada una de ellas y se construyen los memeplexes con el método del plano explicado en el apartado 5.5.

Después se escoge un valor para N que determina el número de saltos que se realizarán en cada una de las iteraciones, entonces se extrae la mejor y peor rana de cada uno de los memeplexes, entonces x_w realiza un primer salgo según la ecuación (5.15), si no se observa mejora, se realiza un nuevo salto en base a la ecuación (5.16). Si finalmente sigue sin observarse mejora, se genera un número aleatorio r en el intervalo [0,1] como se puede ver en la línea 15, si este es

mejor que la constante P_{ab} (al igual que r esta se encuentra entre cero y uno) la rana se muta según el proceso indicado en el apartado 5.8.

En caso de que r sea mayor que P_{ab}, esta rana queda sustituida por una aleatoria y se sigue el bucle hacia el siguiente memeplexe. Cuando ya se han realizado todos los saltos en todos los memeplexes, las ranas se barajan generando de nuevo el plano y redistribuyéndolas en los nuevos memeplexes. El bucle finaliza cuando se han llevado a cada un número máximo de iteraciones.

6. Resultados numéricos

6.1 Introducción

En este capítulo se va a analizar el funcionamiento del algoritmo propuesto en el capítulo 5, el cual se ha implementado en el programa Matlab R2015a. Inicialmente se realizará un ajuste de los parámetros propios del algoritmo para establecer sus valores óptimos atendiendo tanto fitness del filtro obtenido como a la carga computacional de la ejecución del programa (medida mediante el tiempo que se tarda en obtener el resultado final).

La medida del fitness del filtro obtenido en cada una de las ejecuciones se hará mediante la media entre el fitness resultante para la banda de paso y el fitness resultante para la banda eliminada (ambos calculados mediante la ecuación (5.12)) ya que no se hace ningún tipo de ponderación entre ellas, dándoles el mismo peso a ambas. El filtro obtenido para cada simulación será un filtro paso bajo con frecuencia de corte 0.25 (el objetivo será un filtro no realizable con frecuencia de corte en 0.25).

Para la calibración de los parámetros del algoritmo se variarán:

- El número de saltos por iteración.
- El número de memeplexes.

Adicionalmente, en la tabla 6.1 se muestran los parámetros que se mantendrán constantes además de los valores de los mismos.

Parámetro	**Valor**
Número de iteraciones	500
Escalado máximo del salto	1
Escalado mínimo del salto	0.4
Gamma máxima	1
Gamma mínima	0.0001
Tipo de filtro	Paso bajo
Orden del filtro	10
Frecuencia de corte normalizada	0.25

Tabla 6.1. *Parámetros que se mantendrán constantes junto con los valores empleados.*

Después se comparará el filtro obtenido mediante el algoritmo implementado y un filtro elíptico digital basado en su prototipo analógico mediante la transformación de la bilineal Z para distintos órdenes del mismo. Esta comparación se hará mediante el cálculo de la banda de transición para ambos filtros estableciendo unas especificaciones concretas para los rizados de la banda de paso y eliminada, además se calculará el fitness para ambos. Se ha considerado que el filtro elíptico es el más adecuado para realizar la comparación ya que posee rizado en ambas bandas y de entre los modelos clásicos es el que menor orden necesita para cumplir las mismas especificaciones.

Además, se medirá la capacidad del algoritmo de generar un filtro lo más cercano posible a uno de orden superior como se realiza en la mayoría de los estudios relacionados con el diseño de filtros IIR mediante algoritmos bio-inspirados [21, 22, 23, 24, 25], en este caso se ha tratado de generar la respuesta de un filtro paso bajo de orden 5 a partir de uno de orden 4 mediante el MA propuesto.

Finalmente se comprobará si el MA es capaz de generar otros tipos de filtros como paso alto, paso banda y banda eliminada, por tanto, se seguirá el proceso análogo al caso del filtro paso bajo, variando el orden del filtro generado e imponiendo las mismas especificaciones para el caso de un filtro elíptico con el fin de comparar el fitness correspondiente a ambos.

6.2 Ajuste de parámetros

Inicialmente se plantea la variación del parámetro N, que representa el número de saltos que se realizan en cada memeplexe en cada iteración. Es importante mencionar que un aumento en N supone un aumento en el tiempo de ejecución ya que es necesario reevaluar todas las ranas de cada memeplexe cada vez que se realiza un salto, especialmente recalcular el fitness de toda la población. De entre todas las funciones que posee el programa es esta la que más operaciones requiere por el hecho de que para rana debe calcularse su respuesta en frecuencia en base a los coeficientes de la misma. En la tabla 6.2 se refleja el fitness final obtenido con respecto a cada valor de N, manteniendo constantes las variables reflejadas en la tabla 6.1 además de una población de 40 individuos en 5 memeplexes.

Valor de N	Fitness total	Tiempo de ejecución
1	0.9900	132,0264 segundos
2	0.9909	195,1348 segundos

3	0.9906	235,3673 segundos
4	0.9901	316,6139 segundos
5	**0.9924**	**350,8479 segundos**
6	0.9920	432,3992 segundos
7	0.9921	532,9493 segundos

Tabla 6.2. *Valor del fitness del filtro obtenido y el tiempo de ejecución para cada valor de N.*

En base a los resultados presentados en la tabla 6.2 se puede afirmar que los mejores resultados se obtienen para un número de saltos por iteración elevado, no obstante, si se tiene en cuenta la carga computacional, el número óptimo de saltos por memeplexe e iteración es 5 para el caso de 40 ranas distribuidas en 5 memeplexes ya que el valor final del fitness no aumenta más. El filtro obtenido en este caso se ilustra en la figura 6.1, indicando además los valores del rizado en la banda de paso (1-δp = 1dB), rizado la banda eliminada (δs = 25dB) y el ancho de banda resultante (Δw=0.12 en frecuencias normalizadas).

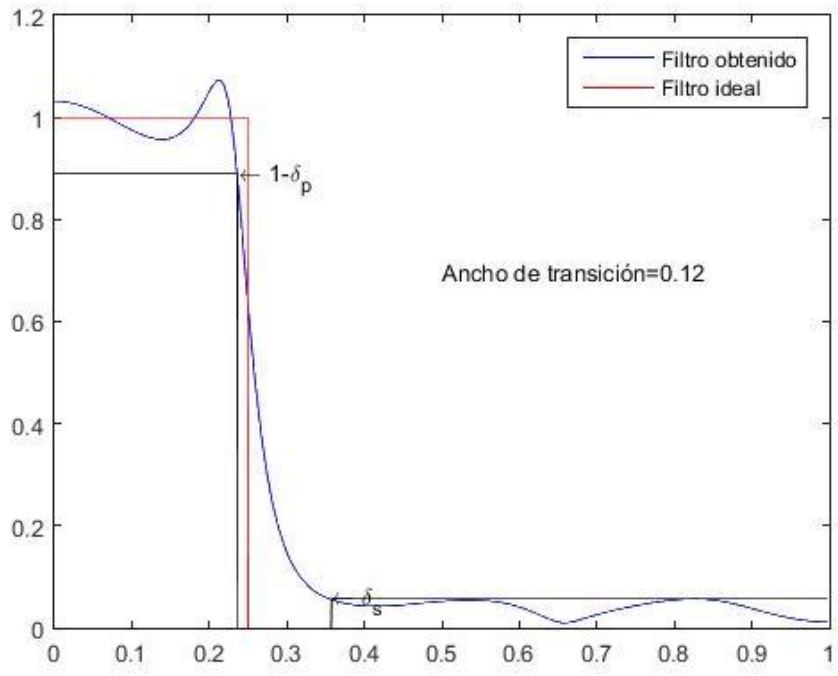

Figura 6.1*. Filtro obtenido con N=5, donde δs = 25dB y 1-δp = 1dB junto con el filtro ideal al que se trata de aproximar.*

Por tanto, si se desea escoger el número óptimo de saltos debe partirse de N=m, donde m es el número de memeplexes, e ir reduciendo N hasta encontrar el valor para el que el fitness final empieza a disminuir, en este caso N=5.

Seguidamente se va a analizar la influencia del número de memeplexes en el valor del fitness final, estableciendo los valores de los parámetros como se indica en la tabla 6.1 y un valor de N cercano a n debido a las conclusiones obtenidas anteriormente. Los resultados obtenidos se presentan en la tabla 6.3, variando el valor de m de forma que el número n sea entero.

Número de memeplexes *m*	***N***	**Número de ranas por memeplexe *n***	**Fitness obtenido**	**Tiempo de la ejecución**
2	17	20	0,9903	1886,354 segundos
4	8	10	0,9908	579,297 segundos
5	6	8	0.9915	446,829 segundos
10	3	4	0.9920	577,981 segundos

Tabla 6.3. *Fitness y tiempo de ejecución obtenidos en función del número de memeplexes y de N.*

En base a los resultados obtenidos reflejados en la tabla 6.3, se ha llegado a conclusión de que el filtro con mayor fitness final se ha obtenido para m=10. El motivo de estos resultados es el equilibrio entre diversidad e intercambio, es decir, si el número de memeplexes escogido es bajo se tiene una gran variedad de ranas por cada uno de ellos, no obstante, el barajado que se realiza tras los N*m saltos no tiene el efecto deseado ya que solo existen dos grupos que evolucionan independientemente, entonces se intercambia escasa información entre ellos.

Por tanto es en aquellas configuraciones donde hay más memeplexes en las que mejores resultados se obtienen debido al intercambio de información entre ellos, de esta forma se consolida la base de los algoritmos meméticos y en concreto del algoritmo SFLA, que es el intercambio de información entre individuos independientes.

La conclusión obtenida es que el número de memeplexes generados m debe ser suficiente como para ampliar la búsqueda a todo el espacio de soluciones y tener en cuenta varias líneas de evolución (cada una correspondiente a un memeplexe distinto).

6.3 Filtro paso bajo

6.3.1 Comparación con IIR Elíptico obtenido con BZT

Para analizar los resultados obtenidos mediante el algoritmo memético empleado se ha variado el orden del filtro a diseñar y se ha calculado el fitness tanto de este filtro como el de uno elíptico mediante la transformada bilineal Z con las mismas especificaciones de rizado en la banda de paso y en la banda eliminada y misma frecuencia de paso Wp. Estos rizados y frecuencia de paso los establece el filtro resultante de la ejecución del algoritmo memético, seguidamente se emplea la función de matlab de la ecuación (6.1) para generar el elíptico, donde orden es el orden del filtro, *Rp* es el rizado en la banda de paso en dB, *Rs* es el rizado en la banda eliminada y Wp es la frecuencia límite de la banda de paso. El orden del filtro elíptico *n_ellip* se calculará mediante la función de matlab presentada en la ecuación (6.2), donde los parámetros son análogos a la ecuación (6.1) excepto *Ws* que representa la frecuencia de corte calculada.

$$[b,a] = \text{ellip}(\text{orden}, Rp, Rs, Wp) \tag{6.1}$$

$$[n\_ellip, Wp] = \text{allipord}((Wp,Ws,Rp,Rs) \tag{6.2}$$

Los resultados del fitness para el filtro generado mediante el algoritmo memético y el elíptico se muestran en la tabla 6.4, todos ellos para un filtro paso bajo con frecuencia de corte normalizada de 0.25, además se muestran los rizados en banda de paso y eliminada en dB y la frecuencia límite de la banda de paso *Wp*. Por otro lado, también se ha calculado el ancho de banda de transición denominado por $\Delta\omega$ reflejando el valor para cada uno de los filtros en función del orden en la tabla 6.5.

Orden con MA	**Orden elíptico**	**1 - δp**	**δs**	**Wp**	**Fitness con MA**	**Fitness elíptico**
5	2	0.708	14.095	0.214	0.9783	0.9750
10	3	0.536	21.072	0.233	0.9918	0.9904
15	4	0.73	22.2	0.244	0.9958	0.9931
20	5	0.872	29,02	0.246	0.9971	0.9947
25	4	0.805	23.267	0.248	0.9979	0.9929

Tabla 6.4. *Valores obtenidos para cada uno de los filtros en función del orden empleado.*

Orden filtro MA	Orden filtro elíptico	$\Delta\omega$ para filtro con MA	$\Delta\omega$ para filtro elíptico
5	2	0.137	0.126
10	3	0.084	0.076
15	4	0.048	0.023
20	5	0.042	0.007
25	4	0.022	0.021

Tabla 6.5. *Anchos de banda de transición en función del orden del filtro.*

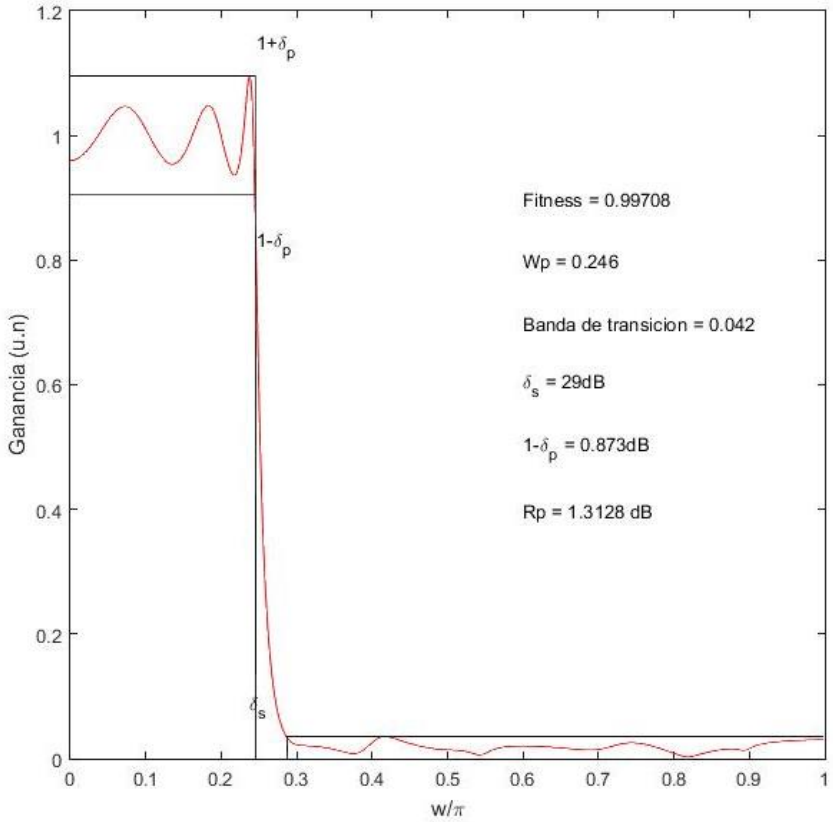

Figura 6.1*. Filtro paso bajo obtenido mediante MA de orden 20 con sus correspondientes especificaciones.*

Observando los resultados presentados en las tablas 6.4 y 6.5, lo más representativo son las diferencias entre los órdenes necesarios para cumplir las mismas especificaciones del filtro mediante los dos métodos. Queda de manifiesto que se necesita un orden mucho más bajo para cumplir las mismas especificaciones si se emplea un filtro elíptico, no obstante, el valor del fitness para el filtro elaborado mediante MA es siempre mayor que el resultante para el filtro elíptico.

Las especificaciones impuestas son los valores máximos permitidos, es decir, el rizado en la banda de paso y banda eliminada impuestos hacen referencia al mayor valor de los mismos en cada una de las bandas, considerando que el filtro elíptico posee un rizado monótono en cada una de las bandas siempre obtendrá un valor del fitness menor ya que el filtro mediante MA reduce su rizado con respecto al máximo a lo largo de ambas bandas. Este fenómeno se aprecia en la figura 6.2 donde tanto en la banda eliminada como en la de paso el filtro mediante MA (color azul) reduce su rizado a medida que nos alejamos de *Wp* y *Ws*, mientras que el filtro elíptico lo conserva.

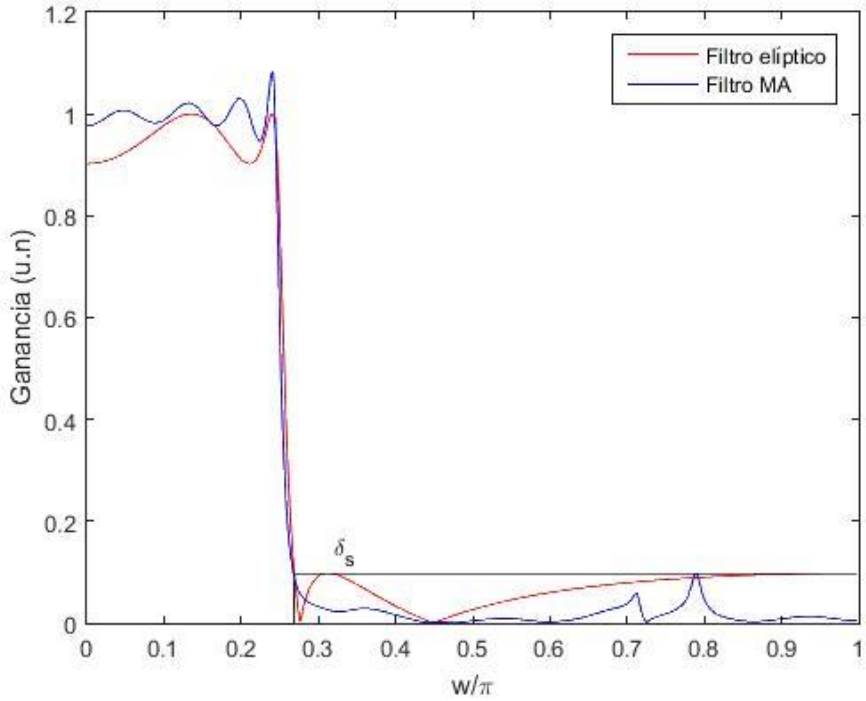

Figura 6.2. *Comparación entre filtro mediante MA (orden 25) y filtro elíptico (orden 5) paso bajo para las mismas especificaciones de rizado.*

Por otro lado, prestando atención al ancho de banda de transición de cada uno de ellos, no se encuentran diferencias notables, no obstante, el ancho de banda de transición para el filtro elíptico siempre es menor principalmente por el hecho de que este tipo de filtros se centran en optimizar esta característica, siendo los filtros con el menor ancho de banda para el mismo orden de entre los filtros de tipos Butterworth, Chebychev I y Chebychev II.

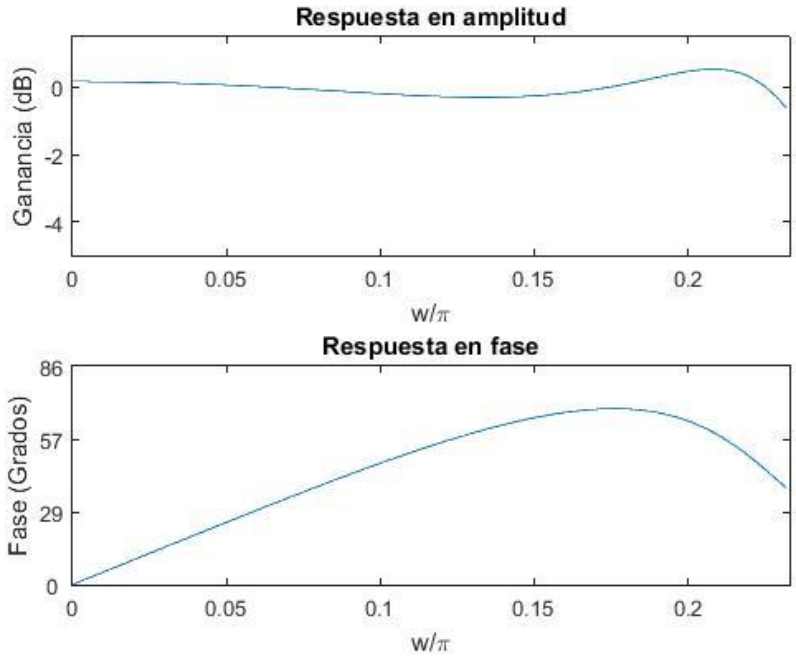

Figura 6.3. *Respuesta en amplitud (arriba) y fase (abajo) del filtro obtenido mediante MA de orden 10.*

Finalmente, en la figura 6.3 se muestra la respuesta del filtro tanto en amplitud como en fase del filtro obtenido mediante el MA basado en SFLA propuesto de orden 10 en el intervalo de frecuencias *[0, Wp]*. Observando el desfase producido frente a la banda de paso se puede afirmar que esta es mayoritariamente lineal excepto en las frecuencias cercanas a *Wp*. Esta linealidad en la banda de paso es bastante deseable ya que puede compensarse, evitando la distorsión en fase de la señal tras ser filtrada. Adicionalmente en la figura 6.4 se representa el filtro elíptico de orden 5 (correspondiente al filtro mediante MA de orden 20) tanto en amplitud como en fase en el intervalo *[0, Wp]*.

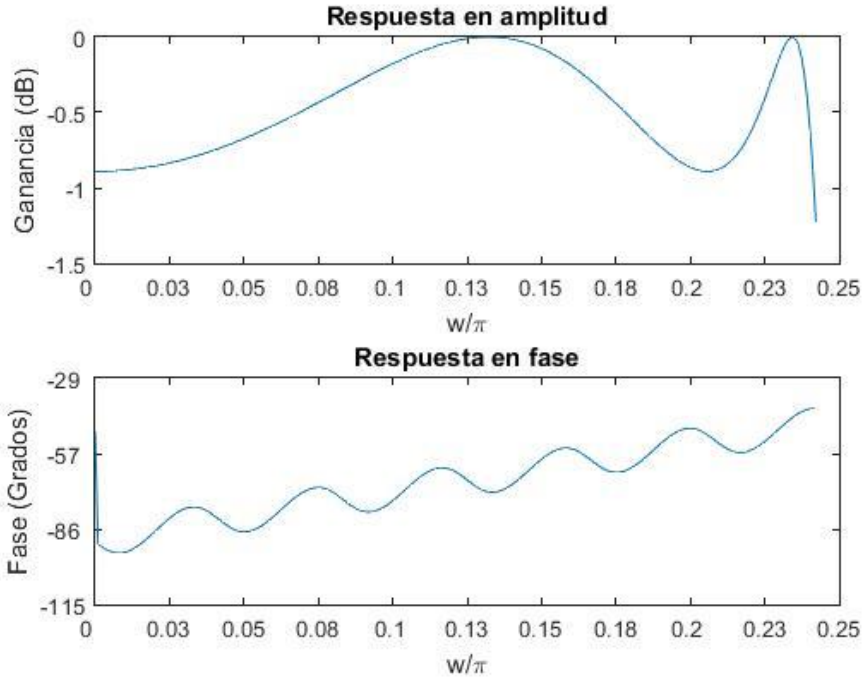

Figura 6.4. *Respuesta en amplitud y fase de filtro elíptico de orden 5.*

Examinando la figura 6.4 se aprecia que el filtro elíptico no posee una respuesta en fase lineal a lo largo de la banda de paso lo que puede provocar una distorsión en fase poco deseable a la señal filtrada, por tanto, es una desventaja con respecto al filtro obtenido mediante MA que poseía una linealidad aceptable.

Como conclusión se obtiene que para las mismas especificaciones de rizado los filtros elípticos necesitan un orden mucho menor y presentan una banda de transición ligeramente menor a los obtenidos mediante MA, no obstante, los segundos siempre obtienen un mayor fitness por su gran atenuación en la banda eliminada para frecuencias mayores a *Ws* y por la reducción del rizado en la banda de paso para frecuencias cercanas a 0. De forma que si el objetivo es optimizar la carga computacional la opción más adecuada son los filtros elípticos, pero si por el contrario la carga computacional es irrelevante y lo que se desea son unas mejores características entonces la mejor opción es el MA como método de diseño.

6.3.2 Aproximación a filtro de mayor orden

En muchos de los artículos empleados como fuentes para la elaboración del capítulo 4 se ha observado que para comparar el algoritmo que en cada uno se propone se trata de obtener la respuesta de un filtro preestablecido mediante otro filtro de orden inferior. Por tanto, se ha realizado el mismo experimento que en el primer ejemplo de [24], donde se trata de conseguir la misma respuesta que el filtro dado por la ecuación (6.2) mediante uno de orden 4.

$$H_s(z) = \frac{0.1084 + 0.5419z^{-1} + 1.0837z^{-2} + 1.0837z^{-3} + 0.5419z^{-4} + 0.1084z^{-5}}{1 + 0.9853z^{-1} + 0.9738z^{-2} + 0.3864z^{-3} + 0.1112z^{-4} + 0.0113z^{-5}} \quad (6.2)$$

Con el fin de comparar los resultados obtenidos con el MA propuesto, se han tomado los representados en la tabla 6.6 que son los obtenidos en [24] por Upadhyay.P, Kar.R, Mandal.D y Ghoshal.S.P.para distintos algoritmos bioinspirados.

Nº Ejecución	RGA		PSO		DE		FFA	
	MSE	Tiempo	MSE	Tiempo	MSE	Tiempo	MSE	Tiempo
1	0.0356	8.907633	0.0277	4.953836	6.8820e−004	6.661808	5.6061e−006	2.907472
2	0.0507	8.166661	0.0103	4.053498	0.0014	6.529037	1.8737e−006	2.869395
3	0.0991	8.082030	0.0068	4.098620	0.0022	6.485849	5.7630e−006	2.887492
4	0.0307	8.042549	0.0177	4.115985	9.7176e−004	6.699796	4.5938e−006	2.871119
5	0.0556	8.983041	0.0035	4.987767	0.0016	6.491667	4.7569e−006	2.872375

***Tabla 6.6**. Resultados obtenidos para distintos algoritmos bio-inspirados en [24].*

Para mantener el formato de comparación empleado en [24] se han realizado 5 ejecuciones por separado y en lugar de extraer el valor del fitness final se ha recalculado el MSE entre el filtro obtenido y el que se trata de conseguir ya que en el MA propuesto se plantean dos funciones de fitness y en [24] solo se trata de optimizar una función objetivo. Además, el objetivo del algoritmo SFLA propuesto es la obtención de un filtro lo más similar posible al filtro ideal correspondiente, por tanto, la función de fitness ha tenido que ser modificada y no todos los parámetros han sido optimizado. Los resultados obtenidos se presentan en la tabla 6.7.

Número de ejecución	MSE	Tiempo (segundos)
1	7,2494e-04	635,8478
2	5,8070e-04	687,4164
3	7,4292e-04	710,7843
4	1e-03	698,7441
5	6,3074e-04	624,8795

***Tabla 6.7.** MSE obtenida en 5 ejecuciones con el algoritmo SFLA propuesto.*

En base a los resultados plasmados en la tabla 6.7 se deduce que el MA propuesto genera un filtro IIR de orden 4 más similar al filtro de orden 5 dado por la ecuación (6.2) que varios de los algoritmos con los que se compara en [24], concretamente mejora los resultados obtenidos con respecto a los algoritmos RGA, PSO y DE. No obstante, el tiempo empleado en cada una de las ejecuciones es mucho mayor que en [24], en gran parte por la optimización de parámetros como el número de individuos o el número máximo de generaciones. El resultado de la

ejecución número 3 se ilustra en la figura 6.4, comparando el filtro que se trata de conseguir (en color rojo) y el filtro obtenido mediante el MA propuesto empleando un orden menor (en color azul).

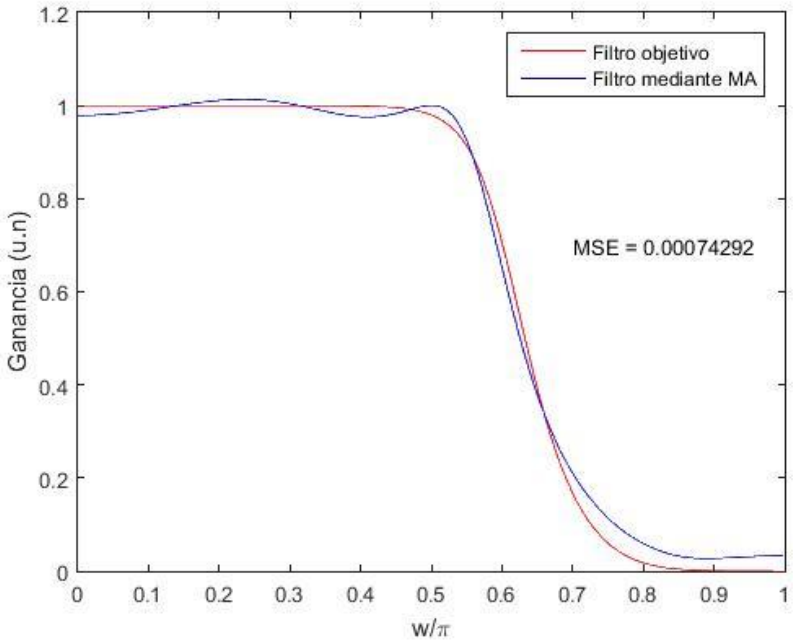

Figura 6.5. *Filtro objetivo de orden 5 (en color rojo) y filtro obtenido de orden 4 (en color azul).*

6.4 Filtro paso alto

Análogamente al caso del filtro paso bajo se ha generado un filtro paso alto variando el orden del mismo y manteniendo los valores de los parámetros que controlan el algoritmo con el fin de probar la capacidad del MA de generar otro tipo de filtros. Para comparar el resultado con el obtenido para un filtro elíptico se ha calculado el orden mínimo del mismo y el fitness que posee.

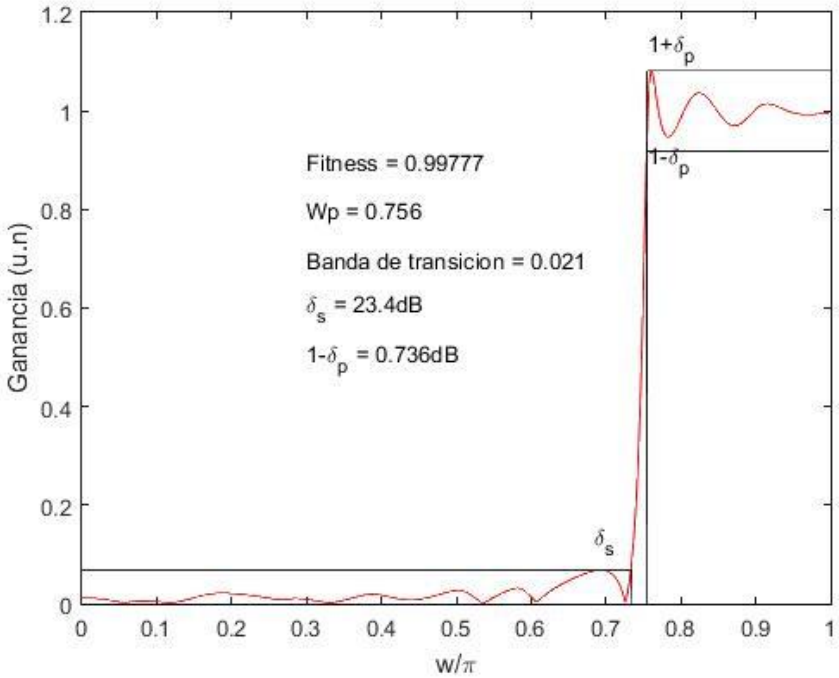

Figura 6.6. *Filtro paso bajo obtenido mediante MA de orden 25 con sus correspondientes especificaciones.*

En la figura 6.6 se representa la respuesta en frecuencia de un filtro paso alto de orden 25 con una frecuencia de corte ideal en 0.25 generado mediante el MA propuesto, además de las especificaciones que luego se le impondrán al filtro elíptico para proceder a su comparación. Observando los rizados en la banda eliminada y de paso se extrae la característica que hace que los filtros generados mediante el MA posean un mayor fitness que los elípticos (como se muestra en la tabla 6.8).

Orden con MA	**Orden elíptico**	**1 - δp**	**δs**	**Wp**	**Fitness con MA**	**Fitness elíptico**
5	4	0.423	20.1	0.772	0.9917	0.9835
10	4	0.379	20.2	0.769	0.9937	0.9876
15	5	0.858	22.8	0.756	0.9972	0.9909
20	5	0.736	23.4	0.756	0.9977	0.9922
25	5	0.549	21.8	0.756	0.9977	0.9929

Tabla 6.8. *Resultados obtenidos para filtro paso alto elíptico y mediante MA.*

Orden con MA	Orden con elíptico	Δω con MA	Δω para filtro elíptico
5	4	0.073	0.025
10	4	0.057	0.026
15	5	0.022	0.008
20	5	0.021	0.010
25	5	0.02	0.009

***Tabla 6.9**. Ancho de banda de transición obtenido para filtro paso alto elíptico y mediante MA.*

Finalmente, en la figura 6.7 se muestra una representación de la respuesta en frecuencia del filtro mediante MA (color azul) y del filtro elíptico (color rojo), de forma que se aprecia la diferencia de rizado entre ambos para cada banda. Este fenómeno es más relevante en la banda eliminada donde el filtro mediante MA obtiene un rizado decreciente a medida que las frecuencias se alejan de *Wc*, al contrario que le filtro elíptico.

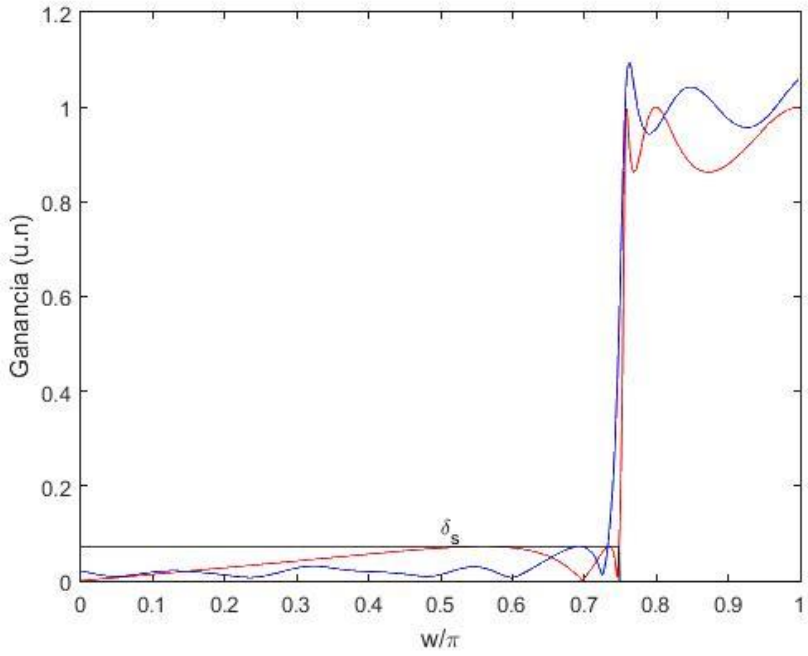

Figura 6.7. *Comparación entre filtro mediante MA de orden 15 y filtro elíptico de orden 5 paso alto para las mismas especificaciones de rizado.*

6.5 Filtro paso banda

Análogamente al caso del filtro paso bajo se ha generado un filtro paso banda variando el orden del mismo y manteniendo los valores de los parámetros que controlan el algoritmo con el fin de probar la capacidad del MA de generar otro tipo de filtros. Los resultados obtenidos se presentan en la tabla 6.10.

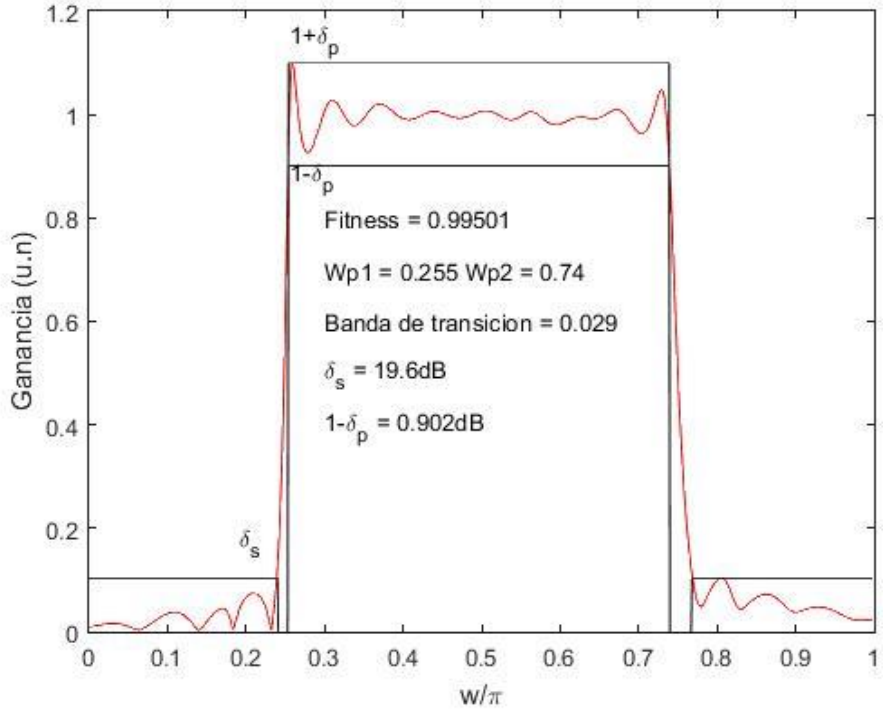

Figura 6.8. *Filtro paso banda obtenido mediante MA de orden 30 con sus correspondientes especificaciones.*

Analizando la figura 6.8 que representa la respuesta en frecuencia de un filtro paso banda de orden 30 generado mediante el MA propuesto con frecuencias ideales de corte de 0.25 y 0.75, se observa que el rizado en la banda eliminada disminuye considerablemente para frecuencias cercanas a 0 y 1, obteniendo una mayor atenuación que la establecida mediante δ_s. El mismo fenómeno se observa para la banda de paso donde el rizado disminuye considerablemente para frecuencias cercanas a 0.5 (centro de la banda de paso), esto lleva que los filtros generados mediante el MA propuesto posean un mayor fitness a los elípticos como se puede comprobar en la tabla 3.10.

Orden con MA	Orden elíptico	1 - δp	δs	Wp	Fitness con MA	Fitness elíptico
10	8	0.844	19.3	[0.282, 0.735]	0.9855	0.9635
16	6	0.629	19.1	[0.266, 0.734]	0.9903	0.9880
20	8	0.901	21.3	[0.259, 0.741]	0.9945	0.9889
26	8	0.723	21.5	[0.258, 0.739]	0.9944	0.9914
30	8	0.902	19.6	[0.255, 0.740]	0.9950	0.9879

***Tabla 6.10**. Resultados obtenidos para filtro paso banda elíptico y mediante MA.*

Orden con MA	Orden para elíptico	Δω con MA	Δω para filtro elíptico
10	8	0.091	0.011
16	6	0.044	0.038
20	8	0.026	0.014
26	8	0.057	0.016
30	8	0.029	0.011

***Tabla 6.11**. Ancho de banda obtenido para filtro paso banda elíptico y mediante MA.*

En la tabla 6.11 se muestra el ancho de banda de transición correspondiente a cada uno de los filtros para cada orden empleado, a pesar de que los filtros paso banda poseen dos bandas de transición se ha tomado el valor de la menos abrupta. Esto es importante en el caso del filtro obtenido mediante MA ya que no tiene por qué ser simétrico a diferencia del elíptico. De estos resultados se obtiene la conclusión de que la banda de transición para los filtros elípticos es siempre menor que para los generados mediante MA, esto es debido a que los primeros se centran en optimizar este parámetro.

A continuación, en la figura 6.9 se muestran las gráficas del filtro mediante MA (en color azul) y del filtro elíptico (en color rojo), poniendo de manifiesto unas bandas de transición mayores para el filtro mediante MA pero un menor rizado en comparación con el elíptico.

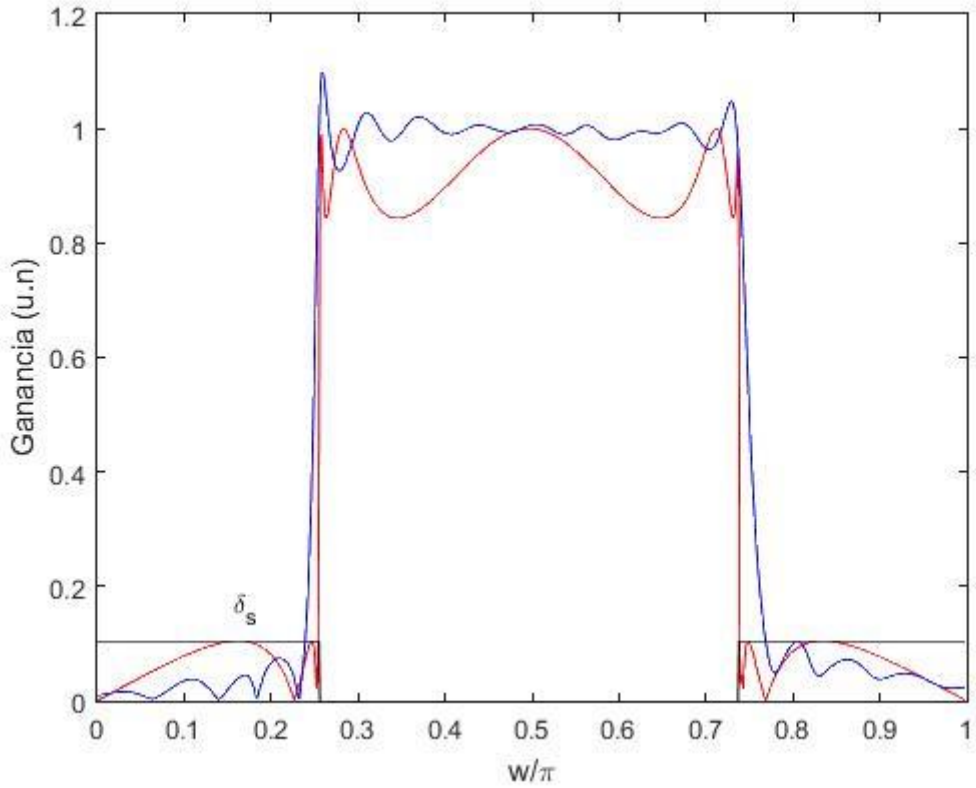

Figura 6.9. *Comparación entre filtro mediante MA de orden 30 y filtro elíptico de orden 8 paso banda para las mismas especificaciones de rizado.*

6.6 Filtro de banda eliminada

Análogamente al caso del filtro paso bajo se ha generado un filtro de banda eliminada variando el orden del mismo y manteniendo los valores de los parámetros que controlan el algoritmo con el fin de probar la capacidad del MA de generar otro tipo de filtros. Para comparar el resultado con el obtenido para un filtro elíptico se ha calculado el orden mínimo del mismo y el fitness que posee. En la figura 6.10 se muestra la respuesta al impulso de un filtro de banda eliminada generado mediante el Ma propuesto para orden 30, además se reflejan las especificaciones del mismo.

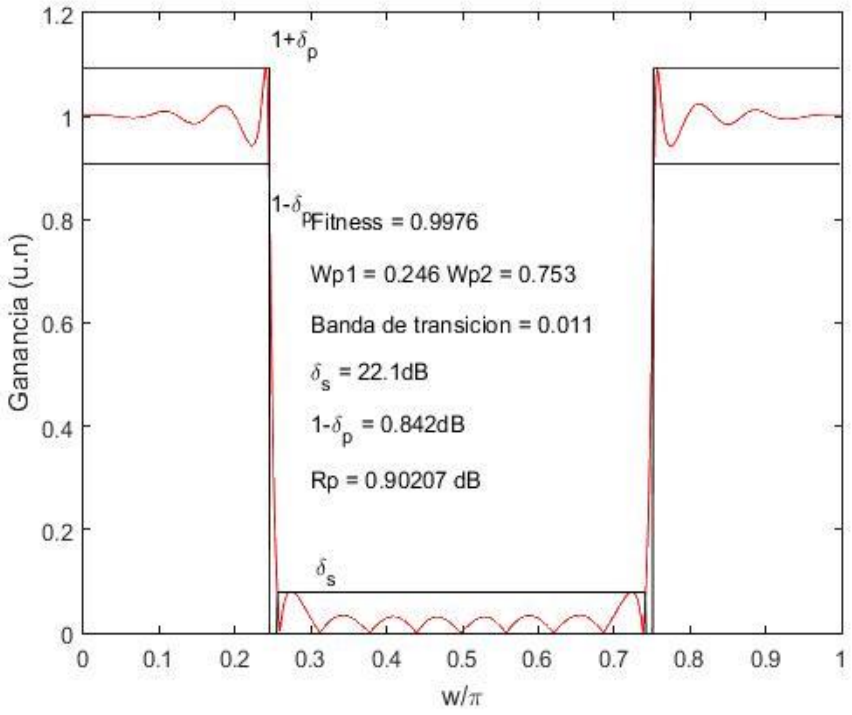

Figura 6.10. *Filtro de banda eliminada obtenido mediante MA de orden 30 con sus correspondientes especificaciones.*

Al igual que en el caso del filtro paso bajo, aunque las especificaciones del filtro no puedan parecer demasiado restrictivas hay que tener en cuenta que el rizado en la banda de paso disminuye considerablemente para frecuencias cercanas a 0 o 1, y lo mismo ocurre para la banda eliminada donde posee una mucha mayor atenuación en frecuencias cercanas a 0.5 en comparación con el Rp mínimo. Estas dos características hacen que el fitness obtenido para el filtro mediante MA se siempre mayor que el correspondiente al filtro elíptico que posee rizado monótono en ambas bandas. Esta comparación queda reflejada en la tabla 6.12.

Orden con MA	**Orden elíptico**	**1 - δp**	**δs**	**Wp**	**Fitness con MA**	**Fitness elíptico**
10	6	1.066	16.2	[0.226, 0.777]	0.9795	0.9729
16	6	1	26.5	[0.239, 0.764]	0.9917	0.9808
20	6	0.83	20	[0.239, 0.761]	0.9911	0.9870
26	10	1.092	23	[0.256, 0.753]	0.9976	0.9944
30	10	0.842	22.1	[0.246, 0.753]	0.9976	0.9955

Tabla 6.12. *Resultados obtenidos para filtro de banda eliminada elíptico y mediante MA.*

Orden con MA	Orden para elíptico	Δω con MA	Δω para filtro elíptico
10	6	0.079	0.023
16	6	0.065	0.058
20	6	0.049	0.039
26	10	0.012	0.006
30	10	0.011	0.006

Tabla 6.13. *Ancho de banda obtenido para filtro de banda eliminada elíptico y mediante MA.*

En la tabla 6.13 se muestran los anchos de banda de transición obtenidos para cada uno de los filtros en función de su orden. A pesar de que los filtros de banda eliminada tienen dos anchos de banda de transición se ha tomado en cada caso la transición menos abrupta, esto es importante en el caso del filtro mediante MA ya que no tienen por que ser iguales ambas bandas de transición, al contrario que en el caso del filtro elíptico. A continuación, en la figura 6.11 se muestran las gráficas del filtro mediante MA (en color azul) y del filtro elíptico (en color rojo), poniendo de manifiesto unas bandas de transición muy similares pero un menor rizado en el caso del filtro mediante MA.

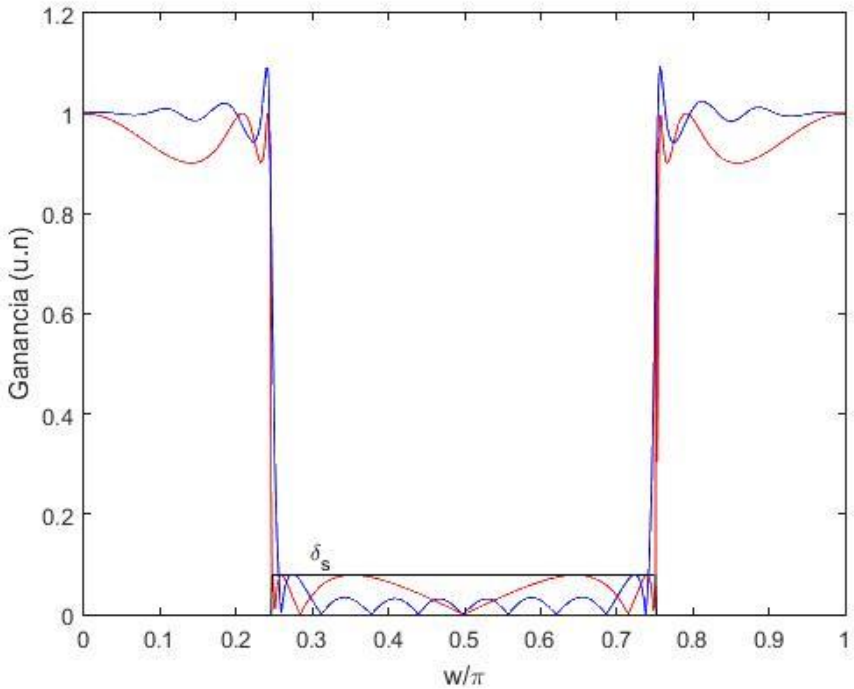

Figura 6.11. *Comparación entre filtro mediante MA de orden 30 y filtro elíptico de orden 8 de banda eliminada para las mismas especificaciones de rizado.*

6.7 Aplicación a filtros FIR

Como método adicional de evaluar el funcionamiento del algoritmo memético propuesto se ha modificado la implementación para adaptarla al diseño del filtro FIR o de respuesta finita. Estos filtros nos poseen polos, su respuesta al impulso está solo formada por los coeficientes del numerador, es decir, solo viene dada por ceros.

Como método de evaluación se han comparado los resultados obtenidos mediante el MA propuesto con el cálculo de filtros FIR mediante el método del enventanado, concretamente empleando ventana de Hamming y de Blackman-Tuckey, en el diseño de un filtro paso bajo con frecuencia de corte ideal en 0.25 (frecuencia normalizada). La primera ventana posee una menor atenuación en la banda eliminada por su lóbulo secundario más pronunciado, sin embargo, presenta un ancho de banda del lóbulo principal mucho menor. En el caso de la ventana de Blackman-Tuckey, la atenuación en la banda eliminada es mayor pero el ancho del lóbulo principal supera al de la ventana de Hamming.

Para cuantificar la calidad de los filtros obtenidos se ha seguido un procedimiento análogo al de los filtros IIR, manteniendo los valores de los parámetros indicados en la tabla 6.1, estableciendo N=5 y m=5, con la excepción de que en este caso los filtros FIR generados tienen el mismo orden que el generado mediante MA. Se han comparado los valores del fitness obtenidos en función del orden del filtro además del ancho de banda de transición.

Para calcular la frecuencia de paso Wp se ha establecido un rizado en la banda de paso de 1dB, mientras que el rizado en la banda eliminada lo establece el filtro que mayor valor posea. Los valores obtenidos para el fitness en función del orden para el filtro mediante MA y el FIR con ventana Blackman-Tuckey quedan reflejados en la tabla 6.14, adicionalmente en la tabla 6.15 se ilustra el ancho de banda de transición resultante.

Orden	δ_s	Wp con MA	Wp con FIR	Fitness con MA	Fitness con Blackman
5	15.488	0.157	0.181	0.9702	0.9568
10	20.258	0.202	0.109	0.9829	0.9568
15	17.278	0.22	0.152	0.9876	0.9754
20	16	0.232	0.203	0.9903	0.9834

Tabla 6.14. *Valores de fitness obtenidos para ordenes distintos en filtro mediante MA y FIR con ventana de Blackman-Tuckey.*

Orden	$\Delta\omega$ con MA	$\Delta\omega$ FIR con Blackman
5	0.224	0.329
10	0.143	0.263
15	0.09	0.17
20	0.062	0.099

Tabla 6.15. *Valores de Δω obtenidos para ordenes distintos en filtro mediante MA y FIR con ventana de Blackman-Tuckey.*

Observando los resultados obtenidos se distingue una clara superioridad en el valor del fitness en el caso del filtro mediante MA, especialmente para ordenes bajos, a medida que este aumenta el fitness de ambos comienza a ser similar. Con respecto al ancho de banda de transición, el filtro mediante MA ha conseguido transiciones más abruptas para todos los órdenes, no obstante, se repite el fenómeno ocurrido con el fitness, a medida que aumenta el orden la ventaja del MA empieza a desaparecer. En la figura 6.12 se muestra la representación en frecuencias de los tres filtros de orden 15: mediante el MA propuesto, FIR con ventana de Blackman-Tuckey y el ideal.

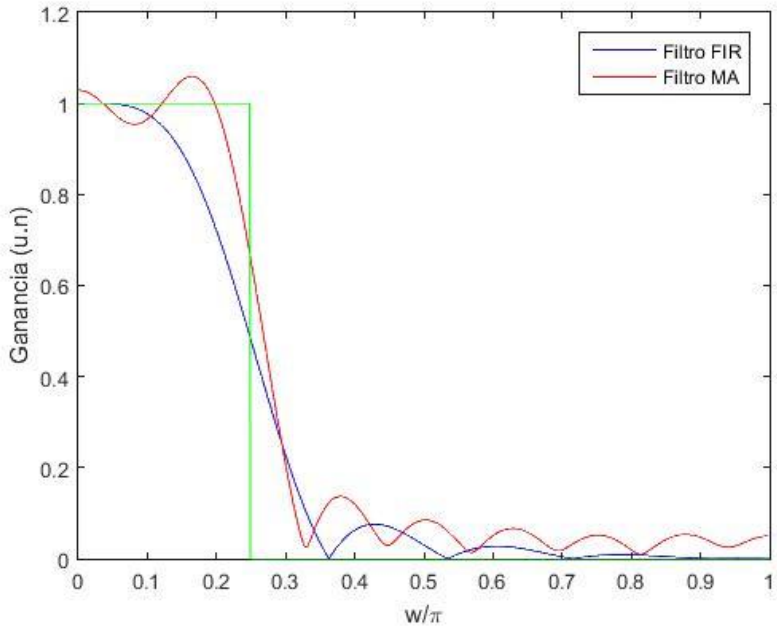

Figura 6.11. *Comparativa entre filtro mediante MA (en color rojo) y FIR con ventana de Blackman-Tuckey (en color azul), ambos de orden 15.*

A modo de comparativa se ha ejecutado el mismo proceso, pero en este caso para filtro FIR mediante enventanado con ventana de Hamming. Los valores del fitness obtenidos para el filtro mediante MA y el FIR con ventana de Hamming se muestran en la tabla 6.16. De forma análoga, en la tabla 6.17 se muestran los anchos de banda correspondientes.

Orden	δs	Wp con MA	Wp con FIR	Fitness con MA	Fitness con Hamming
5	10.191	0.157	0.159	0.9700	0.9446
10	18.581	0.202	0.115	0.9829	0.9650
15	17.278	0.22	0.134	0.9878	0.9710
20	16	0.232	0.162	0.9903	0.9781

Tabla 6.16. *Valores de fitness obtenidos para ordenes distintos en filtro mediante MA y FIR con ventana de Hamming.*

Orden	Δω con MA	Δω FIR con Hamming
5	0.224	0.304
10	0.143	0.31
15	0.09	0.223
20	0.062	0.161

Tabla 6.17. *Valores de Δω obtenidos para ordenes distintos en filtro mediante MA y FIR con ventana de Hamming.*

Al igual que para el diseño del filtro FIR mediante ventana de Blackman-Tuckey, en este caso también dominan los filtros diseñados mediante el MA, obteniendo un valor de fitness superior para todos los órdenes estudiados. En este caso también ocurre que, al aumentar el orden, ambos filtros comienzas a tener valores de fitness similares pero de forma mucho menos evidente que en el caso con ventana de Blackman-Tuckey. Finalmente se observa que el ancho de banda de transición es mucho menor para el caso del filtro mediante MA, incluso para orden 15 o superior.

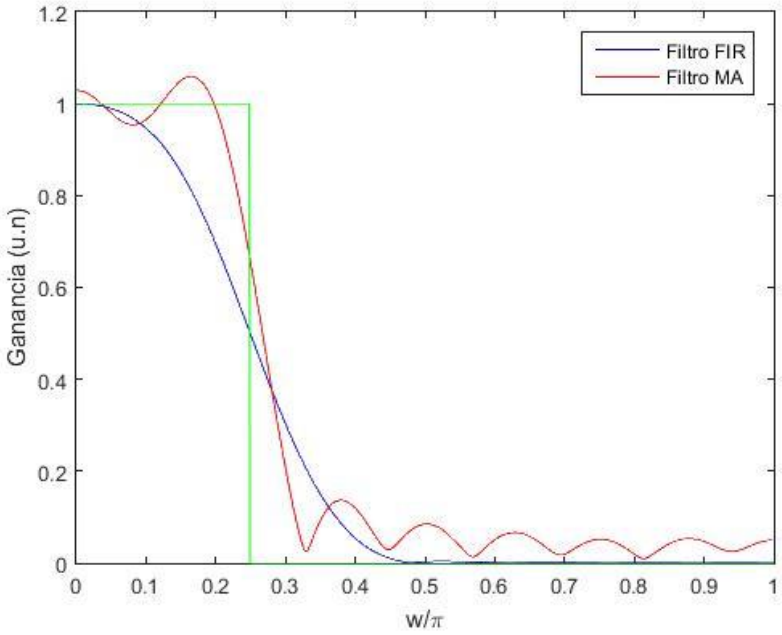

Figura 6.12. *Comparativa entre filtro mediante MA (en color rojo) y FIR con ventana de Hamming (en color azul), ambos de orden 15.*

Finalmente, en la figura 6.13 se muestran las respuestas en fase del filtro obtenido mediante MA, el filtro FIR empleando la ventana de Blackman-Tuckey y el filtro FIR empleando la ventana de Hamming, en el rango de frecuencias *[0,Wc]*.

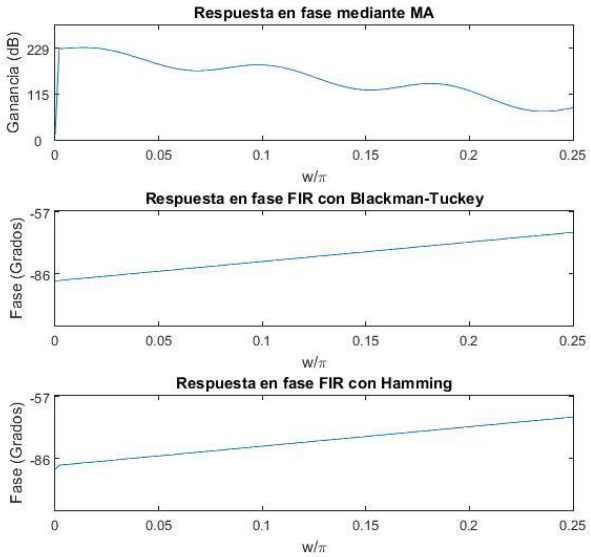

Figura 6.13. *Respuesta en fase del filtro mediante MA (arriba), del filtro FIR con ventana de Blackman-Tuckey (medio) y del filtro FIR con ventana de Hamming (abajo).*

Como se puede apreciar en la figura 6.13, en el caso de los filtros FIR mediante enventanado, se tiene linealidad en la respuesta en fase, que es una de las características propias y más deseables de los filtros FIR. Por otro lado, la respuesta en fase del filtro obtenido mediante MA no es del todo lineal ya que presenta un ligero "rizado", lo que podría producir distorsión en fase. Por tanto, con respecto a la respuesta en fase del filtro, los filtros FIR mediante enventanado poseen una gran ventaja frente a los generados mediante el MA propuesto en este aspecto.

7. Conclusiones y futuras líneas de trabajo

En este trabajo se presenta un método de optimización metaheurístico basado en SFLA que pertenece a los algoritmos de evolución memética, con el fin de abordar el diseño de filtros IIR empleando dos funciones objetivo. El algoritmo SFLA básico no es capaz de abordar problemas con más de una función objetivo y no emplea mecanismos para evitar la convergencia prematura, por tanto, se han propuesto una serie de mejoras para mejorar los resultados obtenidos de cara el diseño de filtros IIR. Primero se emplea un método de generación de grupos basado en el plano de funciones objetivo estableciendo el centro de cada subgrupo memético con el fin de mejorar la convergencia de las soluciones. Adicionalmente se introduce un mecanismo de mutación sobre las mejores soluciones que entra juego cuando se detecta un estancamiento en un óptimo local, de esta forma se asegura un mantenimiento en la convergencia y una solución mejor solución final. Finalmente se ha adaptado el algoritmo a problemas multiobjetivo con el fin de poder controlar qué características de las soluciones son prioritarias en el proceso de optimización. Estas modificaciones suponen una gran mejora en el funcionamiento del algoritmo, a la par que aumentan el abanico de problemas al que puede ser aplicado, concretamente a problemas multiobjetivo.

Uno de los inconvenientes que este algoritmo presenta es el tiempo de ejecución que requiere para obtener unos resultados óptimos, habiéndose realizado todas las pruebas en un ordenador ASUS con AMD A10 a 2.50 GHz. Del tiempo total que lleva cada una de las ejecuciones aproximadamente un noventa por ciento se invierte en el cálculo de fitness de cada uno de los individuos en cada salto de cada iteración. Esto se debe a que para cada uno de los individuos debe calcularse el filtro correspondiente a los coeficientes que lo conforman, por tanto, si la población consta P individuos y se realizan N saltos por iteración, entonces deberán generarse $P*(N+1)$ filtros por iteración, lo que supone un gran número de operaciones. También es importante tener en cuenta el orden empleado ya que a mayor orden mayor es el tiempo de generación de cada filtro, del cálculo del salto y de mutación para cada rana. No obstante, un mayor número de saltos por iteración aseguran una convergencia más rápida y una mayor población supone un espacio mayor de búsqueda al inicio.

En vistas a estos resultados es evidente que el número máximo de individuos y el número de saltos por iteración son dos parámetros que deben ajustarse con cierta precisión para mantener un equilibrio entre la calidad de los resultados obtenidos y el tiempo de ejecución. En esta memoria en particular se han empleado tamaños de población pequeños en comparación con los empleados en [23], cuyo valor ronda los 700 individuos, un tamaño inmanejable para esta implementación concreta y para el ordenador con el que se han realizado las pruebas. Esta decisión se ha tomado en base a que no se obtenía mejora para poblaciones mayores, al menos hasta 100 individuos, entonces la mejor opción es escoger la menor población para la que los resultados no empeoran significativamente. Con respecto al número de saltos por iteración, debe escogerse un valor superior a la mitad de individuos por memeplexes, si se escoge un número de saltos reducido la ejecución se acelera pero también la velocidad de convergencia se reduce, necesitando un aumento en la iteraciones máximas. El otro caso extremo es emplear

un número de saltos muy elevado con el objetivo de acelerar lo máximo posible la convergencia, esta no es una decisión muy acertada ya que experimentalmente se ha observado que llega un valor de *N* para el que la convergencia no se acelera más, e incluso puede producir convergencia prematura.

Con respecto a la aplicación de este algoritmo al diseño de filtros IIR, se han encontrado ciertas limitaciones de diseño que en otros ámbitos no existen y que juegan un papel importante en el proceso de optimización. La más importante ellas es el tamaño del salto, con el fin de mantener la normalización de los coeficientes por debajo de uno, el escalado del salto siempre debe ser menor que uno y no siempre se realiza en la misma dirección exacta que la mejor rana. Esta limitación llevó a descartar la idea de emplear un escalado del salto inicial mayor para ampliar el campo inicial de búsqueda, donde se les permitiera a las ranas saltar más allá de la mejor de ellas.

La calidad de los filtros obtenidos es buena, al menos en comparación con los filtros elípticos, si se renuncia a la agilidad de ejecución que estos últimos presentan. Por otro lado, también se han obtenido buenos resultados a la hora de tratar de conseguir la respuesta de un filtro a partir de otro de menor orden, no obstante, este no es el objetivo principal del MA propuesto ya que su característica más distintiva es su aplicación a problemas con más de una función objetivo. Esta característica se ha aprovechado en este caso en el proceso de creación de una respuesta en frecuencia lo más similar posible a la ideal, se ha planteado la posibilidad de que una de las funciones objetivo fuera destinada a evaluar la respuesta en fase del filtro con el fin de conseguir linealidad. A pesar de no evaluarse la respuesta en fase, se ha demostrado que al conseguir un rizado que disminuye progresivamente en la banda eliminada se consigue de forma paralela una respuesta en fase mayoritariamente lineal, excepto para frecuencias cercanas a la de corte, lo cual es una característica muy deseable en el tratamiento de señales.

En la parte final de resultados numéricos también se comprobó la capacidad del MA propuesto para el diseño de filtros FIR, realizando comparaciones con el método de enventanado para las ventanas de Blackman-Tuckey y Hamming. En todos los casos el fitness obtenido para el filtro mediante MA fue superior, no obstante, a medida que se aumenta el orden esta diferencia se va reduciendo. Como conclusión se extrae que el MA propuesto es un buen método de diseño de filtros FIR sin realizar a penas cambios, para órdenes inferiores a 20, ya que los parámetros eran idénticos al caso de diseño de filtros IIR.

Como futuras líneas de trabajo se propone la mejora del rendimiento del algoritmo, concretamente la disminución de la carga computacional que supone, ya que en este ámbito no puede hacer frente a otra gran variedad de algoritmos más sencillos y que requieren de un menor número de recursos. De esta forma podría ser empleado para sistemas de filtrado adaptativo en tiempo real.

Otra posible mejora a estudio sería la introducción de variaciones en las funciones de fitness con el objetivo de conseguir un mayor control de los filtros de salida, es decir, poder establecer ciertas prioridades para diseño como pueden ser minimizar el ancho de banda de transición,

mejorar la linealidad de la respuesta en fase o escoger en cual de las dos bandas se desea que el rizado sea más favorable.

8. Referencias bibliográficas

[1] Back.T, Hammel.U, Schwefel.H.P. “Evolutionary Computation: Comments on the History and Current State”. IEEE Transaction On Evolutionary Computing, volumen 1, numero 1, año 1997.

[2] Moscato. P, Cotta. C. “Una introducción a los algoritmos meméticos”. Inteligencia artificial. Revista iberoamericana de Inteligencia artificial, año/vol 7, número 019.

[3] Cotta.C. “Una Visión General de los Algoritmos Meméticos “. Dept. Lenguajes y Ciencias de la Computación, ETSI Informática, University of Málaga.

[4] Moscato.P, Cotta.C. ” Memetic Algorithms.”
http://www.lcc.uma.es/~ccottap/papers/memetic_HAAM.pdf

[5] Whitley.D, Mathias.K. “Genetic Operators. The Fitness Landscape and the Traveling Salesman Problem.”. Parallel Problem Solving from Nature. Páginas 219-228, año 1992.

[6] E.Goldberg.D, Lingle.R. “Alleles, loci and the traveling salesman problem”. Proceedings of the First International Conference on Genetic Algorithms and their Applications. Páginas 178-183, año 1985.

[7] Deep.K, Mebrahtu.H. ”Variant of partially mapped crossover for the Traveling Salesman problems.”. International Journal of Combinatorial Optimization Problems and Informatics. Volumen 3, numero 1, páginas 47-69, año 2012.

[8] Krasnogor.N, Aragón.A, Pacheco.J. ”Chapter 12 Memetic Algorithms” .

[9] Zhao.Z, Gao.H, Liu.Y. “Chaotic Particle Swarm Optimization for FIR Filter Design”. International conference on electrical and control engineering. Año 2011.

[10] Gao.H, Diao.M. ”Differential Cultural Algorithm for Digital Filter Design”. Second International Conference on Computer Modeling and Simulation. Año 2010

[11] Storn.R, Price.K. “Differential Evolution-A Simple and Heuristic for Global Optimization over Continuous Spaces”. Journal of global optimization. Volume 11, Issue 4, páginas 341–359, año 1997.

[12] Starkweather.T, McDaniel.S, Whitley.C, Mathias.K, Whitley.D. “A Comparison of Genetic Sequencing Operators”. Proceedings of the fourth International Conference on Genetic Algorithms. Páginas 69-76, año 1991.

[13] Eusuff.M, Lansey.K, Pasha.F. “Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization”. Engineering optimization. Volume 38, Issue 2, paginas 129-154, año 2004.

[14] Yue.Y, Wang.X. “An Improved Ant Colony Optimization Algorithm for Solving TSP”. International Journal of Multimedia and Ubiquitous Engineering, volumen.10, numero.12, páginas.153-164, año 2015.

[15] Dorigo.M, Gambardella.L.M. “Ant Colony System A Cooperative Learning Approach to the Traveling Salesman Problem”. IEEE Transactions on Evolutionary Computation, volumen 1, numero 1, páginas 53-66,año 1997.

[16] Dorigo.M, Birattari.M, Stützle.T. “Ant Colony Optimization. Artificial Ants as a Computational Intelligence Technique”. IEEE Computation Intelligence Magazine, volumen 1, páginas 28-39, año 2006.

[17] “Passive, Active, and Digital Filters. Chapter 19”. Edited by Wai-Kai Chen. (Bibliografía de la asignatura TAS).

[18] Proakis.J.G, Manolakis.D.G. “Digital Signal Processing Principles, Algorithms, and Applications Third Edition”. Capítulo 8, sección 3, páginas 666-692, año 1996.

[19] Wang.J, Shi.P, Peng.H. “Membrane computing model for IIR filter design”. Information Sciences 329, páginas 164–176, año 2016.

[20] Freund.R, Paun.G, Pérez Jiménez.M.J. “Tissue-like p systems with channel-states”. Proceedings of the Second Brainstorming Week on Membrane Computing, páginas 206-223, año 2004.

[21] Kalinli.A, Karaboga.N. “A new method for adaptive IIR filter design based on tabu search algorithm”.Int. J. Electron. Commun 59, páginas 111–117, año 2005.

[22] Sidhu.D.S, Dhillon.J.S, Kaur.D. “Design of Digital IIR Filter with Conflicting Objectives Using Hybrid Gravitational Search Algorithm”. Mathematical Problems in Engineering, volumen 2015, 16 páginas.

[23] Luo.J, Yang.T, Liu.Q, Li.X, Chen.M, Gao.K. “A new hybrid memetic multi-objective optimization algorithm for multi-objective optimization”. Information Sciences, volúmenes 448–449, páginas 164–186, año 2018.

[24] Upadhyay.P, Kar.R, Mandal.D, Ghoshal.S.P. “A new design method based on firefly algorithm for IIR system identification problem”. Journal of King Saud University – Engineering Sciences 28,páginas 174–198, año 2016.

[25] Upadhyaya. P, Kar. R, Mandal. D, Ghoshal. S. P. “Craziness based particle swarm optimization algorithm for IIR system identification problem”. International Journal of Electronics and Communications (AEÜ), volumen 68, páginas 369-378, año 2014.