UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA DE TELECOMUNICACION
SISTEMAS DE TELECOMUNICACION

PROYECTO FIN DE CARRERA

OPTIMIZACIÓN DE ARRAYS MULTIFRECUENCIA MEDIANTE UN ALGORITMO DE OPTIMIZACIÓN GLOBAL

Autor: JORGE MIJARRA MARTÍNEZ
Tutor: ÓSCAR QUEVEDO TERUEL

JULIO 2010

Índice

Agradecimientos

El proyecto descrito en estas páginas es el fruto del trabajo comenzado hace unos cuantos años cuando comencé la universidad. No pensé que el camino hasta este momento iba a ser, en algunos momentos, tan duro y largo, pero por fin puedo decir que se terminó.

Y mucha culpa de que haya terminado la carrera con este proyecto es de todas las personas que han estado conmigo todos estos años, y a los que me gustaría agradecer todo su apoyo.

A mis padres, que siempre han estado y estarán a mi lado en los momentos complicados, que me han apoyado y ayudado y, sobre todo, agradecerles que desde pequeño me educaran para que insistiera y me esforzara en conseguir lo que quería. Esta es una de las cosas que quería, y la he conseguido.

A mi hermano Jesús, una de las personas importantes en mi vida y que, a su manera, siempre me ha ayudado a superar periodos complicados en todos estos años. Gracias por todo. Espero poder ayudarle a terminar su carrera, aunque con la trayectoria que lleva seguro que no necesita de mi ayuda pero siempre tendrá mi apoyo.

A mis grandes amigos que he conocido durante mi etapa en la universidad, con los que he compartido exámenes, prácticas, días malos, días buenos, risas,..... Espero que nuestra amistad perdure durante los años futuros. Para que no haya “envidias”, los voy a nombrar por orden alfabético: Alberto, Charly, Goyo, Iván, Kiko, Mari, María, Nacho, Pablo, Pedro y Sergio. Podría decir muchísimas cosas de cada uno, pero resumiendo, todos ellos son grandes personas y buenos amigos. Y sin olvidarme de José Fernando, “*Basic*”.

Por último, solo recordar al resto de mi familia, los que están y los que, desgraciadamente, no están ya, a mi amigo Ricardo, y a otros muchos amigos y compañeros de clase.

Podría seguir recordando a gente que me ha ayudado y han sido importantes en mi vida universitaria, pero entonces los agradecimientos ocuparían más que el propio proyecto.

Gracias a todos.

Jorge

Capítulo 1. Introducción

En este primer capítulo se presenta una pequeña introducción a los objetivos definidos inicialmente para el proyecto y la estructura que se ha definido para la redacción de esta memoria.

1.1. Objetivos y Estructura del Proyecto

El objetivo principal del presente proyecto es conseguir optimizar arrays de antenas monofrecuencia y multifrecuencia mediante la aplicación de algoritmos de optimización globales. Los algoritmos que se han escogido han sido: algoritmos genéticos, algoritmos basados en enjambres de partículas y algoritmos basados en colonias de hormigas.

El primero de los objetivos del proyecto es realizar el diseño de un array monofrecuencia de antenas que cumpla unos requisitos marcados inicialmente mediante la utilización de un algoritmo de optimización. Estos requisitos van a estar relacionados con la relación de lóbulo principal a secundario, que será la condición con la que, iterativamente, se optimizará el array. Una vez analizados los resultados obtenidos, se procederá a la elección del algoritmo apropiado para el diseño de un array bifrecuencia en el que se intente también minimizar el nivel de lóbulos secundarios.

En cuanto a la estructura del proyecto, éste se va a dividir en 5 capítulos. En el capítulo 2 se realizará una introducción al estado del arte de los algoritmos de búsqueda global que serán posteriormente utilizados en el proyecto. En el capítulo 3 se expondrá la formulación básica derivada de la teoría de arrays, el concepto de *thinned* arrays, y se presentarán los resultados de aplicar los métodos de optimización a arrays monofrecuencia. En el capítulo 4 se extenderán los resultados obtenidos anteriormente al diseño de arrays bifrecuencia. Por último, en el capítulo 5 se extraerán las conclusiones obtenidas de analizar los resultados previamente obtenidos.

Capítulo 2. Estado del Arte

En este capítulo se realiza una introducción a los principios básicos, características y una breve reseña histórica de los algoritmos que se utilizarán posteriormente en este proyecto: Algoritmo Genético (AG), Algoritmo Basado en Enjambres de Partículas (o PSO) y Algoritmo Basado en Colonias de Hormigas (ACO).

2.1. Algoritmo Genético Básico

2.1.1 Fundamentos

Los Algoritmos Genéticos [1] son métodos sistemáticos para la resolución de búsqueda y optimización que aplican los mismos métodos de la evolución biológica: selección basada en la población, reproducción y mutación.

Los Algoritmos Genéticos (AG), como se ha dicho, son métodos de optimización que tratan de hallar un conjunto de soluciones ($x_1,\ldots,x_n$), de manera que maximicen o minimicen, según el tipo del problema, una función de evaluación o función de *fitness*, $F(x_1,\ldots,x_n)$.

2.1.2 Breve introducción histórica

En los años 50 y 60 algunos científicos especialistas en inteligencia artificial, estudiaron los sistemas evolutivos con la idea de que estos podrían ser utilizados como una herramienta de optimización de cualquier tipo de problemas. La idea de todos esos sistemas era usar operadores inspirados en la selección natural para conseguir adaptar poblaciones de soluciones candidatas y así poder resolver un problema.

En los años 60, Rechenberg introdujo las llamadas “estrategias evolutivas” [2], un método que utilizó para optimizar parámetros de valor real en aplicaciones para aeronáutica, una idea que fue profundizada más tarde por Schwefel [3]. Por su parte, Fogel, Owens y Walsh, desarrollaron la “programación evolutiva” [4], técnica en la que cada solución candidata para una tarea determinada es representada como una máquina de estados finitos. La estrategia evolutiva, programación evolutiva y algoritmos genéticos forman la columna vertebral del campo del cómputo evolutivo.

Los algoritmos genéticos fueron originariamente concebidos por John Holland en los años 60 y fueron desarrollados posteriormente por Holland y sus estudiantes en la Universidad de Michigan en los años 60 y 70. Al contrario de las estrategias evolutivas y la programación evolutiva, el objetivo original de Holland no era diseñar algoritmos para resolver un problema concreto, sino estudiar el fenómeno de adaptación como ocurre en la naturaleza y desarrollar formas en la que los mecanismos de adaptación natural pudieran ser utilizados por los sistemas informáticos. En su libro *Adaptation in Natural and Artificial Systems* [5], Holland presentó el

algoritmo genético como una abstracción de la evolución biológica y propuso un enfoque teórico para la adaptación utilizando este algoritmo. El AG de Holland es un método para pasar de una población de "cromosomas" (por ejemplo, cadenas de unos y ceros) a una nueva población usando un tipo de "selección natural" junto con los operadores inspirados en la genética como el cruce, la mutación o la inversión. Cada cromosoma está compuesto de "genes" (por ejemplo, bits), y cada gen será un "alelo" en particular (por ejemplo, 0 o 1). Los operadores de selección eligen esos cromosomas en la población de manera que permitirán la reproducción, donde se elegirán aquellos individuos más aptos de la población para proporcionar más descendientes que los peor adaptados. Así se asegura una mejora en la población de una generación a otra.

La introducción de Holland de un algoritmo basado en poblaciones de individuos junto con el cruce, la mutación y la inversión fue una gran innovación en contraposición a las estrategias evolutivas de Rechenberg que usaban una población de dos individuos, un padre y un descendiente, donde el descendiente era una versión mutada del padre Por su parte, Fogel, Owens y Walsh, en los programas evolutivos, solamente utilizaban la mutación para producir la variación. Además, Holland fue el primero en intentar crear una base teórica para la evolución computacional. Este fundamento teórico ha servido como base en el desarrollo de los algoritmos genéticos posteriores.

2.1.3 Algoritmo Genético Básico

La estructura general del algoritmo genético básico sigue los siguientes pasos:

- **Inicialización.** Se inicializa el algoritmo con una población inicial de N individuos aleatorios, es decir, se crean N posibles soluciones al problema.

Una vez inicializado el algoritmo con la población inicial se procede a pasar de una generación a otra, hasta que se cumpla la condición de finalización. Para obtener los individuos de cada generación se realiza los siguientes pasos:

- **Evaluación.** Se aplica la función de *fitness* a cada uno de los individuos para comprobar la idoneidad de la solución al problema.
- **Selección.** Después de conocer la aptitud de los cromosomas, se escogen aquellos que pasarán a la siguiente generación. Los individuos más aptos de la población tendrán más posibilidades de ser escogidos. En este paso, se puede utilizar lo que se

conoce como elitismo que consiste en escoger al mejor individuo para que pase directamente a la siguiente generación.

- **Cruce.** Se combina a los individuos seleccionados de manera que los descendientes tengan información genética de los progenitores. Existe una probabilidad de cruce que debe ser alta para que se generen nuevos individuos.
- **Mutación.** Se realiza una pequeña modificación en el cromosoma de los individuos. La mutación tiene una probabilidad baja para que exista una exitosa convergencia del algoritmo.
- **Inserción.** Se sustituye la anterior población por los nuevos descendientes.

Estos últimos pasos se realizan iterativamente hasta cumplir una condición de parada, que pueden ser ciertos requisitos alcanzados o simplemente un número determinado de iteraciones.

2.1.4 Operadores Genéticos

Como se ha podido observar en la estructura del algoritmo, para obtener la población de individuos de cada generación, hay que hacer uso de Operadores Genéticos [6]. Los principales operadores son: selección, cruce y mutación, aunque hay algunos otros operadores que no serán analizados en este proyecto.

2.1.4.1 Selección

Durante la evaluación, se obtiene el valor de *fitness*. Este valor es la puntuación que se le da a una determinada solución del problema en función de sus características e idoneidad. Una vez que se tiene el valor del *fitness* para cada uno de los individuos de la población, se tiene que crear la nueva población teniendo en cuenta que los individuos más óptimos deben ser también mayoritariamente seleccionados para la utilización de su información genética como base de las siguientes generaciones.

Antes de comenzar con la selección se puede aplicar elitismo que consiste en escoger a los individuos más aptos de la generación anterior para que pasen directamente a la siguiente. Típicamente se escogen a uno o dos individuos de una población.

Se pueden destacar los siguientes métodos de selección:

Selección por Ruleta

La selección por ruleta [7] es un método propuesto por DeJong, y probablemente, uno de los más utilizados en los Algoritmos Genéticos. A cada uno de los individuos de la población se le asigna una parte proporcional de una ruleta de manera que la suma de los porcentajes debe ser la unidad. Los individuos que más se acerquen a la solución buscada tendrán una mayor porción de la ruleta, y los menos idóneos, una porción menor. La selección por ruleta se hace de la siguiente forma:

- Suma total (T) de los valores *fitness* de los individuos.

- Se obtiene un valor aleatorio r $\in$ (0,1), que multiplicado por T nos dará el valor a buscar. El individuo cuya suma acumulada de los valores *fitness* supere este valor, es el elegido.

Selección por Torneo

La selección por torneo es un método basado en la comparación de dos o más individuos de la población escogiendo el mejor. Se realiza de la siguiente forma:

- Se escogen dos o más individuos de la población y se calcula la función de *fitness* de cada uno de ellos.

- El que mejor *fitness* de los dos presente (es decir, ganador del torneo) será el individuo elegido.

Existen dos versiones de selección mediante torneo:

- **Determinista.** En esta versión se selecciona al azar un número de individuos, que generalmente es dos. De entre los individuos escogidos, se selecciona el más apto.

- **Probabilística.** Esta versión únicamente se diferencia de la anterior, en el paso de selección del individuo más apto. Para el caso de haber seleccionado dos individuos, se escoge un valor aleatorio entre 0 y 1, y se compara con un valor p que se habrá fijado para todo el proceso. Si es mayor que p, se escoge el individuo más apto, y en caso contrario, se escoge el menos apto.

2.1.4.2 Cruce

El operador genético de cruce es el encargado de combinar la información genética para crear la descendencia de los individuos ya elegidos previamente al azar. Así, los descendientes tendrán información genética de ambos progenitores, pero puede que no sea en la misma proporción. El intercambio genético se puede llevar a cabo de distintas formas, aunque los más utilizados son los siguientes:

Cruce de un punto

Es la más sencilla de todas las técnicas de cruce. Una vez seleccionados dos individuos se cortan sus cromosomas por un punto seleccionado aleatoriamente de manera que los descendientes intercambiarán información de cada padre, tal y como puede apreciarse a modo de ejemplo en la figura 2.1.

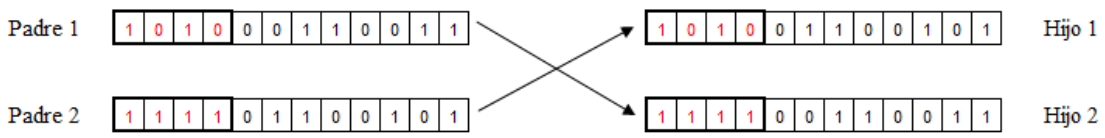

Figura 2.1: Cruce de un punto.

Cruce de dos puntos

Es una generalización del cruce de un punto. Al igual que en el caso anterior, una vez seleccionados los dos individuos, se realizan dos cortes en el cromosoma, de manera que ninguno de los puntos de corte coincida con el extremo de los cromosomas para asegurar que se originen tres partes. Para generar la descendencia se escoge la parte central de uno de los padres y los laterales del otro padre tal y como se ilustra en la figura 2.2.

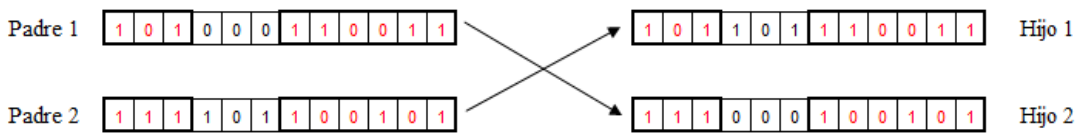

Figura 2.2: Cruce de dos puntos.

Cruce Uniforme

El cruce uniforme es un método de cruce que difiere conceptualmente de los anteriores. En este caso, las posibilidades de que un gen descienda de un padre o de otro son las mismas.

En esta técnica se utiliza una máscara de cruce con valores binarios de forma que si en la máscara hay un 1, el descendiente toma ese gen del primer padre, y si hay un 0, se toma del segundo padre.

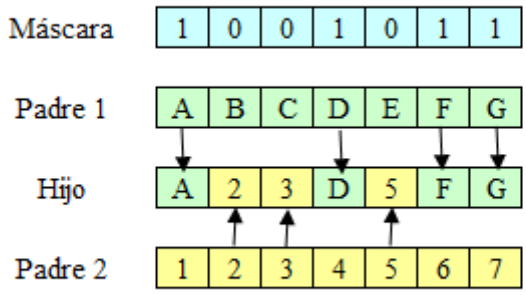

Figura 2.3: Cruce uniforme.

2.1.4.3 Mutación

La mutación de un individuo consiste en variar aleatoriamente uno o varios de los genes de un cromosoma de un individuo. Así, se imita el comportamiento de la naturaleza introduciendo en la descendencia algún tipo de error que, por lo general, no tiene una aparente transcendencia en la transmisión de la carga genética de padres a hijos.

El comportamiento de este operador depende de la Probabilidad de Mutación, p_m, que habitualmente tiene un valor máximo de un 1%. La mutación más utilizada es el reemplazo aleatorio, que consiste en variar aleatoriamente un gen de un cromosoma. Si se trabaja con codificaciones binarias consistirá en cambiar unos por ceros, y viceversa.

2.2. Algoritmo Basado en Enjambres de Partículas

2.2.1 Introducción

El Algoritmo Basado en Enjambres de Partículas o *Particle Swarm Optimization* (PSO) [8], es una técnica de optimización metaheurística desarrollada por primera vez por Russell Eberhart y James Kennedy en 1995. Está basado en el comportamiento y establecimiento de normas sociales en grupos de individuos, en este caso en bandadas de pájaros o bancos de peces, y luego extendido al comportamiento humano. El objetivo del algoritmo es simular el comportamiento de un enjambre, en el que cada individuo realiza una serie de movimientos aleatorios pero que tienden a mantenerse unidos.

El algoritmo utiliza partículas, que representan a cada uno de los individuos del enjambre, que se mueven a través de una región siguiendo a las partículas óptimas de cada iteración e intentando dirigir a todo el conjunto hacia la región óptima del espacio de soluciones. Cada una de estas partículas posee una situación exacta dentro del grupo representando una posible solución al problema, y además, poseen información de la mejor posición por la que han pasado anteriormente y el mejor valor obtenido de todas las partículas que conforman el enjambre.

En los últimos años, el algoritmo PSO ha sido empleado con éxito para optimizar problemas de diversa índole [9]. A diferencia de otros métodos ofrece rápidos resultados con un bajo coste computacional

2.2.2 Breve historia

PSO se basa en la cualidad más importante que caracteriza a los individuos sociales: la auto-organización. La auto-organización [10] es un mecanismo dinámico a través del cual se crean estructuras a nivel global de un sistema a partir interacciones entre elementos de bajo nivel. Las leyes que detallan las interacciones entre las unidades constituyentes del sistema son ejecutadas en base a información puramente local, sin hacer referencia alguna al modelo global. Esta auto-organización dota a estos individuos de la capacidad de adaptarse al medio, es decir, son capaces de encontrar fuentes de alimentación y evitar a los depredadores mediante el intercambio de información.

Las conductas sociales que inspiraron inicialmente este algoritmo fueron las bandadas de pájaros, los bancos de peces o los enjambres de abejas [11]. Las primeras investigaciones y simulaciones informáticas realizadas por Eberhart y Kennedy interpretaban el movimiento de las aves en las bandadas y su conducta dentro del grupo.

Axelrod en 1980 publicó *La Complejidad de la Cooperación*, en donde proponía un modelo computacional para la diseminación de la cultura [12]. Esta simulación ya contenía algunos de los principios que se utilizan en el algoritmo PSO sobre los efectos de la interacción social. Según esta teoría, un pequeño número de principios puede formar un sistema artificial que simule la compleja sociedad humana.

Heppner en 1980, llevó a cabo un nuevo experimento que consistía en grabar grupos de pájaros con cámaras [13]. Con estas grabaciones comprendió como se podía organizar el grupo para cambiar de dirección de forma repentina, o comprobar que no existe un líder dentro de la bandada sino que el grupo mantiene cierto equilibrio sin control central. Lo más interesante de las simulaciones que realizó Heppner fue descubrir que pájaros virtuales (dotados del comportamiento de las aves reales) podían atraer a la bandada a un área denominada *roosting*.

Bibb Latané en 1981 postuló la *Teoría del Impacto Social Dinámico* [14], donde extendía las predicciones del impacto social para demostrar que el comportamiento de los individuos se puede explicar en relación con las propiedades de auto-organización del sistema social que comprende. De forma resumida, los individuos se influencian unos a otros, creándose similitudes entre ellos. Las simulaciones realizadas por Latané demostraron que existían características de adaptación colectiva, relacionando la inteligencia del enjambre con el fenómeno humano.

Ya en 1987, Reynolds intentó simular los movimientos que realizan las aves dentro de las bandadas de pájaros [15]. Asumiendo que las bandadas estaban dirigidas por fuerzas locales, partió de tres reglas sencillas para realizar las simulaciones: evitar el choque, igualar la velocidad y centrar la bandada. Esto es:

- Separarse antes de colisionar uno con otro.
- Intentar llevar la misma velocidad que el resto de vecinos del grupo.
- Intentar moverse hacia el centro de la bandada a medida que lo perciban.

Implementando estas sencillas reglas, las simulaciones de Reynolds resultaron muy exitosas. Trabajó con grupos de aves, simulando vuelos a través del espacio e incluyendo

obstáculos en su camino. De esta manera, comprobó que aunque la bandada se separara en algún momento, volvía a reunirse.

2.2.3 PSO Básico

La estructura del algoritmo PSO Básico se basa en la definición y actualización de la posición de las partículas y su velocidad, que vienen expresadas en las ecuaciones 2.1 y 2.2.

$$v_i^{t+1} = v_i^t + \varphi_1 U_1(0,1) * (p_i - x_i^t) + \varphi_2 U_2(0,1) * (s - x_i^t) \tag{2.1}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{2.2}$$

donde φ_1 y φ_2 son dos constantes llamadas coeficientes de aceleración cognitiva y social respectivamente, $U_1(0,1)$ y $U_2(0,1)$ son dos números aleatorios positivos obtenidos de una distribución uniforme con un límite superior determinado.

Por otro lado, en las ecuaciones aparecen dos vectores con los que se calcula la velocidad de la iteración actual, que son el vector p_i, mejor posición por la que ha pasado la partícula (*local_best*), y *s*, mejor valor obtenido de todas las partículas que conforman el enjambre (*global_best*).

Los pasos a seguir en la ejecución del algoritmo son los siguientes:

1. Se inicializan las posiciones de las partículas x_i y las velocidades v_i de las partículas aleatoriamente.
2. Se evalúa la función de *fitness* para cada una de las partículas.
3. Se actualizan inicialmente los vectores *local_best* y *global_best*.

A partir de este momento, para cada una de las iteraciones, y dentro de cada iteración para cada una de las partículas i, se siguen los siguientes pasos:

1. Actualización de la velocidad de la partícula *i*.
2. Se actualiza la posición de la partícula.
3. Evaluación de la función de *fitness* para cada partícula i, de forma que si dicho valor de es menor que el almacenado en las variables *local_best* y *global_best*, se actualizan.

2.2.4 Otras Variantes del Algoritmo

El algoritmo PSO Básico tiene distintas variantes [16], que brevemente, se describen a continuación.

2.2.4.1 PSO Canónico

Clerc y Kennedy introdujeron un factor de restricción en el PSO [17] para controlar las propiedades de convergencia de las partículas. El factor de restricción (ecuación 2.4) se añade en la expresión de las velocidades tal y como se muestra en la ecuación 2.3.

$$v_i^{t+1} = \chi(v_i^t + \varphi_1 U_1(0,1) * (p_i - x_i^t) + \varphi_2 U_2(0,1) * (s - x_i^t)) \tag{2.3}$$

$$\chi = 2k / \left(\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right| \right) \tag{2.4}$$

donde $k \in [0,1], \varphi = \varphi_1 + \varphi_2$ y $\varphi > 4$. Normalmente, *k* toma valor 1 y las variables φ_1 y φ_2 toman el valor 2.05, dando como resultado χ = 0.729. Esta variante ha sido muy utilizada y es conocida como PSO Canónico.

2.2.4.2 PSO con Inercia Variable en el Tiempo

Shi y Eberhart introdujeron la idea de una inercia variable en el tiempo [18]. La idea era controlar la diversificación-intensificación de la conducta del PSO original. La nueva ecuación para la velocidad es la siguiente:

$$v_i^{t+1} = \varpi(t) v_i^t + \varphi_1 U_1(0,1) * (p_i - x_i^t) + \varphi_2 U_2(0,1) * (s - x_i^t) \tag{2.5}$$

donde $\omega(t)$ es la inercia variable en el tiempo, que normalmente es linealmente adaptada desde un valor inicial hasta un valor final. En muchos casos, φ_1 y φ_2 toman valor 2.

2.2.4.3 PSO con Inercia Estocástica

Eberhart y Shi propusieron otra variante en la cual la inercia es seleccionada aleatoriamente acorde a una distribución uniforme en el rango [0.5, 1] [19]. Este rango fue inspirado por el factor de restricción (ecuación 2.4) de Clerc y Kennedy. En esta versión, los coeficientes de aceleración toman el valor 1.494 como resultado de la multiplicación de $\chi \cdot \varphi_{1,2}$.

2.2.4.4 Fully Informed PSO

En el FIPS (*Fully Informed Particle Swarm*) [20] fue propuesto por Mendes, y en él, una partícula utiliza toda la información de todas sus partículas vecinas. Esta variante está basada en el hecho de que, el factor de restricción de Clero y Kennedy, no implica que el valor φ deba dividirse entre dos coeficientes.

En este caso, $\varphi_k = \varphi / |N| \quad \forall k \in N$ donde N es el barrio de la partícula. Como resultado, la nueva velocidad vendrá dada por la expresión de la ecuación 2.6:

$$v_i^{t+1} = \chi \left[v_i^t + \sum_{k \in N} \varphi_k W(k) U_k(0,1) * (p_k - x_i^t) \right] \tag{2.6}$$

donde W(k) es una función de ponderación.

2.3. Algoritmo Basado en Colonias de Hormigas

2.3.1 Introducción

Los algoritmos de las hormigas (Ant Colony Algorithm) [21] fueron propuestos por Marco Dorigo y sus colaboradores como un enfoque multi-agente para resolver problemas de optimización combinatoria, como el problema del viajante y el problema de asignación cuadrática.

Los algoritmos de las hormigas están inspirados en el comportamiento de las colonias reales de hormigas. Las hormigas son insectos sociales, es decir, viven en sociedad y su comportamiento se dirige en mayor medida a la supervivencia de ésta. Los insectos han llamado la atención de muchos científicos debido a la complejidad estructural que pueden alcanzar sus colonias, especialmente cuando se compara la relativa simplicidad de cada uno de los individuos. Un comportamiento importante e interesante de las colonias de hormigas es la alimentación y sobre todo, su capacidad de encontrar el camino más corto entre la fuente de alimento y su nido.

Mientras caminan entre la fuente de alimento y su nido, las hormigas desprenden una sustancia llamada feromona, dejando a su camino un rastro de feromonas. Las hormigas detectan esta sustancia y toman sus decisiones de movimiento en función de la concentración de la misma. De esta manera, las feromonas desprendidas permiten a las hormigas encontrar el camino de vuelta a la fuente de alimento o al nido, pero también son utilizadas por otras hormigas para encontrar las fuentes de alimento encontradas por sus compañeras de nido.

Se ha demostrado experimentalmente que este rastro de feromonas, una vez empleado por una colonia de hormigas, puede dar lugar a la aparición de las rutas más cortas. Es decir, cuando existen varios caminos desde el nido a una fuente de alimento, una colonia de hormigas puede ser capaz de explotar las rutas de feromonas que dejan las hormigas individuales a su paso, para descubrir el camino más corto desde el nido a la fuente de alimento y la vuelta.

Para estudiar el comportamiento en condiciones controladas de las hormigas, Deneubourg creó el experimento llamado *Puente Binario* [22]. Consistía en separar el nido de una colonia de hormigas y la fuente de alimento con un puente doble en el que cada rama tiene la misma longitud. Las hormigas se dejan libres para moverse entre el nido y la fuente de alimento, y se

observa a lo largo del tiempo el porcentaje de hormigas que optan por una u otra rama. El resultado es que tras una primera fase transitoria en las hormigas se reparten entre las dos ramas, las hormigas tienden a converger en un mismo camino.

En el experimento anterior, inicialmente no hay feromonas en ninguna de las dos ramas, por lo que las hormigas seleccionan con la misma probabilidad uno u otro camino. Sin embargo, las fluctuaciones aleatorias después de una fase transitoria inicial causan más aleatoriedad a la hora de que las hormigas seleccionen una rama u otra. Las hormigas depositan feromonas mientras caminan, por lo que el mayor número de hormigas en una de las ramas, determinará una mayor cantidad de feromona en ella, lo que estimula a más hormigas a elegir dicha rama.

El modelo probabilístico que describe este fenómeno es el siguiente. En primer lugar, se supone que la cantidad de feromona en una rama es proporcional al número de hormigas que utilizó la rama en el pasado. Este supuesto implica que la evaporación de feromonas no se tiene en cuenta, ya que si un experimento suele durar aproximadamente una hora la cantidad de feromona que se evaporó en este periodo es insignificante. En el modelo, la probabilidad de elegir una rama en un momento determinado depende de la cantidad total de feromonas en la rama, que a su vez, es proporcional al número de hormigas que ha pasado por esa rama hasta ese momento. Para mayor precisión, U_m y L_m, son el número de hormigas que han utilizado la rama superior e inferior respectivamente después de cruzar *m* hormigas el puente, con $U_m + L_m = m$. La probabilidad de $P_U(m)$ con la que, la *(m+1)*-ésima hormiga, elige la rama superior es:

$$P_U(m) = \frac{(U_m + k)^h}{(U_m + k)^h + (L_m + k)^h} \tag{2.7}$$

Mientras que la probabilidad $P_L(m)$ de que se elija la rama inferior es $P_L(m) = 1 - P_U(m)$. Esta manera de seleccionar mediante probabilidades una rama u otra se obtuvo de realizar experimentos en la ruta de seguimiento; los parámetros *h* y *k*, permiten ajustar el modelo para los datos de experimentación. La dinámica de selección de las hormigas se desprende de la ecuación: $U_{m+1} = U_m + 1$ si $\psi \le P_U, U_{m+1} = U_m$ donde ψ es una variable aleatoria uniformemente distribuida en el intervalo *[0,1]*.

Mediante un proceso de Monte Carlo se comprobó la correspondencia de este modelo con el real, y los resultados de las simulaciones establecían que para $k \approx 20$ y $h \approx 2$ [23], los experimentos se correspondían con las hormigas reales.

Es fácil modificar el experimento anterior para el caso en que las ramas del puente sean de diferente longitud (figura 2.4), y extender el modelo a la ecuación 2.7 para que pueda describir esta nueva situación. En este caso, la rama menor se seleccionará con mayor frecuencia, ya que las primeras hormigas que llegan a la fuente de alimento, son las que llegaron a través de la rama más corta, de modo que cuando estas hormigas inicien el viaje de regreso, habrá mayor cantidad de feromonas en la rama corta que en la rama más larga, y estimulará a las hormigas a elegir la rama más corta.

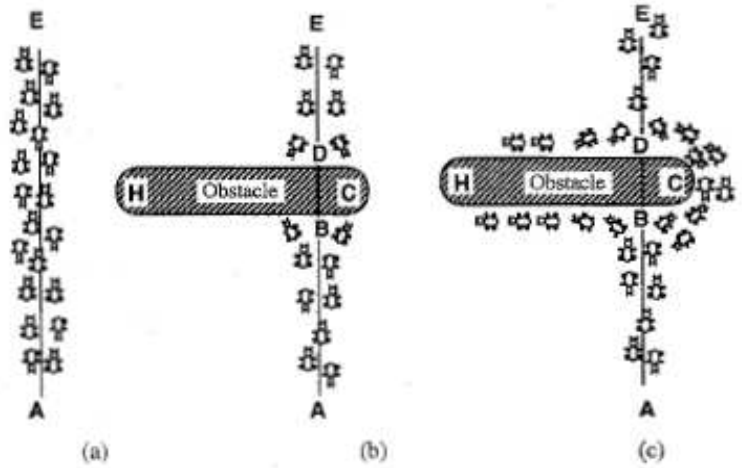

Figura 2.4: Bifurcación de Caminos

Aunque una sola hormiga sea capaz de construir una solución (encontrar un camino entre el nido y el depósito de alimentos), es sólo la colonia de hormigas quién pueden encontrar el camino más corto. También es interesante observar que las hormigas pueden realizar este comportamiento específico utilizando una forma simple de comunicación indirecta como es la feromona. Este tipo de mecanismos se conoce como *stigmergy*.

Grasse, en su trabajo *Bellicositermes Natalensis* and *Cubitermes*, definió *stigmergy* [24] como “estimulación de los trabajadores para el desempeño de sus logros”. Observó que los insectos son capaces de responder a los llamados “estímulos significativos” que activan una reacción genéticamente codificada. En los insectos sociales, por ejemplo termitas y hormigas, los efectos de estas reacciones pueden actuar como nuevos estímulos significativos, tanto para el insecto que los produce como para otros insectos de la colonia. Este hecho genera una forma de coordinación de las actividades y se puede interpretar como una forma de comunicación indirecta.

Lo que caracteriza a *stigmergy* de otros métodos de comunicación es (i) la naturaleza física de la información generada por los insectos, que corresponde a la modificación de los estados

físicos visitados por los insectos, y (ii) la naturaleza local de la información publicada, a la que sólo pueden acceder los insectos que visitan el estado en que fue puesto en libertad.

En consecuencia, en este trabajo se toma la postura de que es posible hablar de comunicación *stigmergy* cada vez que hay una "comunicación indirecta mediada por modificaciones físicas de los estados a los que sólo son accesibles a nivel local los agentes".

El modelo de comunicación stigmergy está inspirado en el comportamiento de alimentación de las hormigas, en particular es un modelo interesante para sistemas artificiales multi-agente aplicados a la solución de problemas complicados de optimización. De hecho, las características mencionadas de stigmergy se pueden ampliar fácilmente a los agentes artificiales por (i) asociar al problema variables de estado apropiadas a los estados, y (ii) dando a los agentes artificiales acceso local a los valores de esas variables.

Otro aspecto importante del comportamiento de las hormigas reales, que es explotado por las artificiales, es el acoplamiento entre la realimentación positiva (mecanismo autocatalítico) y la evaluación implícita de soluciones. Mediante la evaluación implícita de soluciones se entiende el hecho de que los caminos más cortos (que corresponden a las soluciones de menos coste en las hormigas artificiales) se completarán antes que los más largos, y por lo tanto, recibirán un refuerzo de feromonas rápidamente.

2.3.2 Enfoque de la Optimización con el Algoritmo de Colonias de Hormigas

En la Optimización meta-heurística basada en Colonias de Hormigas (ACO), una colonia de hormigas artificial coopera para encontrar soluciones óptimas a los difíciles problemas de optimización discreta. La cooperación es un componente clave del diseño de algoritmos ACO: la elección es asignar los recursos computacionales a un conjunto de agentes relativamente simples (hormigas artificiales) que se comunican indirectamente por *stigmergy*.

Las hormigas artificiales tienen una naturaleza doble. Por un lado, son una abstracción de los rasgos de comportamiento de las hormigas reales, que se encuentran realizando la búsqueda del camino más corto, comportamiento observado en las colonias de hormigas reales. Por otra parte, se han enriquecido algunas capacidades que no se encuentran en la naturaleza. Es razonable otorgar algunas capacidades a las hormigas artificiales que, aunque no se correspondan con las capacidades de las hormigas reales, hagan que sean más eficaces y eficientes.

2.3.2.1. Similitudes y diferencias con las hormigas reales

La mayoría de las ideas de ACO derivan de las hormigas reales. En particular, el uso de (i) una colonia de individuos colaboradores o cooperantes, (ii) una ruta (artificial) de feromonas para comunicación local, (iii) una secuencia de movimientos para encontrar el camino más corto, y (iv) una política de decisión estocástica usando información local.

1. Colonia de individuos cooperantes. Las colonias de hormigas reales están compuestas de entidades cooperantes a nivel global para encontrar una buena solución al problema en cuestión. Aunque la complejidad de cada hormiga artificial es tal que puede crear una solución viable (como una verdadera hormiga puede encontrar un camino entre el nido y el alimento), las soluciones de mejor calidad son el resultado de la cooperación entre los individuos de toda la colonia. Las hormigas cooperan por medio de la información que leen/escriben en los estados (nodos) que visitan, como se explica en el siguiente punto.

2. Ruta de Feromonas y Stigmergy. Las hormigas artificiales modifican algunos aspectos de su entorno como las hormigas reales. Mientras que las hormigas reales desprenden una sustancia química en los lugares por donde pasan, las hormigas artificiales modifican información numérica almacenada en los estados (nodos) que visitan. Esta información tiene en cuenta el rendimiento/historia actual de la hormiga y puede ser leída/escrita por algunas de las hormigas que accedan al estado. Su efecto principal es cambiar la forma en que una hormiga percibe el medio del problema en función de los antecedentes de toda la colonia de hormigas. Por lo general, en los algoritmos ACO existe un mecanismo de evaporación, similar a la evaporación real de feromona, que modifica la cantidad de feromona artificial a lo largo del tiempo, y permite que la colonia de hormigas olvide poco a poco su historia pasada.

3. Ruta óptima y movimientos locales. Las hormigas artificiales y resales comparten una tarea común: encontrar el camino más corto (coste mínimo) desde el origen (nido) hasta el destino (comida). Las hormigas reales simplemente caminan por los terrenos adyacentes, que es lo mismo que hacen las hormigas artificiales.

4. Política estocástica y a corto plazo de transición de estados. Las hormigas artificiales, al igual que las reales, construyen soluciones aplicando una política de decisión probabilística para desplazarse por los estados adyacentes.

Como hemos dicho, las hormigas artificiales tienen también algunas características que no se encuentran en las hormigas reales:

- Las hormigas artificiales viven en un mundo discreto y sus movimientos consisten en transiciones entre estados discretos.
- Las hormigas artificiales no tienen un estado interno. Este estado privado contiene la memoria de las acciones pasadas por la hormiga.
- La sincronización en las hormigas artificiales es un problema de dependencia y habitualmente no se refleja en el comportamiento de las hormigas reales. Por ejemplo, en muchos casos, las hormigas artificiales actualizan los rastros de feromona solamente después de haber generado una solución.
- Para mejorar la eficiencia general del sistema, los algoritmos ACO pueden enriquecer algunas capacidades como "visión de futuro" (*lookahead*), optimización local, *backtracking*, etc., que no se encuentran en las hormigas reales. En muchas implementaciones las hormigas se han mezclado con procedimientos de optimización local.

2.3.3 Algoritmo Basado en Colonias de Hormigas Básico

La estructura general del algoritmo ACO básico, es la siguiente:

1. **Inicialización.** Se inicializa el algoritmo con una población inicial de N hormigas.
2. **Evaluación.** En este paso, se aplica la función de *fitness* a cada uno de los nodos y se actualiza el nivel de feromona que inicialmente se va a colocar en cada uno de los nodos.

Una vez que se tienen inicializadas las variables necesarias y evaluada la primera colonia inicial del algoritmo, se procede a ejecutar el Ciclo de vida de cada hormiga:

3. Se comprueban los nodos disponibles y no visitados con anterioridad.
4. Se aplica la política de decisión para elegir el siguiente nodo.
5. Se mueven las hormigas hacia el nodo seleccionado, actualizando el histórico de estados ya visitados.
6. Se deposita el nivel de feromona correspondiente a la solución obtenida
7. Si no se han completado todas las rutas para las hormigas, se vuelve al paso 3.

Por último se procede a la actualización de la feromona y la evaluación:

8. Se evalúan las soluciones finales de cada hormiga, añadiendo la feromona correspondiente.
9. Se produce la evaporación de cierta cantidad de feromona de todos los nodos visitados.

Si no se ha llegado a la condición de parada o se han ejecutado todas las iteraciones se vuelve de nuevo al paso 1.

Capítulo 3. Síntesis de Arrays

En este capítulo, se detallan los resultados obtenidos tras optimizar arrays monofrecuencia con los algoritmos de optimización descritos en el capítulo anterior: Algoritmo Genético, Algoritmo Basado en Enjambres y Algoritmo Basado en Colonias de Hormigas. Así mismo, se describen las configuraciones que se han utilizado de cada uno de ellos, tanto los parámetros como las técnicas. Todas las implementaciones realizadas en cada uno de dichos algoritmos son de carácter discreto.

3.1. Teoría de Arrays

Se define como array al conjunto de *N* antenas iguales que radian o reciben simultáneamente [25]. El diagrama de radiación del array se obtiene como la interferencia de los campos radiados por cada una de las antenas.

Como se ve en la ecuación (3.1), el diagrama de campo radiado por la agrupación es igual al producto del diagrama de la antena básica, $\vec{E}_0(\vec{r})$, multiplicado por un factor que tiene en cuenta la interferencia de las *N* ondas generadas por las *N* antenas.

$$\vec{E}(\vec{r}) = \vec{E}_0(\vec{r}) \sum_{n=0}^{N-1} a_n e^{jn\psi} \tag{3.1}$$

Este factor depende únicamente de la separación entre los elementos de la agrupación, de la alimentación y de la frecuencia de trabajo, y se denomina factor de array (FA),

$$FA(\psi) = \sum_{n=0}^{N-1} a_n e^{jn\psi} \tag{3.2}$$

El FA tiene las siguientes propiedades:

- Es una función periódica del ángulo ψ, de periodo 2π, tal que los coeficientes de su serie de Fourier son los coeficientes de alimentación a_n.
- El factor de array es la transformada de Fourier de la secuencia discreta de los coeficientes de la alimentación, a_n.
- Si los coeficientes de la alimentación a_n son reales y positivos, el máximo del factor de la agrupación se encuentra en el origen $\psi = 0$.

  El ángulo θ, que indica la dirección de radiación en el espacio, sólo toma valores reales entre 0 y π, los cuales corresponden a un intervalo de variación de ψ: $\psi \in [-kd + \alpha, kd + \alpha]$. Solamente la parte de FA comprendida en el intervalo anterior pertenece al diagrama de radiación. A este intervalo se le llama el margen visible. La longitud del margen visible es *2kd* y está centrado en $\psi = \alpha$, de forma que su tamaño es proporcional al espaciado del array normalizado con respecto a la longitud de onda, y su posición en el eje ψ varía con la fase progresiva.

- Para coeficientes de alimentación reales y positivos, cuando el margen visible incluye el origen $\psi = 0$, según $|\alpha| \leq kd$, el máximo del diagrama de radiación se encuentra en la dirección del espacio

$$\psi = kd\cos\theta_{\max} + \alpha = 0$$

$$\theta_{\max} = \arccos\left(-\frac{\alpha}{kd}\right), |\alpha| \leq kd$$

  Por lo tanto, puede controlarse la dirección del máximo de radiación variando la fase progresiva.

- Como el factor de array es periódico con periodo 2π, si el máximo está en ψ_{max} existen máximos periódicos en los múltiplos enteros de 2π, $\psi = 2m\pi + \psi_{\max}$. Cuando estos máximos periódicos se encuentran dentro del margen visible, si $kd + \alpha \geq 2\pi$ o $-kd + \alpha \leq 2\pi$, aparecen múltiples máximos de radiación en el espacio real, denominados lóbulos de difracción (*grating lobes*).

3.2. Thinned Array

El concepto de *Thinned Array* [26] significa colocar una serie de elementos radiantes dispuestos en forma de array con espaciado no-uniforme o aperiódico para conseguir la amplitud deseada a lo largo de la apertura. Una posible implementación de *thinned array* consiste en situar elementos una distancia fija, y cuya amplitud puede ser 1 ó 0 (es decir, son o no alimentados) [26]. Este tipo de implementación tiene 2^Q combinaciones posibles, donde Q es el número de elementos del array. Sin embargo, si el array es simétrico el número de posibilidades es sustancialmente más reducido.

Un *thinned array* grande, que es necesario para conseguir un nivel muy bajo de lóbulos secundarios, requiere la comprobación de un número amplio de posibilidades hasta encontrar la mejor distribución de los elementos. Comprobar exhaustivamente todas las combinaciones posibles de organización de los elementos sólo es posible si el array es reducido. Métodos de optimización como el Método de Powell o el Gradiente Conjugado, no se adaptan bien para los *thinned array* ya que solamente pueden optimizar variables continuas y suelen quedar atrapados en mínimos locales. Además, estos métodos fueron desarrollados para parámetros continuos, mientras que los *thinned arrays* implican elementos discretos.

El propósito de este capítulo es la de encontrar un *thinned* array que proporcione el mínimo nivel de lóbulos secundarios posible, mediante el uso de algoritmos de búsqueda global tales como genéticos, basados en enjambre de partículas o colonias de hormigas.

3.3. Función de Fitness

Para evaluar los algoritmos de optimización se utiliza habitualmente una función de evaluación o función de *fitness*, que indica la idoneidad de la solución alcanzada. La función de *fitness* depende de los resultados que se deseen obtener. Algunos ejemplos de funciones de evaluación utilizadas en este tipo de algoritmos pueden ser:

- Envolvente del diagrama de radiación.
- Relación de lóbulo principal a secundario
- Dirección de máxima radiación
- Ancho de haz

En este caso, se ha elegido como función de evaluación la relación de nivel de lóbulos secundarios en el diagrama de radiación. Para ello, se ha de calcular el factor de array, que puede ser obtenido a partir de la siguiente expresión:

$$S(\theta) = \frac{1}{N} \sum_{n=1}^{N} a_n \cos[kd(n-0.5)u] \tag{3.3}$$

donde:

- u $\sin(\theta)$
- θ ángulo
- d distancia entre elementos
- k $2\pi/\lambda$
- λ longitud de onda
- a_n amplitud de los elementos que forman el array
- 2N número de elementos en el array

Para una solución concreta, es decir, un conjunto $\{a_n\}$ cuyos valores sólo pueden ser 1 ó 0 (antena activada o carga adaptada), su función de *fitness* vendrá dada por la diferencia entre el máximo de radiación que se encuentra en u = 0 y el máximo valor de lóbulos secundarios.

Para el cálculo de la función de *fitness* se sigue el siguiente procedimiento:

1. La función de *fitness* que se ha implementado recibe un array con dimensión *N*, el cual se aplica en la fórmula anterior.

2. Una vez que se ha obtenido un valor de *S* para cada uno de los valores de *u* que se han implementado (siendo *u* un array con valores en el intervalo de -1 a 1), se busca el máximo valor y se normaliza. De esta forma, se tiene un array con los valores del factor de array normalizados.
3. Finalmente, de entre los valores obtenidos en el paso anterior se busca el máximo exceptuando el máximo de radiación que se encuentra en u=0. Y ese será el valor de *fitness* asociado al array que se ha recibido como parámetro.

3.4. Implementación del Algoritmo Genético

El Algoritmo Genético es el primer algoritmo de optimización que se va a emplear. Este algoritmo está basado en los mismos métodos de la evaluación biológica, consiguiendo la diversidad genética mediante mutaciones y la reproducción sexual, o lo que es lo mismo, aplicando los distintos operadores genéticos que se deben de configurar antes de ejecutar el algoritmo.

En todos los casos a evaluar, aplicando los distintos algoritmos genéticos implementados, van a comenzar teniendo una población de 50 individuos con cromosomas de 60 bits (lo que indica que se optimizarán arrays de 120 elementos) cada uno. Esta elección otorgará la suficiente diversidad a la población, de manera que, inicialmente, los individuos diferirán mucho unos de otros, y conforme vayan dándose los distintos cruces, formándose las nuevas generaciones (iteraciones), estos individuos serán cada vez más parecidos genéticamente, consiguiendo la convergencia en una solución óptima para el problema. Obviamente, cuanto más grande sea la población, más tiempo tardará en hallar una solución.

Los parámetros que se han utilizado en las simulaciones realizadas, son los siguientes:

- **Selección.** Se han desarrollado dos versiones del AG diferenciadas por el tipo de selección, selección por Torneo y selección por Ruleta.
- **Mutación.** Este operador, varía aleatoriamente uno o dos bits del total que se compone el cromosoma. Por tanto, se aplica una mutación aproximada del 1%.
- **Cruce.** Se utiliza el **Cruce de un punto**, detallado en el capítulo anterior, combinando los cromosomas de los individuos progenitores y obteniendo los nuevos individuos que contienen información genética de la generación parental.

Como se ha dicho, se han implementado dos versiones del AG diferenciadas por el tipo de selección utilizada. Por un lado, la selección por torneo, la cual se implementa realizando una selección aleatoria de individuos de la población, habitualmente dos, y escogiendo de entre ellos el que mejor solución va a ofrecer. Por otro lado, la selección por Ruleta es mucho menos restrictiva, aunque el mejor de los individuos tiene más posibilidades de ser escogido (ocupará más área en la ruleta al generar una mejor solución), también puede ser elegido uno que sea menos apto, haciendo así que se ralentice la convergencia del algoritmo.

En la figura 3.1 se muestra el diagrama de flujo que sigue nuestro AG.

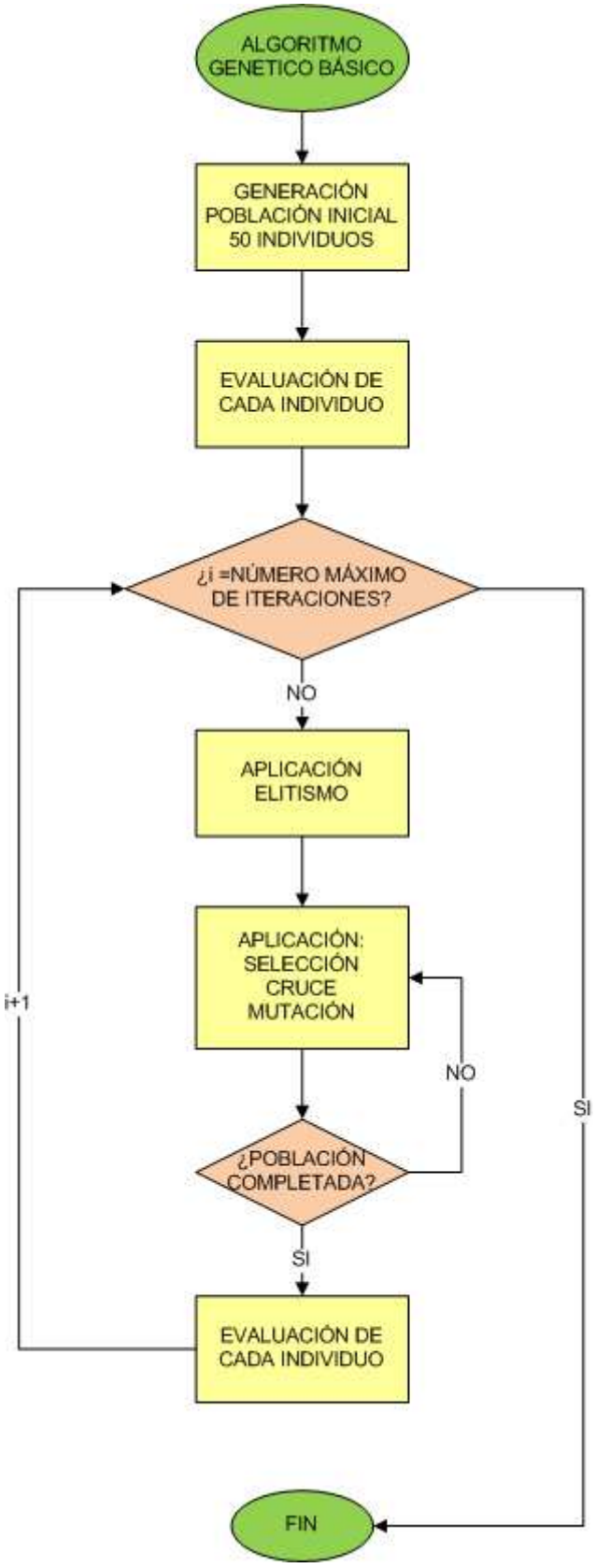

Figura 3.1: Diagrama de Flujo de un Algoritmo Genético Básico

3.4.1 Resultados Simulaciones Realizadas

3.4.1.1 Algoritmo Genético con Selección por Ruleta

Los pasos que se seguirán en el AG con selección por Ruleta son los siguientes:

1. Se genera una población aleatoria de 50 individuos, formados por cromosomas de 60 bits.
2. Se evalúan cada una de las posibles soluciones (cada uno de los cromosomas de los individuos), obteniendo el que inicialmente es el individuo más sobresaliente (la mejor solución de esta generación) y se guarda en una variable llamada *best_fitness*.
3. Ciclo de cada una de las generaciones/iteraciones:
   - Se aplica elitismo a la población, de manera que se escogen los dos individuos más aptos de la generación anterior. Estos individuos, pasan directamente a la siguiente generación garantizando la convergencia del algoritmo.
   - Se realiza la selección por Ruleta, donde se escogen dos individuos aleatoriamente. A estos individuos, se les aplica un cruce y la mutación obteniendo dos soluciones, que serán dos individuos para la nueva población.
   - Se repite el paso anterior hasta completar la población de 50 individuos, teniendo en cuenta que ya teníamos dos individuos, que pertenecían a la anterior generación.
4. Una vez que la nueva generación ha sido completada, se evalúan de nuevo las soluciones obtenidas, comparando cada uno de los valores de *fitness* con la variable *best_fitness*, y guardando la solución que la mejore.
5. Si no se ha llegado al máximo de iteraciones propuesto o a la condición de parada, se vuelve al paso 3 para comenzar de nuevo con la generación de una nueva población.

Tras realizar pruebas para un AG con selección por ruleta con un bit de mutación, se obtuvieron los resultados mostrados en las figuras 3.2, 3.3 y 3.4. En primer lugar, en la figura 3.2 se observa el máximo valor de los lóbulos secundarios del diagrama de radiación a lo largo de las generaciones donde se comprueba cómo, a medida que van pasando las iteraciones, este nivel es descendente. Este nivel de lóbulos secundarios, que se muestra en cada iteración, pertenece al mejor individuo de cada generación, es decir, al individuo cuya solución es la más idónea para el problema propuesto.

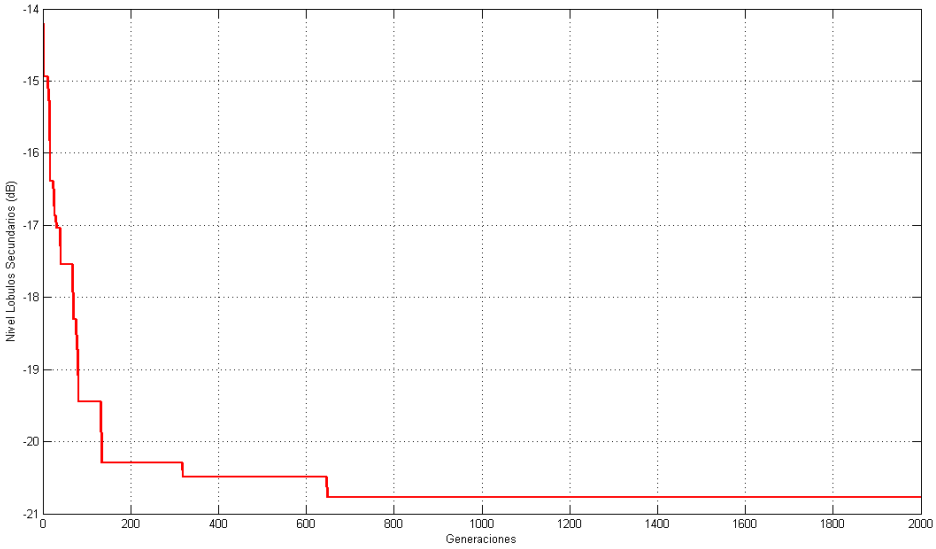

Figura 3.2: Relación de lóbulo principal a secundario con AG con Selección por Ruleta.

Como se puede ver en la figura 3.2, inicialmente la primera generación cuenta en su diagrama de radiación con un nivel de -14dB, y posteriormente, a medida que avanzamos en las generaciones, este nivel va decayendo hasta casi los -21dB, es decir, aplicando el algoritmo genético a una población de 50 individuos, inicialmente escogida aleatoriamente, se consigue que el nivel de los lóbulos secundarios descienda considerablemente. Esto se puede observar gráficamente en las figuras 3.3 y 3.4, que representan el diagrama de radiación al inicio del algoritmo y en la generación final.

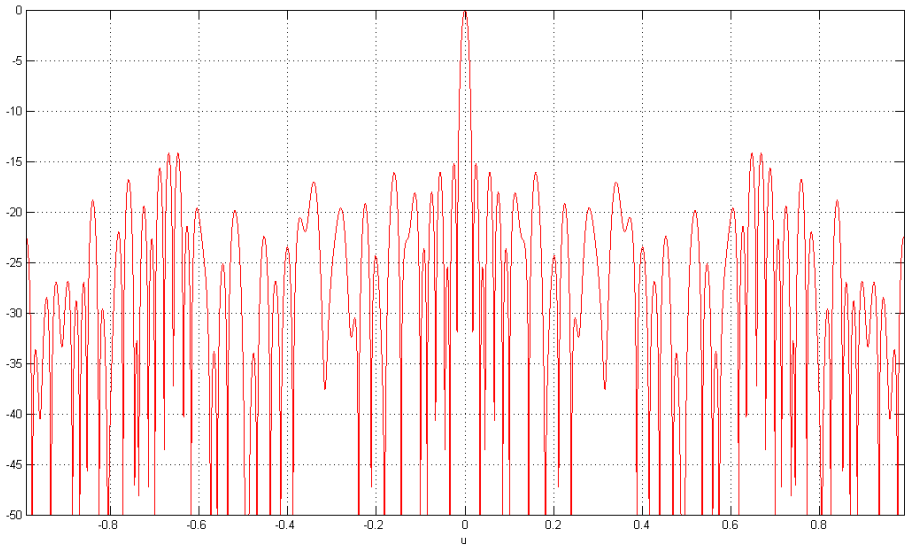

Figura 3.3: Diagrama de radiación inicial del AG con Selección por Ruleta

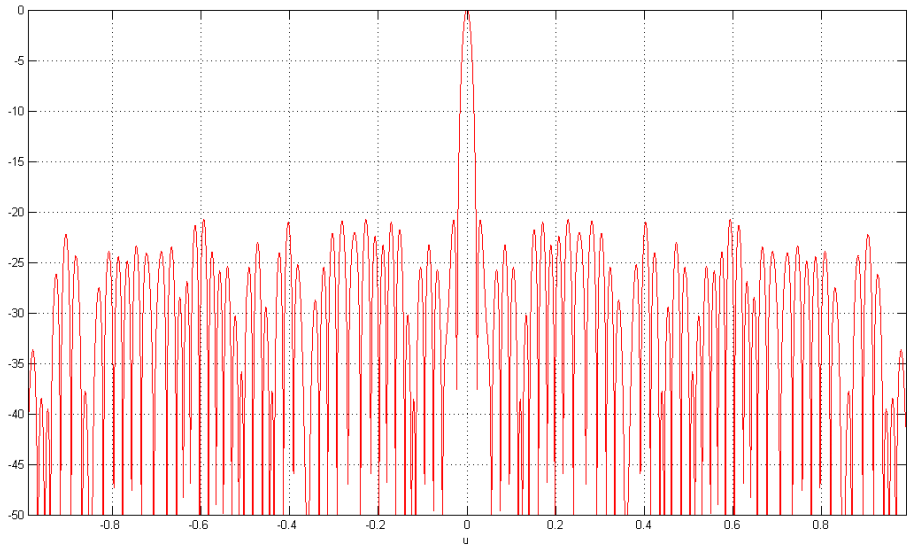

Figura 3.4: Diagrama de radiación final del AG con Selección por Ruleta

Se puede apreciar cómo, el diagrama de radiación representado para la población inicial (población más apta de la primera generación), figura 3.3, tiene su nivel de lóbulos secundarios por encima de -15dB y como, después de ejecutar el algoritmo, figura 3.4, ha conseguido descender estos lóbulos por debajo de la barrera de los -20dB, valor que se obtiene del mejor individuo de la última generación.

En este apartado, se ha expuesto un caso particular, pero se han realizado diversas pruebas para este caso, obteniéndose en todos los casos valores para el nivel de lóbulos secundarios inferiores a los -20dB.

3.4.1.2 Algoritmo Genético con Selección por Torneo

Los pasos principales que siguen los AG con selección por Torneo son muy similares a los del caso anterior, únicamente varia la forma en que se seleccionan los individuos:

1. Se crea una población aleatoria de 50 soluciones formadas genéticamente por 60 bits cada una.
2. Se evalúan cada una de las posibles soluciones, obteniendo el que inicialmente es el mejor individuo (la mejor solución de esta generación inicial) y que se guarda en una variable llamada *best_fitness*.

3. Ciclo de cada una de las generaciones/iteraciones:
   - Se aplica elitismo a la población de manera que se escogen los dos mejores individuos de la generación anterior y que pasan directamente a la siguiente.
   - Aplica la selección por Torneo escogiendo a los individuos que posteriormente se van a cruzar y mutar. De esta manera, se obtienen de dos en dos, las soluciones para la nueva población.
   - El paso anterior se ejecuta tantas veces hasta completar la nueva generación compuesta por 50 individuos.
4. Una vez que la nueva generación ha sido formada, se evalúan de nuevo las soluciones obtenidas, comparando cada uno de los valores de *fitness* con la variable *best_fitness*. Si alguna de esas soluciones mejora este valor, se guarda esa solución como la mejor.
5. Si no se ha llegado al máximo de iteraciones propuesto o a la condición de finalización, se vuelve al paso 3 para comenzar de nuevo con la generación de una nueva población.

En este caso, tras realizar las pruebas con el AG con selección por torneo con un bit de mutación, se obtienen resultados muy parecidos a los logrados en la selección por ruleta, aunque tal y como se observa en la figura 3.5, se consigue descender el nivel de lóbulos secundarios más rápidamente que en el caso anterior. Como en el caso anterior, en la figura se muestra el valor del nivel de lóbulos secundarios para el individuo más apto de cada una de las generaciones. Así, cuando han pasado poco más de 600 generaciones el nivel ha descendido por debajo de los -21dB y se mantiene constante.

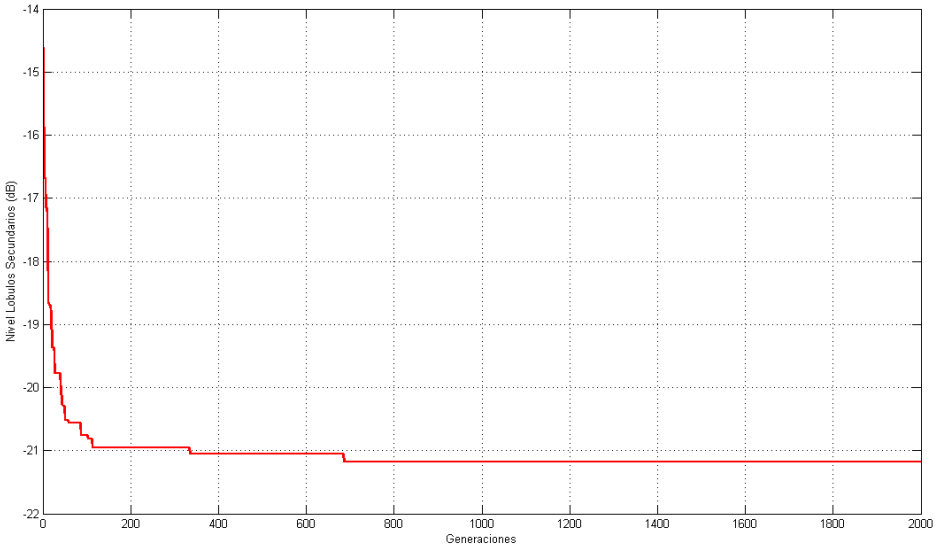

Figura 3.5: Relación de lóbulo principal a secundario con AG con Selección por Torneo

Tal y como se ha mostrado anteriormente para el caso con selección por ruleta, el hecho de que descienda el nivel de lóbulos secundarios se puede observar mejor en los diagramas de radiación tanto para la generación inicial y la última generación que se ha obtenido.

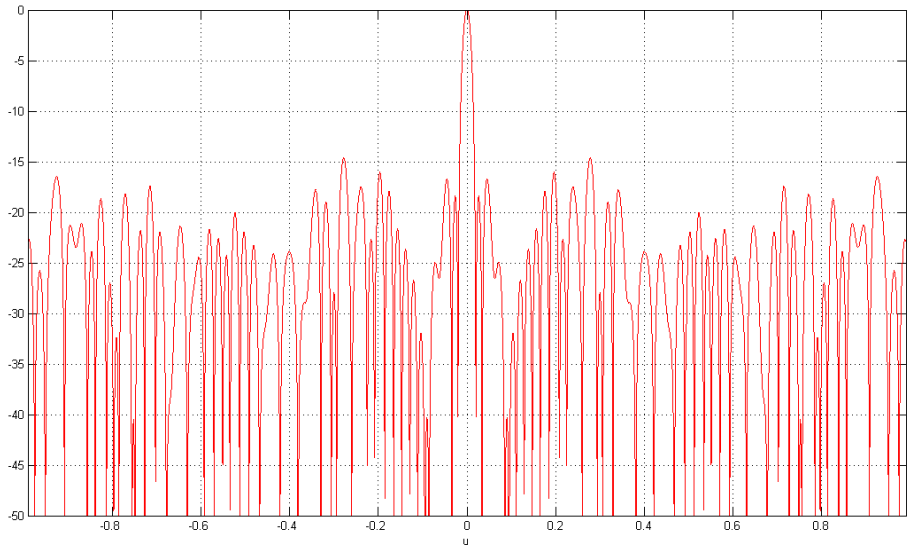

Figura 3.6: Diagrama de radiación inicial del AG con Selección por Torneo

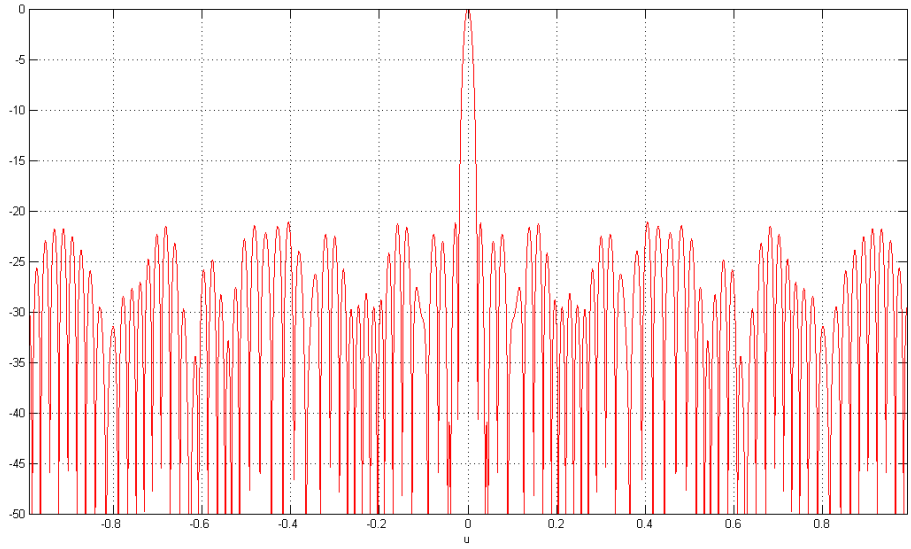

Figura 3.7: Diagrama de radiación final del AG con Selección por Torneo

Se puede comprobar, observando los diagramas de radiación, como se consigue descender el nivel de lóbulos secundarios desde un valor inicial por encima de -15dB en el diagrama de radiación inicial (individuo más adecuado de la población perteneciente a la generación inicial), figura 3.6, hasta un valor por debajo de los -20dB en el diagrama de radiación final (individuo más apto de la población de la última generación), figura 3.7, tras aplicar emplear un algoritmo genético con selección por torneo.

Como ocurría en el caso anterior, para mayor simplicidad solo se ha expuesto un caso particular, aunque se han realizado diversas simulaciones para esta versión del AG. En todas ellas, se han obtenido resultados parecidos consiguiendo un descenso del nivel de lóbulos por debajo de los -20dB, incluso como en el caso expuesto, se han conseguido simulaciones que descendían el valor de los lóbulos secundarios por debajo de los -21dB.

Comparando ambos casos, los diagramas de radiación y las gráficas de la evolución del nivel de lóbulos secundarios a lo largo de las generaciones, tanto para el AG con selección por ruleta como para el AG con selección por torneo, se puede asegurar que con la selección por torneo, se consigue un más rápida convergencia antes el algoritmo y que además, proporcione un nivel inferior al que se obtiene aplicando selección por ruleta. Esto se debe, como se ha dicho anteriormente, a que el AG con selección por torneo es más eficiente que el AG con selección por ruleta, debido a que este último escoge en más ocasiones a individuos que no mejoran necesariamente la generación anterior.

3.4.2 Conclusiones

Después de analizar los resultados obtenidos, se observa cómo, aplicando el algoritmo genético a una población de individuos inicial aleatoria, desciende el nivel de lóbulos secundarios del diagrama de radiación de manera considerable en ciertos casos.

El algoritmo genético se ha aplicado de dos maneras distintas: uno con Selección por Torneo y otro con Selección por Ruleta. En ambos casos, el algoritmo consigue descender los lóbulos secundarios pero en diferente magnitud. El algoritmo genético aplicando selección por Torneo, consigue descender los lóbulos más rápidamente que aplicando la selección por Ruleta, además de que consigue un menor nivel de radiación. En el ejemplo expuesto para Torneo se consiguió un nivel por debajo de los -21dB mientras que para la selección por ruleta, el nivel no descendió por debajo de los -20dB.

No obstante, se puede decir que la optimización realizada con el algoritmo genético para ambos casos es muy favorable, consiguiendo reducir el nivel de los lóbulos considerablemente.

3.5. Implementación del Algoritmo PSO

El Algoritmo Basado en Enjambres o Algoritmo PSO (*Particle Swarm Optimization*), tal y como se explica en el capítulo anterior, se basa en el movimiento de individuos dentro de una bandada de pájaros o un banco de peces, estudiando el modo en que el grupo es capaz de dirigirse en una dirección sin que haya "choques" entre los individuos que lo forman.

En este algoritmo se utilizan dos fórmulas detalladas en el capítulo anterior y que definen la velocidad y la posición de cada una de las partículas dentro del conjunto. Estas fórmulas son:

$$v_i^{t+1} = v_i^t + \varphi_1 U_1(0,1) * (p_i - x_i^t) + \varphi_2 U_2(0,1) * (s - x_i^t) \tag{3.4}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{3.5}$$

En este caso, el número de partículas que va a componer el grupo va a ser de 50 partículas, asociadas cada una de ellas a un vector de posición binario de 60 bits (lo que indica que se optimizarán arrays de 120 elementos) y un vector de velocidad con valores enteros del mismo tamaño. Inicialmente se tienen que inicializar estas variables con valores aleatorios para garantizar que las partículas se dispersen dentro del grupo.

Como se puede ver en la fórmula (3.4), hay una serie de parámetros que se pueden modificar para realizar diferentes versiones del algoritmo. Por un lado, están los parámetros aleatorios *U(0,1)* que se modifican cada vez que se ejecuta la fórmula, debido a su aleatoriedad. Y por otro lado, los parámetros φ, cuyo valor habitualmente se encuentra en el intervalo entre 0 y 1, aunque se han visto anteriormente que algunas versiones del algoritmo PSO permiten utilizar valores mayores a 1.

Los pasos principales que ha seguido la implementación de algoritmo PSO, son los siguientes:

1. Se crean aleatoriamente 50 partículas, colocadas en posiciones definidas por 60 bits cada una. Además cada una de estas partículas tendrá un vector de velocidades que se iniciará totalmente aleatorio.
2. Se evalúan cada una de las posibles soluciones (cada una de las posiciones que toman las partículas), obteniendo la mejor posición inicial para cada partícula que se guarda en la variable *local_best*, y la mejor posición de todas y que ofrece la mejor solución al problema, y que se guarda en la variable *global_best*.

3. El siguiente paso es definir los parámetros φ. En este caso, se han realizado varias pruebas para distintos valores de los parámetros.
4. Ciclo de cada una de las iteraciones:
   - Se aplica la fórmula para obtener las velocidades de cada una de las partículas.
   - Después de obtener los valores de las velocidades, se aplica la otra fórmula para obtener las nuevas posiciones de cada partícula. Como se está aplicando el algoritmo al caso binario, estas posiciones han de ser cadenas de bits.
   - Una vez que se tienen todas las nuevas posiciones se comprueba si estas posiciones son mejores que las anteriores para cada una de las partículas y si es mejor que en los instantes anteriores se guarda.
   - Se obtiene cual es la mejor posición de todas para compararlo con el vector s que tenemos guardado del instante anterior. Si la comparación da que la actual posición es mejor solución para el problema que la anterior, se sustituye.
   - Se vuelve al inicio del paso 4 si no se ha llegado a la condición de parada, que será simplemente un número máximo de iteraciones.
5. Cuando se llega al final de las iteraciones que se han definido inicialmente, se tendrá la mejor solución encontrada para el problema, que estará guardada en la variable s.

Estos pasos se pueden observar gráficamente en la figura 3.8, donde se representa diagrama de flujo de nuestro algoritmo PSO.

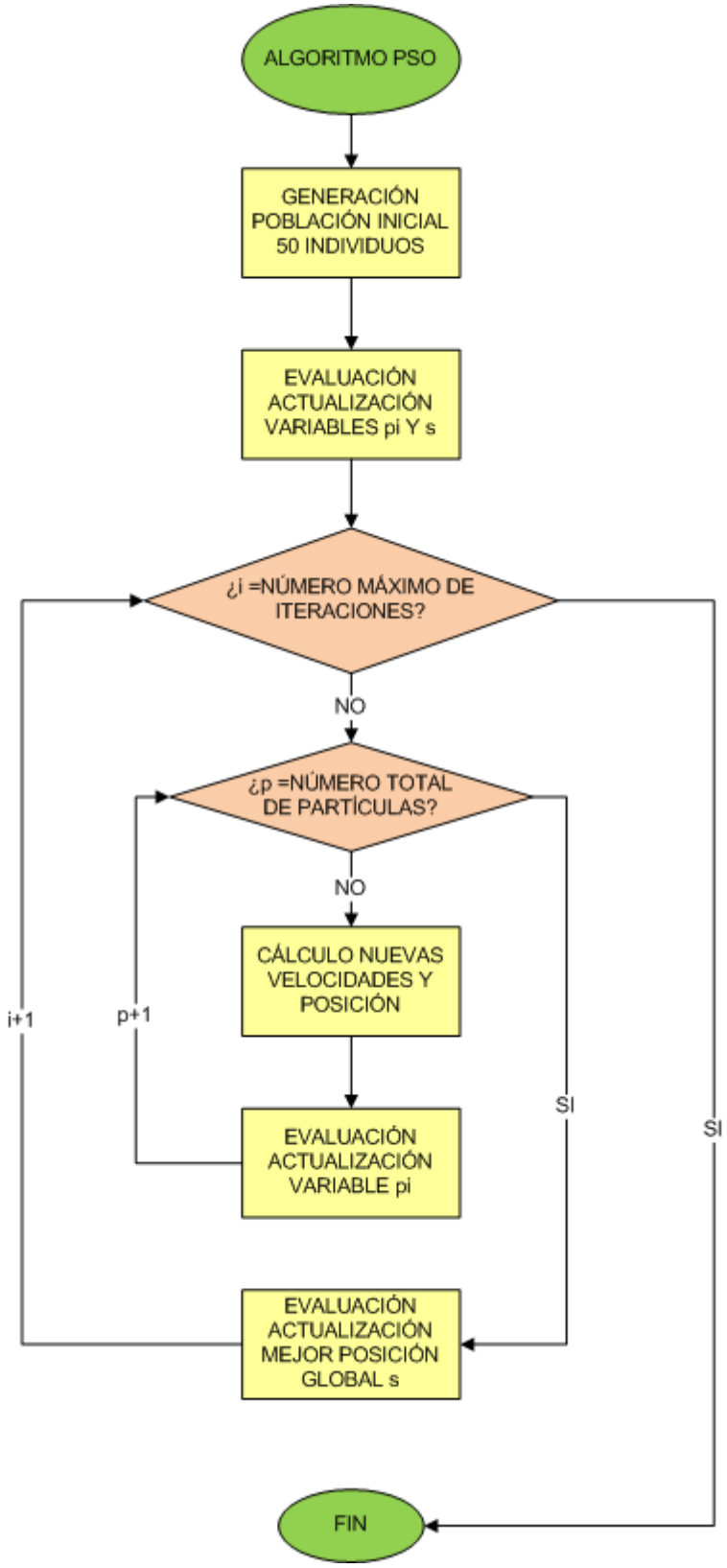

Figura 3.8: Diagrama de Flujo de un Algoritmo PSO

3.5.1 Resultados Simulaciones Realizadas

Se han realizado tres pruebas: la primera para un Algoritmo PSO con los parámetros φ_1 = 0.9 y φ_2 = 0.1, otra para valores de los parámetros φ_1 = 0.5 y φ_2 = 0.5 y por último, modificando los parámetros de nuevo, φ_1 = 0.1 y φ_2 = 0.9.

3.5.1.1 Algoritmo PSO con φ_1 = 0.9 y φ_2 = 0.1

En la primera prueba realizada con el Algoritmo PSO, se han escogido los parámetros φ_1 = 0.9 y φ_2 = 0.1, lo que significa que se está dando más importancia a la mejor solución de cada partícula individualmente que a la mejor solución global de todas las partículas a lo largo del tiempo (ver ecuación (3.4)).

En primer lugar, se puede observar en la figura 3.9 la evolución del máximo valor de los lóbulos secundarios a lo largo de las iteraciones. Como se puede apreciar en la dicha figura, creando un grupo totalmente aleatorio de 50 partículas, la relación inicial de lóbulo principal a secundario es de -13.5dB aproximadamente, y a medida que van pasando las iteraciones, este máximo valor va descendiendo progresivamente hasta que converge, hecho que ocurre cuando se llevan alrededor de 5000 iteraciones, donde el algoritmo se mantiene en un valor de -17.5dB. Este hecho, como ocurre anteriormente con el Algoritmo Genético, se puede advertir mejor si se observan, a continuación, los diagramas de radiación para la iteración inicial (partícula más apta en la iteración inicial) y para la iteración final (partícula más apta en la última iteración), figuras 3.10 y 3.11, en donde se puede apreciar como los lóbulos secundarios descienden desde un valor por encima de los -13.5dB hasta un valor por encima de los -20dB, exactamente -17.5dB.

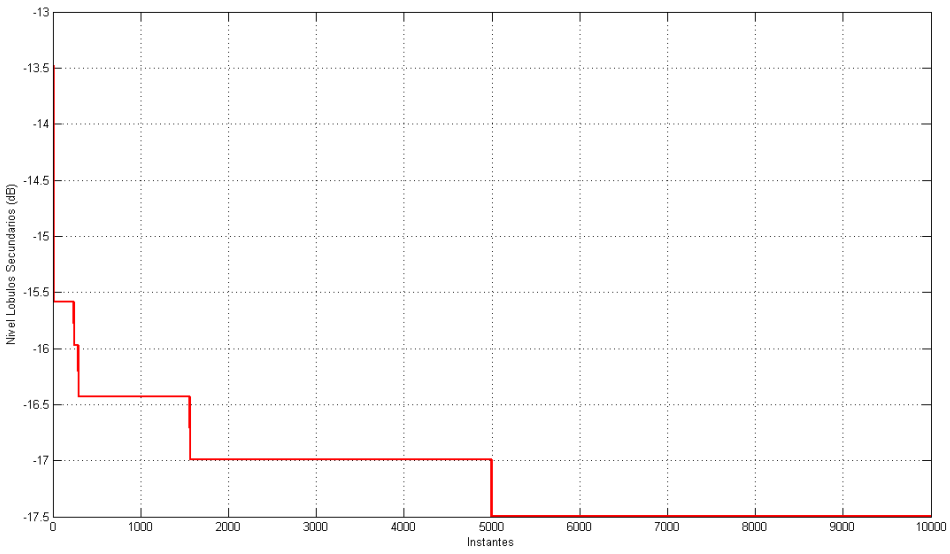

Figura 3.9: Relación de lóbulo principal a secundario con APSO con φ_1=0.9 y φ_2=0.1

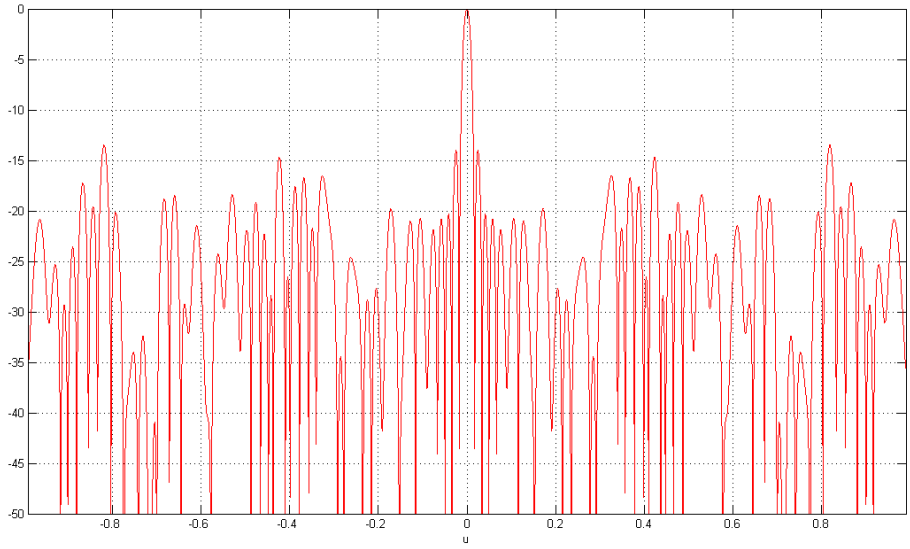

Figura 3.10: Diagrama de radiación inicial con APSO con φ_1=0.9 y φ_2=0.1

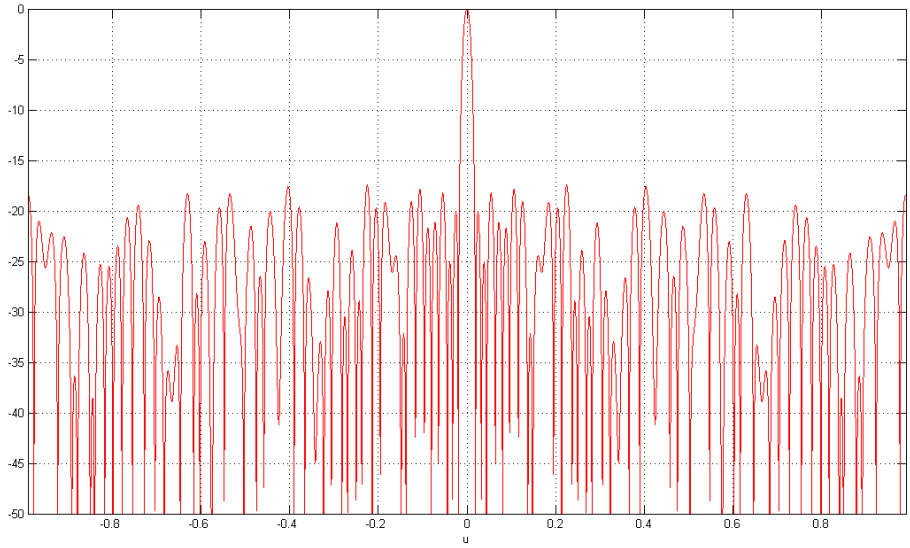

Figura 3.11: Diagrama de radiación final con APSO con φ_1=0.9 y φ_2=0.1

En todas las pruebas que se han realizado para este algoritmo con estos parámetros, se han obtenido resultados muy parecidos, siempre obteniendo un valor cercano -17.5dB, aunque en alguna ocasión el valor de los lóbulos secundarios ha descendido hasta casi los -18dB.

3.5.1.2 Algoritmo PSO con φ_1 = 0.5 y φ_2 = 0.5

En la segunda prueba realizada con el Algoritmo PSO, el algoritmo ha sido implementado con los parámetros φ_1 = 0.5 y φ_2 = 0.5, lo que hace que se dé la misma importancia tanto a la mejor solución de cada partícula en cada iteración como a la mejor solución global. Teniendo en cuenta lo anterior, los resultados que se obtienen son los siguientes.

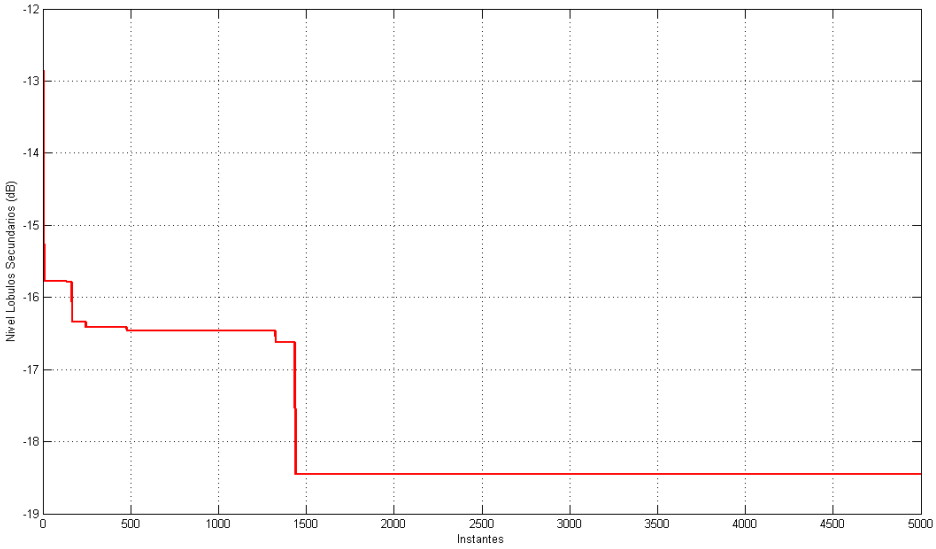

Figura 3.12: Relación de lóbulo principal a secundario con APSO con φ_1=0.5 y φ_2=0.5

Como se ve en la figura 3.12 el nivel de lóbulos secundarios desciende notablemente, a pesar de ejecutar el algoritmo durante 5.000 iteraciones (en el caso anterior, se ejecutaba durante 10.000 iteraciones). De estar inicialmente en un valor de casi -13dB, después de aplicar el algoritmo PSO se consigue descender el valor por debajo de los -18dB.

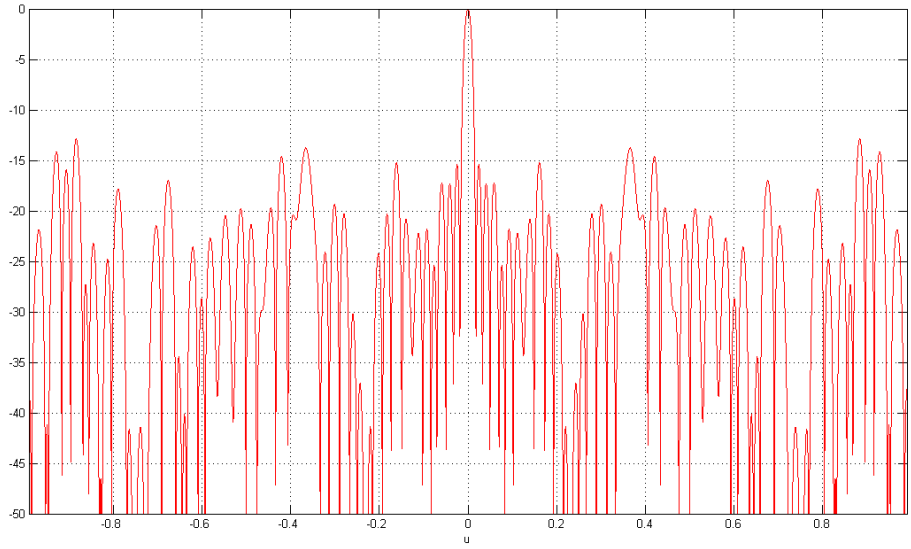

Figura 3.13: Diagrama de radiación inicial con APSO con φ_1=0.5 y φ_2=0.5

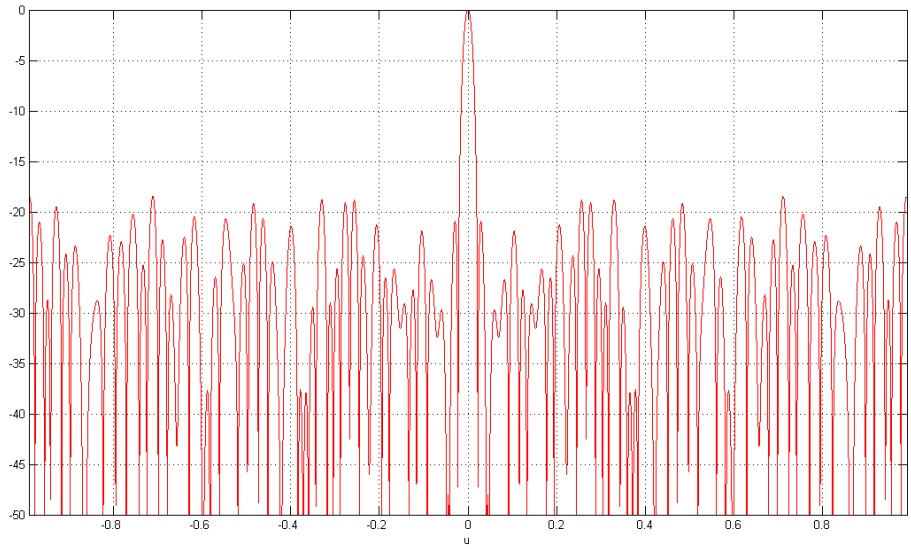

Figura 3.14: Diagrama de radiación final con APSO con φ_1=0.5 y φ_2=0.5

Las figuras 3.13 y 3.14 muestran los diagramas de radiación al inicio (la partícula más adecuada de la iteración inicial) y al final de la ejecución del algoritmo (mejor partícula en la iteración final), en donde se aprecia como el nivel de los lóbulos secundarios tiene un valor muy inferior en el diagrama de radiación final.

3.5.1.3 Algoritmo PSO con φ_1 = 0.1 y φ_2 = 0.9

En la última simulación realizada para el algoritmo PSO, se han tomado unos parámetros totalmente opuestos a los escogidos en la primera implementación. En este caso, vamos a dar mayor importancia a la mejor solución de todas las partículas en todos los instantes por encima de la mejor solución de cada partícula.

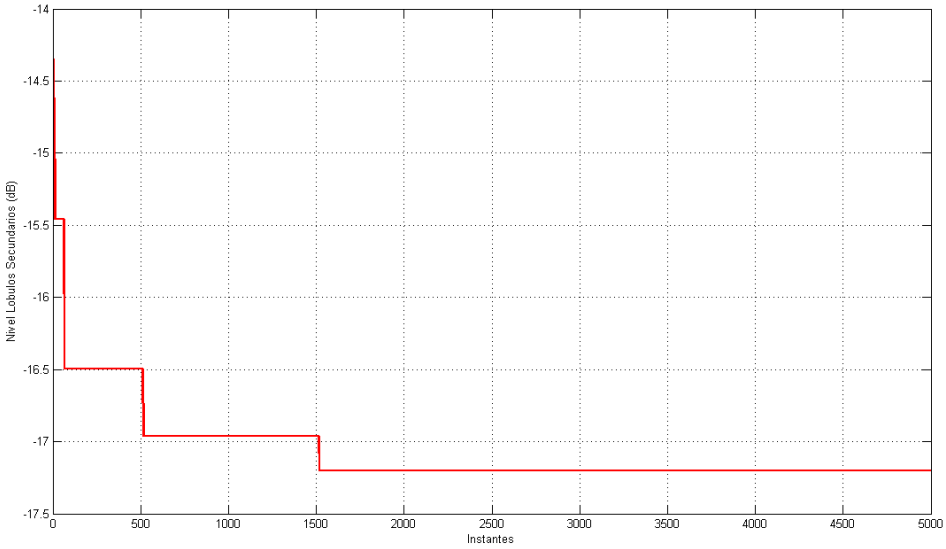

Figura 3.15: Relación de lóbulo principal a secundario con APSO con φ_1=0.1 y φ_2=0.9

En la figura 3.15, donde se ve la progresión del máximo valor del nivel de lóbulos secundarios a lo largo de las iteraciones, se puede observar como este nivel desciende desde un valor de -14dB hasta un valor cercano a -17.5dB. Este descenso se puede apreciar mejor en las figuras 3.16 y 3.17 de los diagramas de radiación inicial y final. Como se aprecia en dichas figuras, el descenso del valor de los lóbulos secundarios es mayor que en los ejemplos detallados anteriormente.

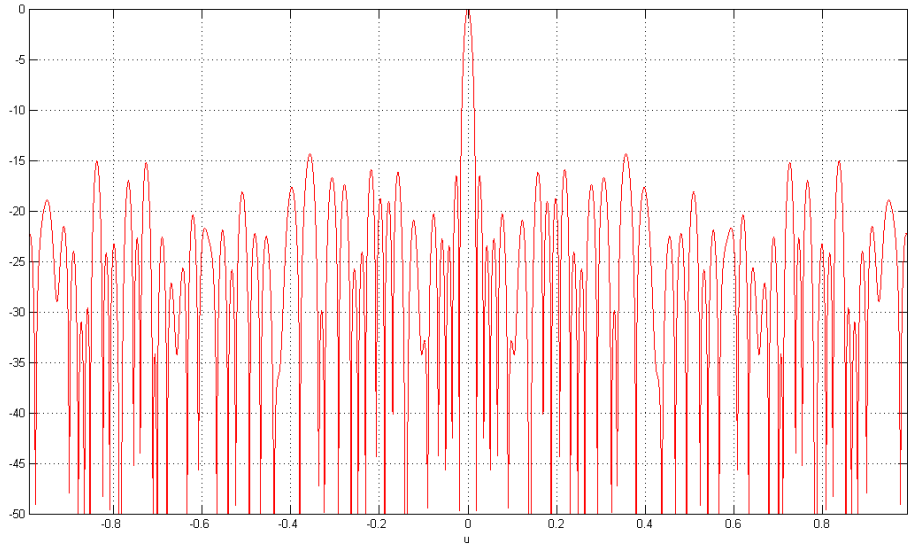

Figura 3.16: Diagrama de radiación inicial con APSO con φ_1=0.1 y φ_2=0.9

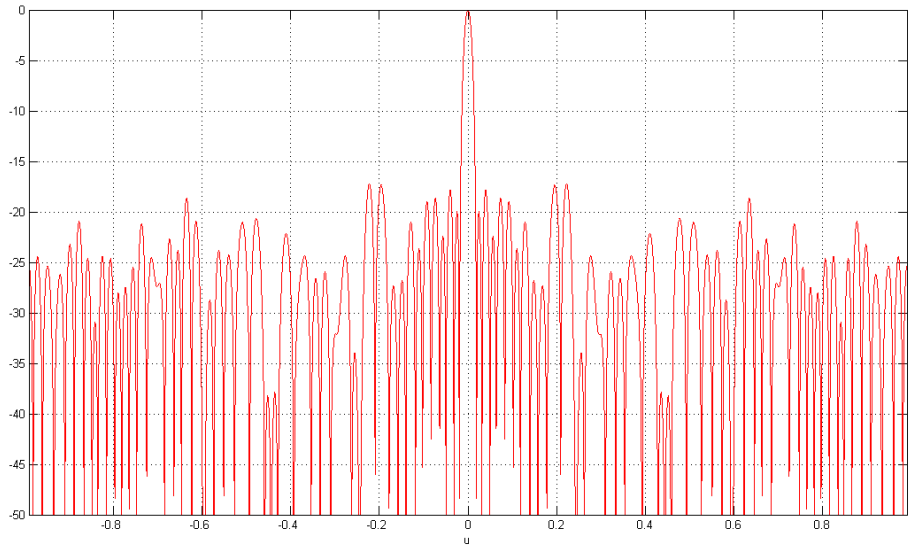

Figura 3.17: Diagrama de radiación final con APSO con φ_1=0.1 y φ_2=0.9

Aunque por brevedad aquí se ha expuesto un caso particular, se han realizado diversas simulaciones también para este caso, obteniéndose valores similares para el nivel de los lóbulos al finalizar la aplicación del algoritmo, rondando los -17.5dB.

3.5.2 Conclusiones

El Algoritmo basado en Enjambre de Partículas o PSO se ha aplicado para disminuir el nivel de lóbulos secundarios de un *thinned array*. Para ello se han realizado varias simulaciones en las que hemos modificado los parámetros φ_1 y φ_2 del algoritmo básico consiguiendo resultados aceptables aunque peores que para el caso del algoritmo genético.

En cuanto al número de iteraciones, después de realizar bastantes pruebas, se comprobó que el Algoritmo PSO necesitaba un elevado número de las mismas para conseguir una solución aceptable y que al mismo tiempo, el algoritmo convergiera.

Por tanto, se puede concluir que el coste computacional de emplear PSO para la síntesis propuesta en este proyecto es mayor que para el caso del algoritmo genético.

3.6. Implementación del Algoritmo Basado en Colonias de Hormigas (ACO)

El último de los algoritmos que se han utilizado es el Algoritmo Basado en Colonias de Hormigas (ACO). Tal y como se explicó en el capítulo anterior, se basa en el movimiento de las hormigas dentro de la colonia y su forma de seleccionar la ruta entre la fuente de alimento y el nido, de manera que terminan utilizando el camino más corto.

Para la implementación de este algoritmo son necesarios nodos que representan la ubicación en la que se encuentra cada hormiga. Estos nodos estarán compuestos en nuestro caso de 25 bits cada uno, lo que significa que existen 2^{25} posibles nodos, exactamente 33.554.432. Se ha reducido el número de bits de los nodos (posibles soluciones al problema propuesto), debido a que cada hormiga evalúa las soluciones de todos los nodos adyacentes no visitados en cada paso que dé. Esto hace que el coste computacional sea mayor que en los AG y los PSO. Por tanto, para evitar que a mitad del proceso se quede sin memoria virtual y que las simulaciones se realicen en un tiempo adecuado, se reduce la longitud del array.

Todo lo anterior conlleva que el algoritmo necesite menos iteraciones que los anteriores para converger a una solución que consideremos óptima. Se determina que se realicen alrededor de 150 iteraciones y dentro de cada una de ellas, cada hormiga dará 12 pasos para completar el camino.

La construcción de la ruta se realiza aplicando una regla de decisión basada en probabilidades a la hora de elegir el siguiente nodo que van a visitar. Para ello se utiliza la ecuación (3.6), donde la probabilidad p_{ij} de que una hormiga k en el nodo i pase al nodo j que pertenece al conjunto N en cada iteración será:

$$p_{ij}^{k}(t) = \frac{a_{ij}(t)}{\sum_{l \in N_i^k} a_{il}(t)} \tag{3.6}$$

donde N_i es el conjunto de estados del nodo i que la hormiga k no ha visitado todavía. Y $a_{ij}(t)$ es una tabla de decisión del nodo i que se obtiene mediante la fórmula (3.7):

$$a_{ij}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum_{l \in N_i}\left[\tau_{il}(t)\right]^{\alpha}\left[\eta_{il}\right]^{\beta}} \qquad \forall j \in N_i \tag{3.7}$$

donde:

- $\tau_{ij}(t)$ es la cantidad de feromona asociada al arco (i,j) en la iteración t.
- N_i es el conjunto de nodos adyacentes del nodo i.
- $\eta_{ij} = \frac{1}{c_j}$ donde c_j es el coste de la solución asociada al nodo j.
- α y β son dos parámetros que determinan la influencia del rastro de feromona.

Una vez que todas las hormigas hayan completado su recorrido, se actualiza el rastro de feromona. El primer paso consiste en aplicar la evaporación en todos los arcos de la siguiente manera:

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t-1) \tag{3.8}$$

donde ρ es el índice de evaporación que se encuentra en el intervalo $0<\rho<1$. Esta acción se realiza para evitar la acumulación del nivel de feromona a lo largo de las iteraciones, además de esta manera, en los caminos que aportan una peor solución, su nivel de feromona desciende poco a poco a medida que pasan las iteraciones.

Después de la evaporación, las hormigas depositan feromona en los arcos que se incluyen en su camino con la ecuación (3.9):

$$\tau_{ij}(t) \leftarrow \tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{3.9}$$

donde $\Delta\tau_{ij}$ es la cantidad de feromona que deposita la hormiga por donde pasa. Se define por:

$$\Delta\tau_{ij}(t) = \frac{1}{c_j}, \text{ si el arco (i,j) pertenece a la ruta de la hormiga.} \tag{3.10}$$

Como se puede ver, el aporte de feromona depende del coste, valor de *fitness* de la solución que ofrece el nodo, perteneciente a la ruta.

A medida que aumenta el tamaño de las muestras, el algoritmo tiende a empeorar su comportamiento, por lo que para evitarlo existen dos mejoras relativas a la actualización de la feromona:

- **AS Elitista**. Se trata de reforzar, mediante la adicción de una cantidad constante de feromona, los arcos pertenecientes a la mejor ruta encontrada desde el principio del algoritmo. Esta cantidad se corresponde con el coste mínimo de la mejor ruta, que se denomina c^*.
- **AS Rank**. Esta mejora sigue el camino de AS Elitista pero en esta versión no sólo influye la mejor ruta sino que las hormigas que componen los mejores caminos van a depositar una cantidad adicional de feromona proporcional a su rango. Antes de actualizar la feromona, todas las hormigas son clasificadas y ordenadas en función del mínimo coste alcanzado por cada ruta.

En el algoritmo que se ha implementado para la realización de las simulaciones que se verán a continuación en este apartado se ha escogido la mejora AS Elitista.

Por último, se realiza la disminución del aporte inicial de feromona que se da a los nodos visitados por primera vez. En el algoritmo implementado el aporte inicial es de 3 unidades de feromona. Para llevar a cabo esta disminución se aplica la siguiente fórmula:

$$\tau_0 \leftarrow (1-\rho)\tau_0 \tag{3.11}$$

Los pasos principales que siguen los Algoritmos Basado en Colonias de Hormigas, son los siguientes:

1. Se crean 25 hormigas que se colocan aleatoriamente en 25 nodos existentes. Cada uno de los nodos viene definido por 25 bits.
2. Se evalúan cada una de las posibles soluciones (cada nodo), actualizando tanto el *fitness* asociado a esa solución como el nivel de feromona.
3. En el siguiente paso se inicializan los parámetros necesarios para la ejecución del algoritmo: α, β, el índice de evaporación ρ y el aporte inicial de feromona en los nodos visitados por primera vez.
4. Ciclo de cada una de las iteraciones:
    - Ciclo para cada hormiga:

      Se actualiza el histórico de nodos por los que ha pasado la hormiga ya que no puede pasar por un nodo por el que ha pasado previamente. Para ello se utiliza una variable en el que se guarda la ruta que ha recorrido.

Las hormigas dan doce pasos cada una para completar la ruta. En cada uno de estos pasos se realizan las operaciones pertinentes para escoger el siguiente nodo, de forma que se aplica la ecuación (3.6).

Por último, una vez que se ha completado la ruta, es evaluada para comprobar si la solución a la que se ha llegado es mejor que la que se tenía guardada previamente.

Estos pasos se repiten hasta que se han culminado las rutas de todas las hormigas.

- Una vez que se han completado todas las rutas, se procede a la evaporación y actualización de feromona en todos los nodos.
- Se comprueba si la solución a la que se ha llegado en esta iteración es mejor que la que teníamos en las anteriores. Si es mejor, se guarda para continuar descendiendo el nivel y realizar comparaciones posteriores.
- Por último, se genera una nueva colonia aleatoria de hormigas.

5. Si se comprueba que no se ha llegado a la condición de parada, se vuelve de nuevo al paso 4 para volver a ejecutar todo el proceso.

Estos pasos a seguir en la ejecución del Algoritmo ACO, está detallado en el siguiente diagrama de flujo de la figura 3.18.

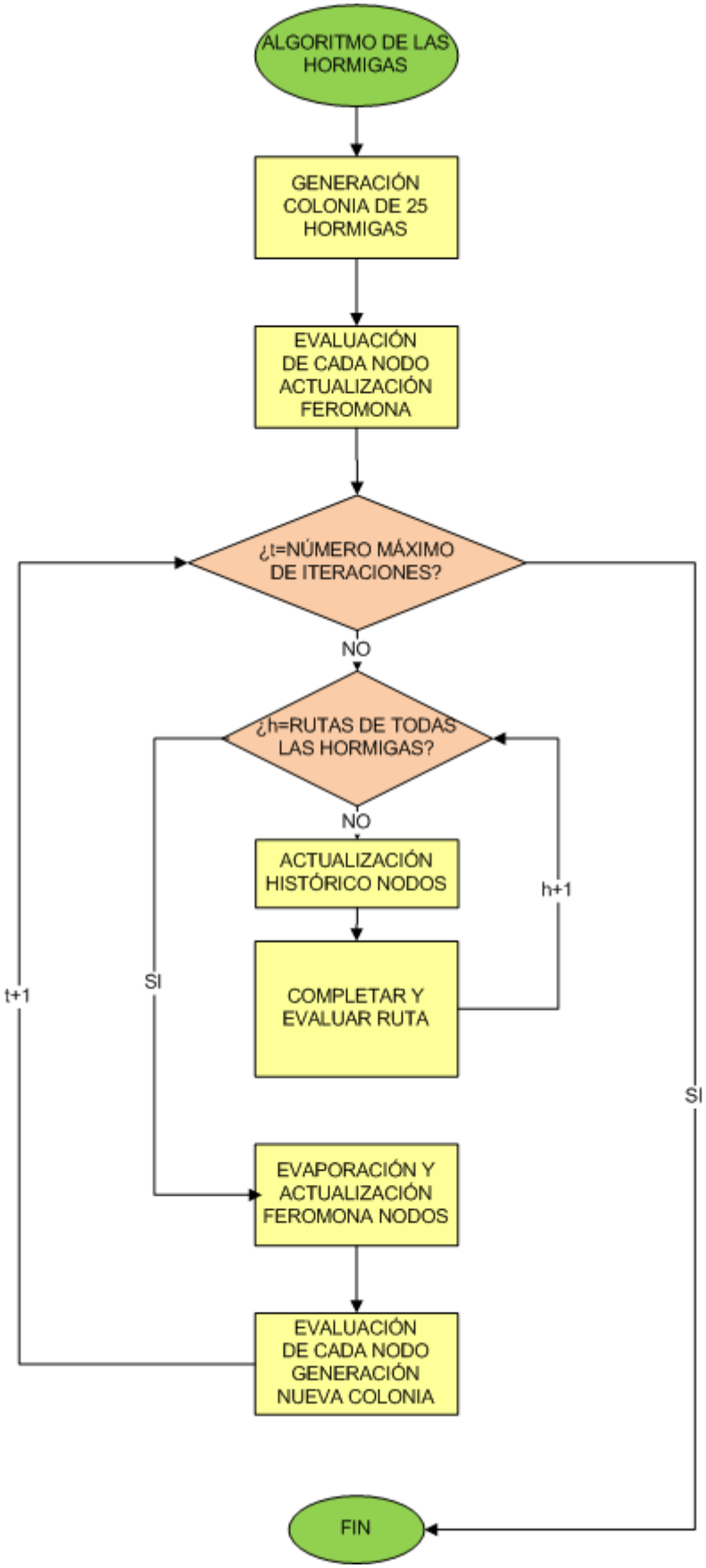

Figura 3.18: Diagrama de Flujo de un Algoritmo Basado en Colonias de Hormigas

3.6.1 Resultados Simulaciones Realizadas

Se han realizado simulaciones para una única versión del algoritmo ACO dado que el coste computacional del algoritmo es elevado y requiere de mucho tiempo de ejecución. Los parámetros toman los valores, $\alpha = 0.6$, $\beta = 0.4$ y $\rho = 0.4$.

3.6.1.1 Algoritmo ACO con $\alpha = 0.6$, $\beta = 0.4$ y $\rho = 0.4$

En esta primera simulación se han escogido como parámetros $\alpha = 0.6$ y $\beta = 0.4$, lo que significa que otorgamos más valor a la cantidad de feromona asociada a un arco que a la solución que otorga dicho arco (véase ecuación (3.6)). Independientemente de este hecho, también se escogen estos valores para equilibrar la diferencia entre la solución que se obtendrá de dicho nodo y la cantidad de feromona. Por último, el otro parámetro que hay que escoger, es el índice de evaporación, que en este caso, será del 40% o $\rho = 0.4$.

Para empezar, en la figura 3.19 se muestra la evolución del máximo valor para este nivel a lo largo de las iteraciones. Si se estudia detenidamente esta figura, se puede determinar que el nivel desciende considerablemente a pesar de realizar muy pocas iteraciones, comparado con las que se realizan para los dos algoritmos anteriores. Se puede observar que el nivel inicial se encuentra por encima de los -12dB y que, poco a poco, este nivel va descendiendo hasta converger en un valor menor a -16dB.

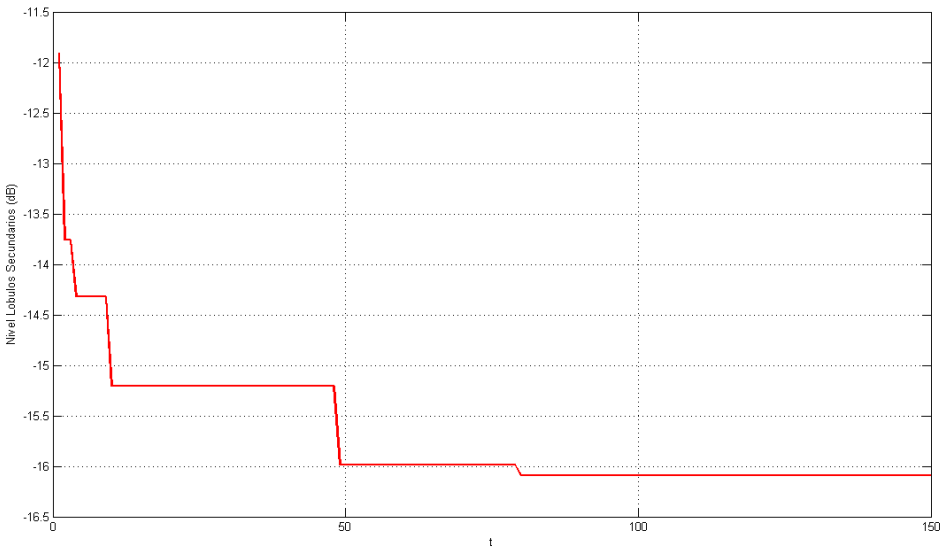

Figura 3.19: Relación de lóbulo principal a secundario con ACO con α=0.6, β=0.6 y ρ=0.4

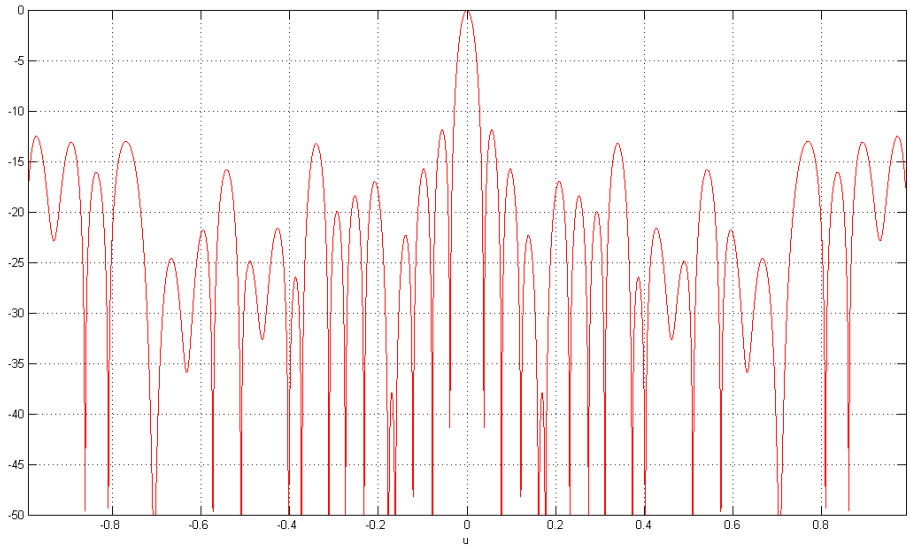

Figura 3.20: Diagrama de radiación inicial con AH con α=0.6, β=0.6 y ρ=0.4

Figura 3.21: Diagrama de radiación final con AH con α=0.6, β=0.6 y ρ=0.4

Como se ve en las figuras 3.20 y 3.21, los diagramas de radiación al inicio de las iteraciones y al final, muestran claramente el descenso del nivel de los lóbulos secundarios. En el diagrama de radiación inicial, figura 3.20, los lóbulos secundarios se encuentran muy por encima de la marca de los -15dB, mientras que al final, figura 3.21, estos se encuentran todos por debajo de los -15dB. Sin embargo, debido al reducido número de iteraciones y a que el número de elementos de array más reducido que en el caso de AG o PSO, el algoritmo no consigue

alcanzar los valores obtenidos por estos otros algoritmos. Estos resultados han sido repetidos con diferentes valores iniciales obteniendo resultados similares.

3.6.2 Conclusiones

En este apartado se ha usado una implementación de algoritmo basado en colonia de hormigas para resolver el problema *thinned arrays*. En este caso, tras las simulaciones realizadas se ha conseguido descender el nivel de los lóbulos secundarios hasta valores en torno a los -16dB. Dicho nivel es superior al de los anteriores algoritmos debido al reducido número de iteraciones, y a que el tamaño del array empleado fue menor.

Si se presta atención a los diagramas de radiación de este algoritmo, se observa como los lóbulos son más anchos que en los apartados anteriores. Esto se debe a que el número de bits (que significa el número de elementos del array) que conforman la posición de las hormigas es más pequeño que en el caso de los bits que formaban el cromosoma de un individuo en el algoritmo genético o de las partículas de PSO.

Capítulo 4. Síntesis de Arrays Bifrecuencia

En este capítulo se realiza un estudio sobre la Síntesis de Arrays Bifrecuencia. A diferencia del capítulo anterior los arrays contienen radiadores que podrán ser empleados indistintamente para dos frecuencias diferentes. Dado que en los casos anteriores se demostró que el algoritmo basado en colonias de hormigas tenía un elevado coste computacional solamente fueron empleados en este capítulo el algoritmo genético y PSO.

4.1. Introducción

En el capítulo anterior, se ha detallado la implementación de los Algoritmos Genético, Basado en Enjambres (PSO) y Basado en Colonias de Hormigas (ACO), para resolver el problema de *thinned arrays* para una única frecuencia, esto es, todos los elementos del array distan lo mismo unos de otros, d = 0.5λ, y estaban formados por elementos de amplitud 0 ó 1.

En este capítulo, se van a optimizar arrays bifrecuencia, es decir, haciendo uso del concepto de *thinned* array, se optimizará una única configuración que funcione a dos frecuencias diferentes. Para la realización de las siguientes implementaciones, se han escogido los algoritmos Genético y PSO, debido al elevado coste computacional del algoritmo ACO.

La base para realizar estos diseños de arrays bifrecuencia están inspirados en el concepto de *thinned* array entrelazados definido por Haupt en [27], para la realización de dos arrays (ambos a la misma frecuencia) que pueden ser uno suma y otro diferencia. Sin embargo, en este proyecto, lo que se intentará es utilizar el mismo concepto para el diseño de dos arrays que funcionen a dos frecuencias diferentes. Se utilizarán arrays con frecuencia *f*, 1.25*f* y 1.5*f*, de manera que los elementos se encuentren a distintas distancias relativas unos de otros dependiendo a la frecuencia en la que se evalúen los resultados.

4.2. Interleaved Thinned Linear Array

Los *Thinned* Arrays Lineales e Intercalados (*Interleaved Thinned Linear Array*) tal y como los concibe Haupt en [27], están espaciados periódicamente, mediante un múltiplo entero de un conjunto mínimo de elementos espaciadores. Los arrays se conjuntan de forma que la apertura resultante parece ser uniforme, pero realmente está compuesto por dos arrays diferentes en la misma apertura. Dichos arrays son optimizados para tener un ancho de haz estrecho con el menor nivel de lóbulos secundarios posible.

Un array lineal con un par de elementos distribuidos en el *eje x* tiene un factor de array dado por la fórmula (4.1):

$$S(\theta) = \frac{1}{N} \sum_{n=1}^{N} a_n \cos\left[kd(n-0.5)u\right] \tag{4.1}$$

donde:

- u $\sin(\theta)$
- θ ángulo
- d distancia entre elementos
- k $2\pi/\lambda$
- λ longitud de onda
- a_n amplitud de los elementos que forman el array
- 2N número de elementos en el array

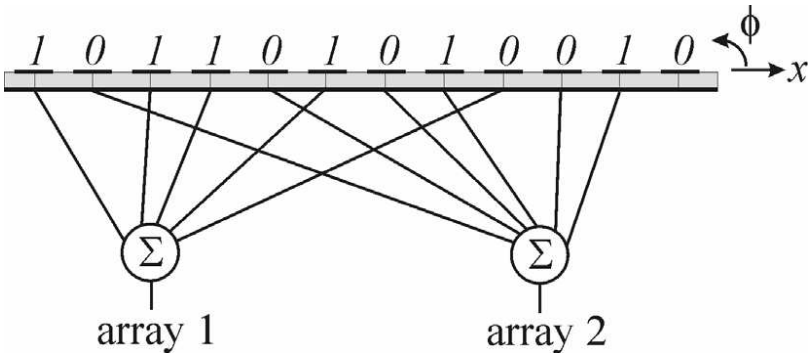

Figura 4.1: Diagrama de dos arrays intercalados

La amplitud de los elementos de un *thinned array* está limitada a *1* (*on* para el array 1 y *off* para el array 2), y 0 (*off* para el array 1 y *on* para el array 2). La figura 4.1 muestra el esquema de dos arrays intercalados. Los dos arrays son antisimétricos, lo que significa que si un array tiene los elementos *on*/*off*, el otro tiene los elementos en *off*/*on*.

La eficiencia del *taper* de un array puede ser calculado con la fórmula (4.2):

$$\eta_{ar} = \frac{n}{t} \tag{4.2}$$

donde *n*, es el número de elementos del array en *on* y *t*, el total de elementos del array. La amplitud se representa como:

$$a = [a_1, a_2, ..., a_N, 1 - a_N, ..., 1 - a_2, 1 - a_1] \tag{4.3}$$

y

$$a' = 1 - a \tag{4.4}$$

De esta manera la eficiencia total de *a* y de *a'*, para este caso particular sería de η_{ar}=100%.

A continuación se va a proceder a explicar cómo se ha aplicado un algoritmo genético y PSO para la optimización de este tipo de arrays intercalados bifrecuencia, en la que todos los elementos se encuentran equiespaciados λ/2 a una frecuencia f_1 aunque sin embargo, cada elemento del array puede pertenecer al array de dicha frecuencia, al de una frecuencia f_2, o a ninguno de los anteriores.

4.3. Función de Fitness

La función de *fitness* elegida para los array bifrecuencia es muy parecida a la que se utilizó en la síntesis de arrays de una única frecuencia en el capítulo anterior. De hecho se utiliza la misma fórmula del factor de array (ecuación (4.1)), pero aplicándola dos veces. De esta manera, cada vez que se ejecute esta función se hallarán dos valores.

A modo de ejemplo, teniendo la representación de un array como sigue (en el que existen tres estados 0, 1 y 2):

Expresión total del array:	0	0	1	2	2	2	1	0	0	0	1	1	1	1	2	2	0	1	2

Podemos obtener los dos arrays teniendo en cuenta que 1 representa los elementos con frecuencia f_1, 2 los que tienen frecuencia f_2 y 0 representa el estado sin elemento radiante (o carga adaptada), que quedarían expresados de la siguiente forma:

Array f_1:	0	0	1	0	0	0	1	0	0	0	1	1	1	1	0	0	0	1	0
Array f_2:	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	1	0	0	1

De esta manera, cuando se forma el array para los elementos de f_1, se mantienen aquellos con valor 1 y los demás, tanto los elementos a 0 como a 2, se colocan a 0. Ocurre lo mismo para formar el array para los elementos con f_2, solo que para poder aplicar el factor de array, en las posiciones donde se ha representado un 2, se colocan elementos con valor 1, para así tener un array con elementos formados por 1 y 0.

Una vez que se tienen los dos arrays, se puede calcular la función de *fitness* para cada uno de ellos (de forma idéntica a la empleada en el capítulo anterior para *thinned array* de frecuencia única), para más tarde poder calcular la **Función de *Fitness* Conjunta (FC)**, que se ha definido la ecuación (4.5), en la que se tiene en cuenta las dos funciones de *fitness* y se introducen dos constantes (k_1 y k_2 para ponderar de forma diferentes los diagramas de cada frecuencia).

$$FC = k_1\, fitness_1 + k_2\, fitness_2 \tag{4.5}$$

Los algoritmos empleados para optimizar esta función de *fitness* serán AG y PSO.

4.4. Implementación del Algoritmo Genético

El primer algoritmo que se va a aplicar es el Algoritmo Genético, dado que en el apartado de Síntesis de Arrays se comprobó que era el que mejor funcionaba para las directrices propuestas en cuanto a número de elementos de los arrays, iteraciones, etc.

En todas las simulaciones se ha implementado con una población de 50 individuos con 200 elementos de array cada uno. Como se ve, los individuos cuentan con más elementos en su cromosoma, y esto es debido a que, para conseguir mayor diversidad utilizando elementos con distinta frecuencia, las cadenas debían de tener mayor longitud.

Por otra parte, el número de iteraciones también se ha incrementado para conseguir que el algoritmo converja en un valor. En este caso, se ejecutarán alrededor de unas 2.500 iteraciones. Otro aspecto reseñable es la mayor complejidad que tienen estas implementaciones respecto de las anteriores, ya que la duración de cada iteración es mayor que las iteraciones de los algoritmos genéticos del capítulo anterior.

Por último, ha sido necesario revisar algunos operadores genéticos, con el fin de ajustarlos a los objetivos de este apartado, utilizando dos frecuencias distintas dentro del mismo array. En este sentido, el operador que más se ve perjudicado es el operador de Mutación que funcionará de distinta forma que anteriormente, ya que en estas implementaciones debe variar el número de bits que se desee (1 ó 2 típicamente) entre tres valores posibles: 0, 1 (frecuencia f_1) y 2 (frecuencia f_2). Para la selección, en todas las simulaciones se ha escogido la Selección por Torneo, ya que se obtienen mejores resultados que si se utiliza la selección por Ruleta y además tiene un menor coste computacionalmente, por lo que tarda menos en ejecutarse.

Todo lo presentado anteriormente, se encuentra detallado a continuación en las simulaciones realizadas para el algoritmo genético.

4.4.1 Algoritmo Genético con frecuencias *f* y 1.5*f*

La primera implementación realizada para el Algoritmo Genético con selección por torneo se ha realizado utilizando un array con elementos compuestos por 0, 1 y 2, representando 1 para f_1 y 2 para f_2. En este caso, *f* será f_1 y 1.5*f*, f_2. Esto significa que la distancia entre elementos para la frecuencia f_1 será de d = 0.5λ y para la frecuencia f_2 de d = 0.75λ. Estos datos son muy importantes a la hora de calcular la función de *fitness* detallada anteriormente, donde se separa

el array original para generar dos arrays utilizando las dos frecuencias y que también sirven para representar el problema.

Para el cálculo de la *FC* del algoritmo, se definió la ecuación 4.5, en el apartado anterior. Para aplicar dicha ecuación hay que definir los pesos k_1 y k_2, que hacen que se le dé más importancia al $fitness_1$ o al $fitness_2$ (al array con frecuencia f_1 o al de frecuencia f_2). En este caso, se ha escogido como valores, k_1=0.6 y k_2=0.4, otorgando ligeramente más importancia al array con frecuencia f_1. Estos valores han sido determinados empíricamente tras haber lanzado un cierto número de veces el algoritmo.

Para mostrar el funcionamiento del algoritmo en la figura 4.2 se observa el decrecimiento del nivel de lóbulos secundarios para una ejecución del algoritmo durante 2500 iteraciones. Dicho nivel desciende desde un valor inicial de -12dB hasta un valor inferior a los -16dB.

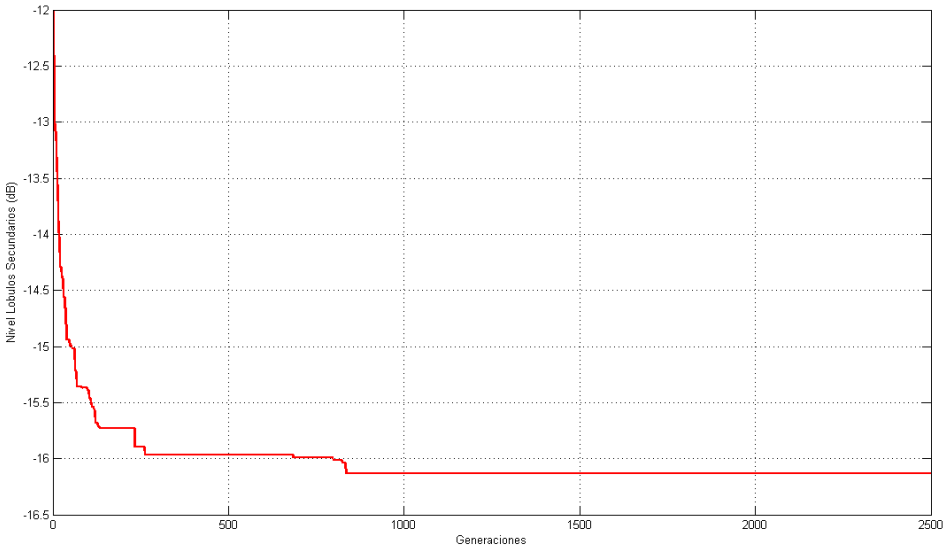

Figura 4.2: Relación de lóbulo principal a secundario con AG con selección por torneo y frecuencias *f* y 1.5*f*

El nivel que se obtiene, tal y como se ha explicado anteriormente, es el valor que se consigue tras sumar ponderadamente los valores del $fitness_1$ y $fitness_2$. A continuación, se pueden observar los diagramas de radiación iniciales (mejor individuo de la primera generación) para las dos frecuencias, figuras 4.3 y 4.4, donde se ve como el nivel de los lóbulos laterales se encuentra por encima de los -15dB en ambos casos.

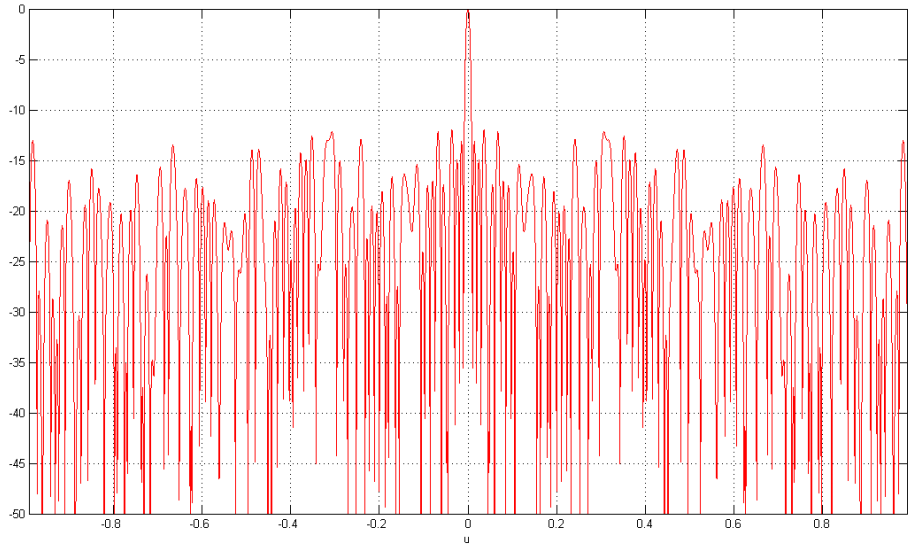

Figura 4.3: Diagrama de radiación inicial para AG con selección por torneo y frecuencia *f*

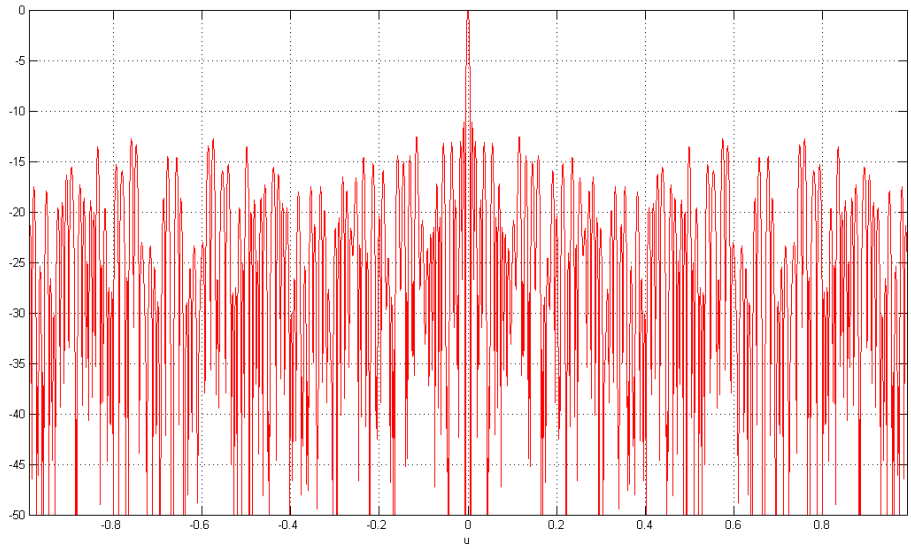

Figura 4.4: Diagrama de radiación inicial para AG con selección por torneo y frecuencia 1.5*f*

Tras aplicar el algoritmo genético, se consiguen los factores de array mostrados en las figuras 4.5 y 4.6, donde se puede observar como se ha logrado descender, en ambos casos, el nivel de los lóbulos secundarios por debajo de los -15dB.

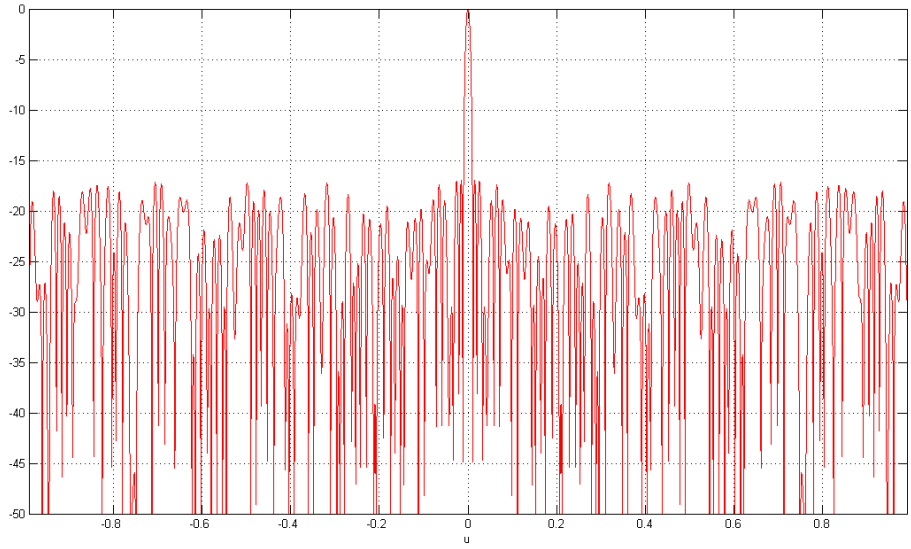

Figura 4.5: Diagrama de radiación final para AG con selección por torneo y frecuencia *f*

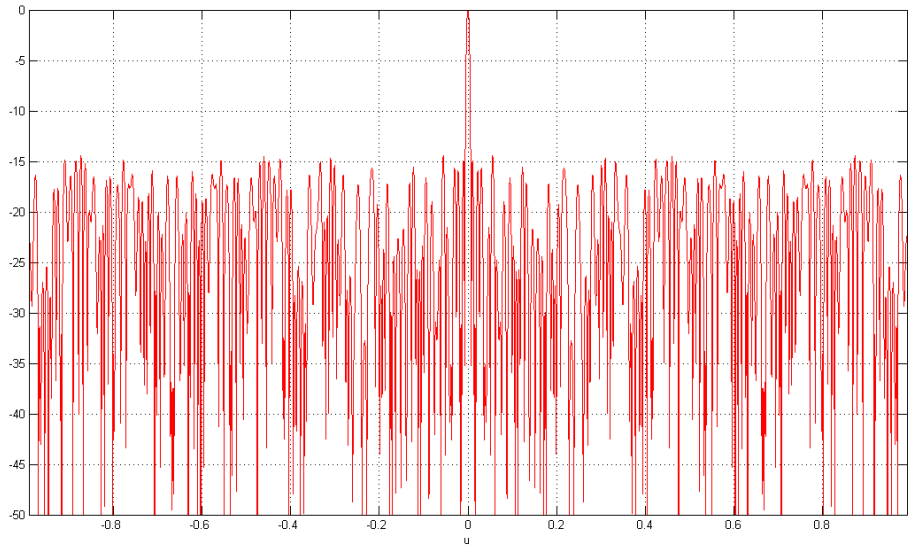

Figura 4.6: Diagrama de radiación final para AG con selección por torneo y frecuencia 1.5*f*

4.4.2 Algoritmo Genético con frecuencias *f* y 1.25*f*

En la segunda implementación llevada a cabo para el algoritmo genético se ha ejecutado utilizando una f_1 igual a *f* y una f_2 igual a 1.25*f*. De esta manera, la distancia entre elementos para el array con frecuencia f_1 será la habitual, d=0.5λ, y para el array con frecuencia f_2, en este caso será de d=0.625λ. En cuanto a los pesos k_1 y k_2, que se han de definir para calcular la *FC*, en este caso tomarán los valores k_1=0.3 y k_2=0.7, donde se da más importancia al array con

frecuencia 1.25*f*. Estos valores han sido determinados empíricamente tras haber lanzado un cierto número de veces el algoritmo.

De nuevo, ejecutando el algoritmo durante 2.500 iteraciones/generaciones se obtiene una convergencia como la de la figura 4.7 donde se puede observar el progreso del nivel de lóbulos secundarios. Se observa que este nivel desciende desde un valor inicial de unos -12dB hasta converger en un valor final de alrededor de -16.5dB.

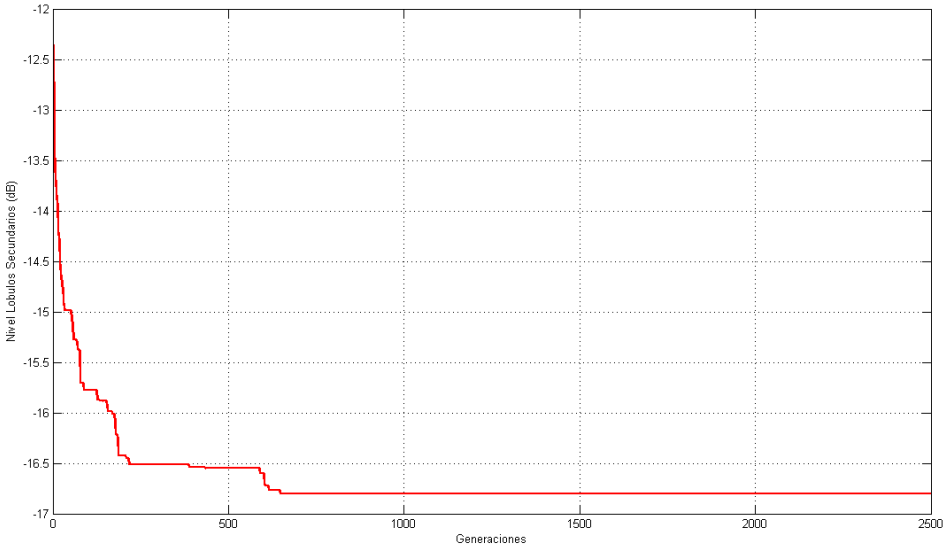

Figura 4.7: Relación de lóbulo principal a secundario con AG con selección por torneo y frecuencias *f* y 1.25*f*

A continuación, en los diagramas de radiación iniciales, figuras 4.8 y 4.9, se pueden observar los factores de array iniciales de los arrays y con los que se obtiene el valor con el que comienza la gráfica anterior. Concretamente, el valor inicial para el diagrama de radiación del array con frecuencia f_1 es -10dB, y para el diagrama del array con frecuencia f_2 el valor es de, aproximadamente, -14dB.

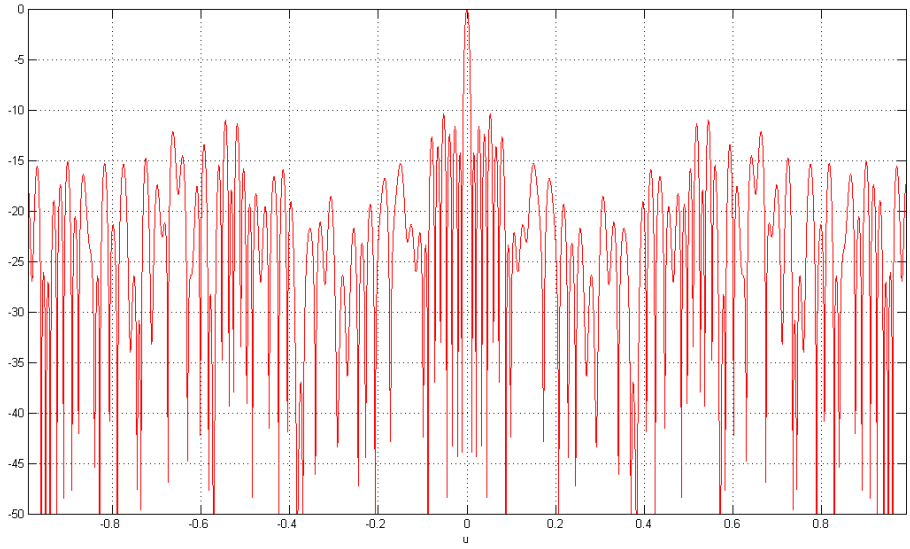

Figura 4.8: Diagrama de radiación inicial para AG con selección por torneo y frecuencia *f*

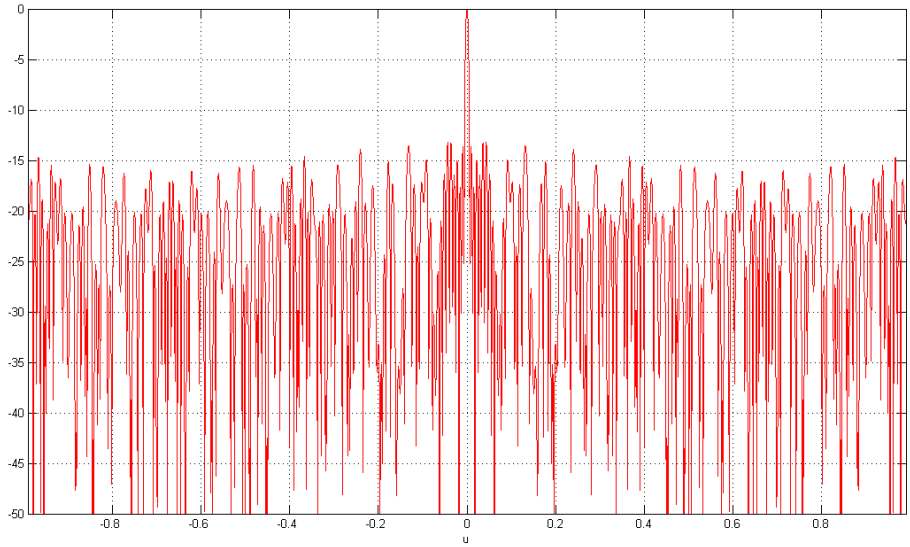

Figura 4.9: Diagrama de radiación inicial para AG con selección por torneo y frecuencia 1.25*f*

Por último, en los diagramas de radiación finales (mostrados en las figuras 4.10 y 4.11) se puede comprobar cómo el descenso final de lóbulos secundarios conjunto indicado en la figura 4.7.

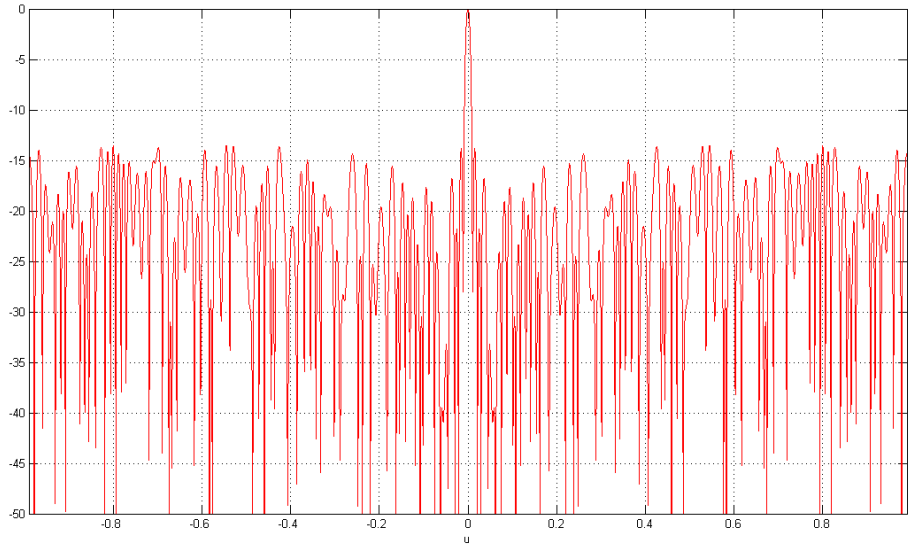

Figura 4.10 Diagrama de radiación final para AG con Selección por Torneo y frecuencia *f*

Figura 4.11: Diagrama de radiación final para AG con Selección por Torneo y frecuencia 1.25*f*

4.5. Implementación del Algoritmo PSO

El segundo y último algoritmo que se ha aplicado en este capítulo, es el Algoritmo Basado en Enjambres de Partículas (PSO). Se ha escogido este algoritmo en vez del Algoritmo Basado en Colonias de Hormigas por el hecho de que computacionalmente es menos costoso.

En todas las implementaciones de este apartado, al igual que el algoritmo genético, se va a simular para un grupo de 50 partículas colocadas en posiciones de 200 elementos de array cada una. El vector de posiciones es mayor que en las ejecuciones del capítulo 3 (array de única frecuencia), sin embargo, las iteraciones realizadas en este caso son menores aunque con un coste computacional más alto debido a la cantidad de operaciones realizadas en el algoritmo como consecuencia de tener elementos a distinta frecuencia dentro de cada array de posiciones.

Por último, en este caso se han tenido que modificar algunas funciones internas del algoritmo para adecuarlas a la utilización de arrays con elementos a dos frecuencias distintas, recordando que en estos arrays los valores posibles de amplitud de los elementos pueden ser 0, 1 y 2. Concretamente, una de las operaciones que ha tenido que ser ajustada, es el cálculo iterativo de las posiciones de las partículas ya que en este caso tiene que dar tres valores posibles para cada elemento, en vez de dos como ocurría anteriormente.

A continuación, se presentan ejemplos particulares de simulación realizadas para dos relaciones de frecuencia diferentes.

4.5.1 Algoritmo PSO con frecuencias *f* y 1.5*f*

En la única implementación llevada a cabo para el algoritmo PSO se ha ejecutado utilizando una frecuencia f_1 igual a f y una frecuencia f_2 igual a $1.5f$. De esta manera, la distancia entre elementos para el array con f_1 será la habitual, $d=0.5\lambda$, y para el array con f_2, en este caso será de $d=0.75\lambda$.

Por otro lado, para ejecutar el algoritmo, tal y como se vio en anteriores capítulos, se deben definir los parámetros de la ecuación (3.4), φ_1 y φ_2, a los que se les ha dado el valor de 0.5 a ambos. De esta manera, estamos dando la misma importancia a la mejor posición de la partícula a lo largo de las iteraciones como a la mejor posición global de todas las iteraciones.

En cuanto a los pesos k_1 y k_2, que se han de definir para calcular la *FC*, en este caso tomarán los valores k_1=0.5 y k_2=0.5, por lo que estamos otorgando la misma importancia tanto al array con frecuencia f_1 como al de frecuencia f_2.

De nuevo, ejecutando el algoritmo durante 2.000 iteraciones/generaciones se obtiene la figura 4.12 donde se puede observar el progreso del nivel de lóbulos secundarios. El nivel de lóbulos secundarios desciende desde un valor de -12.5dB hasta un nivel menor a -15.5dB. Observando la gráfica es reseñable como desciende inicialmente de forma progresiva el valor del nivel de los lóbulos y como a partir de las 500 iteraciones aproximadamente, durante muchos instantes de tiempo se mantiene en un valor constante hasta que, cuando llega a las 1.500 iteraciones, vuelve a descender el nivel.

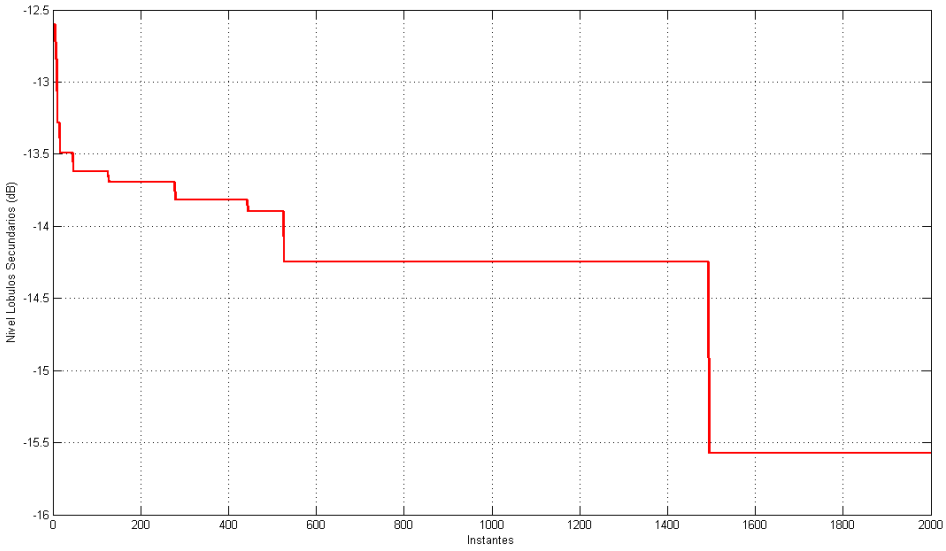

Figura 4.12: Relación de lóbulo principal a secundario con PSO con φ_1=0.5 y φ_2=0.5 y frecuencias *f* y 1.5*f*

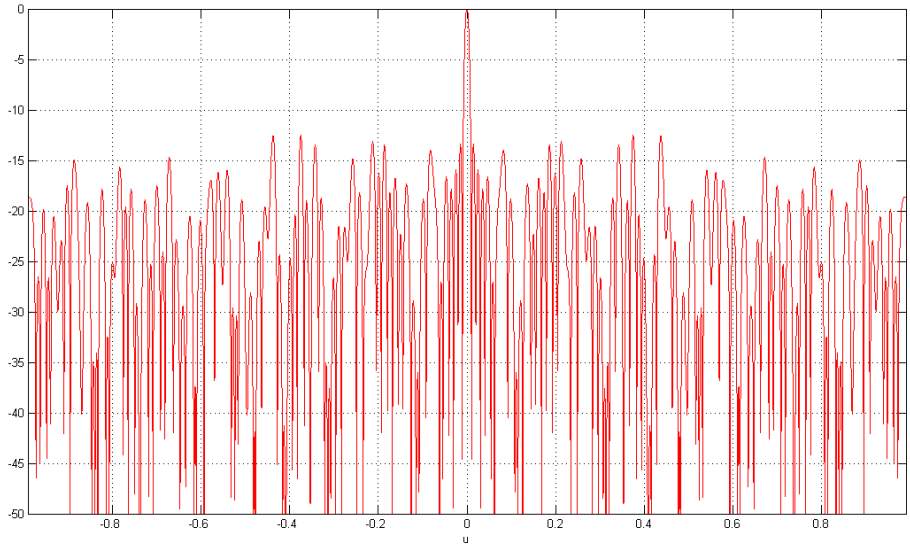

Figura 4.13: Diagrama de radiación inicial con APSO con φ_1=0.5 y φ_2=0.5 y frecuencia f

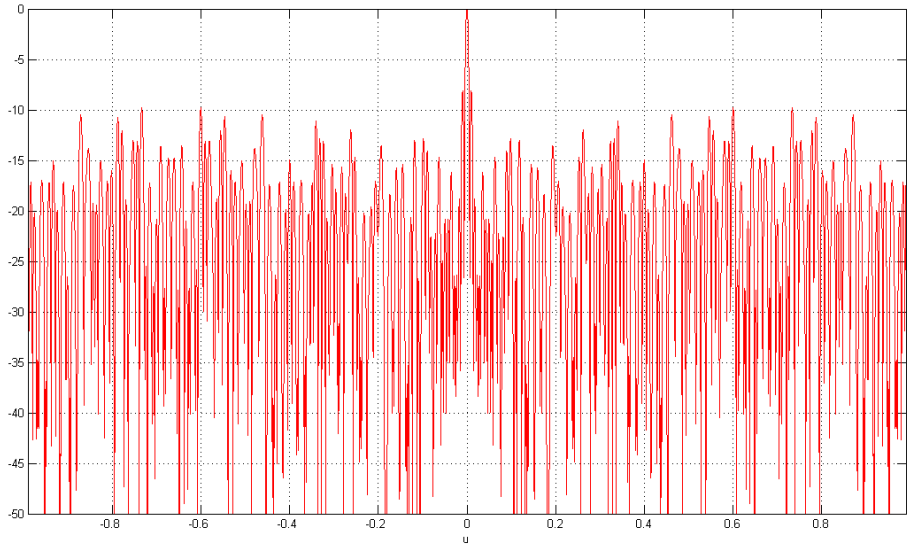

Figura 4.14: Diagrama de radiación inicial con APSO con φ_1=0.5 y φ_2=0.5 y frecuencia 1.5f

En las figuras 4.13 y 4.14, que representan los diagramas de radiación iniciales del algoritmo, se observa un nivel de lóbulos secundarios muy alto, sobre todo en el diagrama de radiación del array con frecuencia f_2, en el que los lóbulos se encuentran en un valor cercano a -10dB.

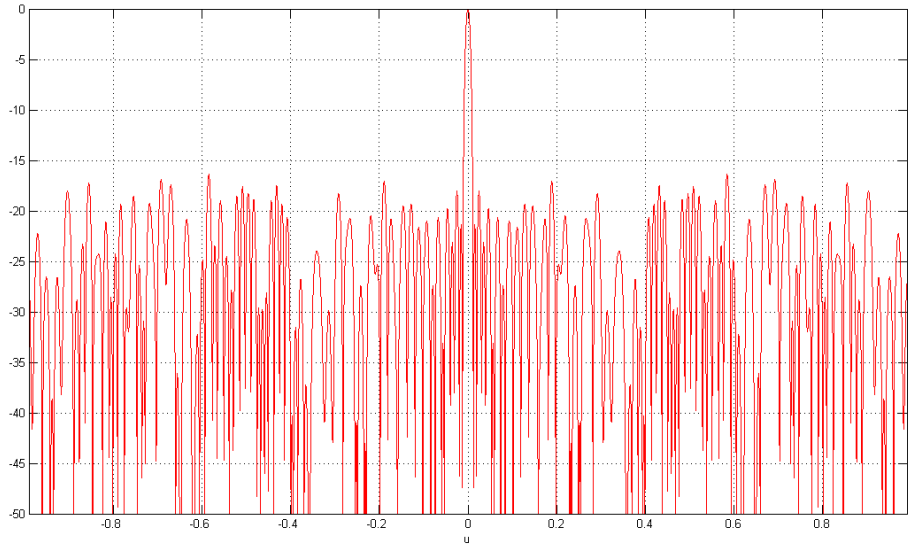

Figura 4.15: Diagrama de radiación final con APSO con φ_1=0.5 y φ_2=0.5 y frecuencia f

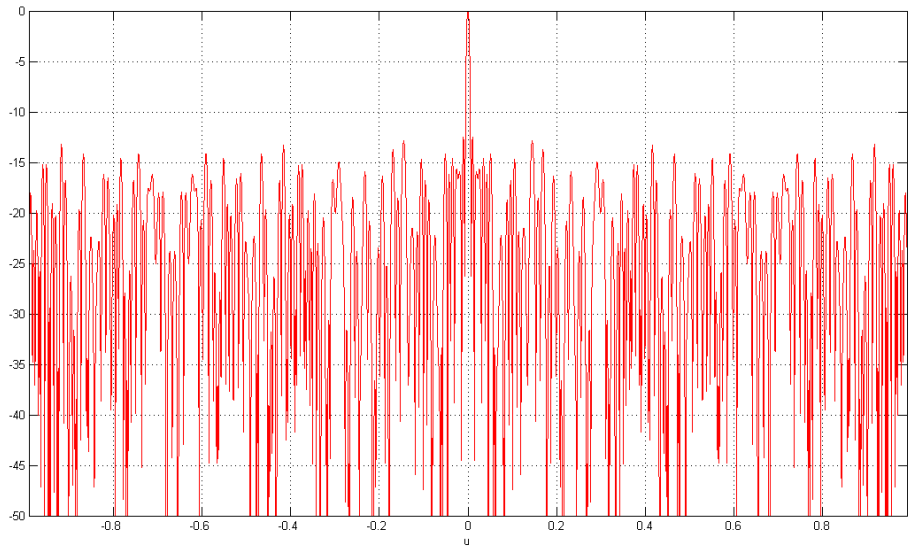

Figura 4.16: Diagrama de radiación final con APSO con φ_1=0.5 y φ_2=0.5 y frecuencia 1.5f

Finalmente, después de ejecutar el algoritmo se obtiene el diagrama de radiación mostrado en las figuras 4.15 y 4.16, donde se observa cómo los lóbulos secundarios han descendido notablemente.

4.6. Conclusiones

Aunque en este capítulo sólo se muestra un ejemplo de cada uno de los casos simulados, se han realizado diversas simulaciones dejando las más relevantes para la demostración y entendimiento de los objetivos. Así, si se observan los resultados obtenidos, se puede ver como en todos los casos, el algoritmo genético ofrece mejores resultados que el algoritmo PSO, además de ser mejor computacionalmente.

En general, todas las implementaciones para el AG consiguen descender el nivel de lóbulos secundarios utilizando la función *FC* hasta los -16.5dB, aunque el aspecto más importante de estos resultados es que se logra reducir el nivel de lóbulos secundarios de los diagramas de radiación obtenidos para las dos frecuencias que forman el array.

Por el contrario, las implementaciones realizadas para el algoritmo PSO, no logran descender el nivel de lóbulos secundarios por debajo de los -16dB, aunque sí que se logran obtener unos diagramas de radiación finales con un nivel de lóbulos menor que en los iniciales.

Capítulo 5. Conclusiones

En este capítulo se presentan las conclusiones extraídas de la realización de este proyecto, así como las líneas futuras que quedan abiertas.

5.1 Conclusiones

5.1.1 Síntesis de Arrays

En este proyecto se han estudiado las posibilidades de tres algoritmos de optimización global para el diseño de arrays entrelazados bifrecuencia. Para ello, el primer paso fue el estudio de dichos algoritmos para el caso de síntesis de *thinned arrays*, buscando como objetivo de diseño minimizar el nivel de lóbulos secundarios, Las conclusiones de dicho estudio se resumen a continuación

5.1.1.1 Algoritmo Genético

El primero de los algoritmos que se utilizó en este proyecto fue un algoritmo genético que está basado en los principios darwinianos de la transmisión genética. Esta transmisión se encuentra supeditada a los operadores genéticos tales como la selección, el cruce o la mutación, que hacen que los nuevos individuos de una generación no sean exactamente iguales que los progenitores.

En las simulaciones, se han implementado dos variantes del algoritmo respecto del operador genético de la selección: la selección por torneo y la selección por ruleta. Para nuestro caso particular de estudio la selección por torneo ha obtenido mejores resultados que la selección por ruleta, debido al hecho de que en esta última elije individuos menos aptos de la población y por lo tanto su convergencia es más lenta. Sin embargo, en relación a la solución final obtenida, ambas implementaciones han conseguido resultados parecidos con un nivel final de lóbulos secundarios por debajo de los -20dB.

5.1.1.2 Algoritmo PSO

El Algoritmo Basado en Enjambres de Partículas o PSO, fue el segundo algoritmo empleado en este proyecto. El algoritmo se inspira los movimientos producidos dentro de las bandadas de pájaros o los bancos de peces.

Computacionalmente, el algoritmo PSO ha demostrado ser mucho más costoso que el genético para nuestro caso particular, y por tanto, el tiempo que empleó en conseguir una solución óptima al problema propuesto fue ligeramente mayor debido a la cantidad de

comparaciones y operaciones que realiza, sobre todo en el cálculo de las nuevas posiciones de las partículas en cada iteración. En cuanto a los resultados alcanzados, aunque el algoritmo PSO ofrece buenos resultados, han sido peores que los obtenidos con el algoritmo genético.

5.1.1.3 Algoritmo Basado en Colonias de Hormigas

El último algoritmo estudiado en este proyecto, ha sido el Algoritmo Basado en Colonias de Hormigas (ACO). El algoritmo está basado en el movimiento de las hormigas para encontrar el mejor camino, entre el hormiguero y el alimento, utilizando los niveles de feromona que desprenden cada vez que pasan por un determinado lugar, finalizando el proceso en el momento que todas recorren la misma ruta para alcanzar la fuente de alimento.

La implementación del ACO se ha realizado mediante la inclusión de un grafo, donde cada nodo representa una posible solución. Las hormigas se mueven de un nodo a otro teniendo en cuenta la cantidad de feromona y la solución asociada, de este modo cada una va a evaluar los nodos adyacentes no visitados en cada paso que dé. Esto hace que el coste computacional asociado a este algoritmo sea mayor que el del genético y del PSO. Debido a este problema, en este caso, se han tenido que reducir tanto el número de hormigas que forman la población como el número de bits que componen cada solución. Esta es una de las desventajas del algoritmo, la imposibilidad de ampliar el área de aplicación, ya que cada bit que se añada a las posibles soluciones (nodos) va a aumentar en gran medida el número de nodos que compone el problema y, por tanto, el número de posiciones adyacentes donde se puede mover una hormiga en cada paso.

En cuanto a los resultados obtenidos tras la aplicación del algoritmo, las simulaciones han obtenido soluciones finales aceptables para el problema definido en un principio. Si se comparan estos resultados con los obtenidos con los algoritmos anteriores, genético y PSO, es el algoritmo genético el que obtiene mejores resultados globales. Sin embargo, debemos tener en cuenta que dado que el número de elementos de array era más reducido para este caso, también lo era el mínimo nivel de lóbulos secundarios que se puede teóricamente conseguir. .

5.1.2 Síntesis de Arrays Bifrecuencia

Tras el estudio de los algoritmos de optimización para el diseño de arrays monofrecuencia, se intentó aplicar dichas técnicas para reducir el nivel de lóbulos secundarios de thinned arrays bifrecuencia entrelazados.

En este caso, se ha utilizado una función de *fitness* conjunta (FC), que tenía en cuenta el nivel de lóbulos secundarios para cada una de las frecuencias, aplicándose una cierta ponderación a cada uno de ellos que fue determinada ad hoc para cada uno de los ejemplos.

Los algoritmos que se han usado para estas simulaciones han sido el genético y el PSO, ya que su coste computacional demostró ser menor que en el caso de colonia de hormigas. Se han estudiado dos casos, uno en el que la una de las frecuencias de operación es 1.25 veces la otra, y un segundo caso en el que era 1.5 veces mayor. En ambos casos y para ambos algoritmos se ha conseguido una disminución significativa del nivel de lóbulos secundarios. Sin embargo, el algoritmo genético, también en este caso obtiene, como norma general, mejores soluciones finales al problema.

5.2. Líneas Futuras

El presente proyecto ha abierto las diversas líneas futuras de trabajo. En primer lugar, el estudio de la síntesis basada en algoritmos de búsqueda global se ha realizado para elementos radiantes isotrópicos con excelentes resultados, sin embargo, es necesaria una futura extensión a elementos radiantes reales, como pueden ser antenas de parche. Del mismo modo, conviene no sólo un estudio en el que se incluya el diagrama de radiación de dichos elementos, sino el acoplo que cada antena produce sobre sus adyacentes, puesto que el acoplo entre elementos es un fenómeno que puede llegar a producir distorsiones en los diagramas de radiación finales y que siendo incluido en la síntesis, podría ser también eliminado o corregido mediante el uso del algoritmo de optimización global.

Extensión de la optimización de arrays bifrecuencia a multifrecuencia, que podría ir asociado a estudios de la anchura de banda de mantenimiento de las características optimizadas de los arrays. De esta forma, podrían estudiarse las posibilidades de diseño de arrays entrelazados de banda ancha mediante algoritmos de optimización, en las que las sub-bandas de funcionamiento de cada uno de los arrays se hicieran adyacentes y proporcionaran una banda total efectiva muy superior a las convencionales.

Referencias

[1] Mitchell, M. *An Introduction to Genetic Algorithms*. MIT Press. 1998.

[2] Rechenberg, I. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Fromman-Holzboog. 2001

[3] Schwefel, H.P. *Cybernetic Evolution as Strategy for Experimental Research in Fluid Mechanics*. Diploma Thesis, Univerisdad de Berlín. Marzo 1965

[4] Fogel, L.J., Owens, A.J., Walsh, M.J. *Artificial Intelligence through Simulated Evolution*. John Wiley & Son. 1966

[5] Holland, J.H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor. MIT press, 1975

[6] Algoritmo Genético. *http://sabia.tic.udc.es/mgestal/cv/AAGGtutorial/node1.html*

[7] Jong, K.A.D. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

[8] Kennedy, J., Eberhart, R. *Particle Swarm Optimization*. Proceedings of IEEE International Conference on Neural Networks. pp. 1942-1948.

[9] Particle Swarm Optimization. *http://www.swarmintelligence.org/*

[10] Bonabeau, E., Dorigo, M., Theraulaz, G. *Swarm Intelligence From Natural to Artificial Systems*. Oxford University Press. 1999

[11] Kennedy, J.; Eberhart, R.C. *Swarm Intelligence*. Morgan Kaufmann. 2001.

[12] Axelrod, R. The Complexity of Cooperation. Princeton University Press. 1997.

[13] Heppner, F., Grenander, U. *A Stochastic Nonlinear Model for Coordinated Bird Flocks*. Ed. Krasner. Washington DC. 1990.

[14] Latané, B. *The Psychology of Social Impact*. American Psychologist, 36, pp. 343 – 356, 1981.

[15] Reynolds, C.W. *Flocks, Herds and Schools: A Distributed Behavioral Model*. Computer Graphics, 21, pp. 25 – 34, 1987.

[16] Montes de Oca, M., Stützle, T., Birattari, M., Dorigo, M., *A Comparison of Particle Swarm Optimization Algorithms Based on Run-Length Distributions*. Ant Colony Optimization And Swarm Intelligence, 5TH International Workshop, pp. 1 – 12, 2006.

[17] Clerc, M., Kennedy, J. *The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space*. IEEE Transactions on Evolutionary Computation, pp. 58 – 73, 2002.

[18] Shi, Y., Eberhart, R.C. *A Modified Particle Swarm Optimizer*. Proceedings of the IEEE International Conference on Evolutionary Computation, Piscataway, NJ: IEEE Press, pp. 69-73, 1998.

[19] Eberhart,R. C., Shi, Y. *Tracking and optimizing dynamic systems with particle swarms*. Proceedings of the 2001 IEEE Congress Evolutionary Computation Piscataway, NJ, IEEE Press, pp. 94 – 100, 2001.

[20] Mendes, R., Kennedy, J., Neves, J. *The Fully Informed Particle Swarm: Simpler, Maybe Better*. IEEE Transactions on evolutionary computation, Vol. 8, pp. 204 – 210, Junio 2004.

[21] Dorigo, M., Di Caro, G., Gambardella, L.M. *Ant Algorithms for Discrete Optimization*. Artificial Life, Vol. 5, pp. 137 – 172, 1999.

[22] Deneubourg, J.L., Aron, S., Goss, S., Pasteels, J.M. *The Self-Organizing Exploratory Pattern of the Argentine Ant*. Journal of Insect Behavior, 3, pp. 159 – 168, 1990

[23] Pasteels, J.M., Deneubourg, J.L., Goss, S. *Self-Organization Mechanisms in Ant Societies: Trail recruitment to newly discovered food sources*. Experience Supplementum, 54, pp. 155 – 175. 1987

[24] Grassé, P.P. *La Reconstruction du Nid et les Coordinations Interindividuelles chez Bellicositermes Natalensis et Cubitermes sp. La Théorie de la Stigmergie: Essai d'interprétation du Comportement des Termites Constructeurs*. Insectes Sociaux, 6, pp. 41 – 81, 1959

[25] Cardama, M.A. *Antenas*. Ediciones Universidad Politécnica de Cataluña, 2001

[26] Haupt, R.L. *Thinned Arrays Using Genetic Algorithms*. IEEE Transactions on Antennas and Propagation, Vol. 42, pp. 993 – 999. 1994

[27] Haupt, R. *Thinned Interleaved Linear Array*. IEEE Trans. Antennas Propag., Vol. 53, No. 9, pp. 2858–2864, 2005.

[28] Balanis, C., *Antenna Theory: Analysis and Design*. John Wiley & Sons. 2005

[29] Dorigo, M., *Ant Colony Optimization*. Mit Press, 2004.

[30] Robles, M., *Sistemas de Optimización en el Diseño de Arrays de Antenas*. Proyecto Fin de Carrera de la Universidad Carlos III de Madrid, 2006

[31] Quevedo, O., Rajo, E. *Ant Colony Optimization in Thinned Array Synthesis With Minimum Sidelobe Level*. Antennas and Wireless Propagation Letters, IEEE. pp. 349 – 352, 2006.

[32] Haupt, R., Aten, D.W., *Low Sidelobe Arrays via Dipole Rotation*. IEEE Transactions on Antennas and Propagation. Vol. 57, pp. 1575 – 1579. Mayo 2009,

[33] Michalewicz, Z., *How to Solve It: Modern Heuristics*. Springer. 2004