Pontificia Universidad JAVERIANA Cali

Facultad de Ingeniería y Ciencias

Acta de Correcciones al Proyecto de Grado
Ingeniería Electrónica

Fecha: 28/01/2022

Autores: Juan José Mora Brito

Nombre del Proyecto de Grado: Conformado de Haz para MIMO Masivo usando Algoritmos de Optimización Híbridos

Director: Dr. Dimas Mavares

Como indica el artículo 2.27 de las Directrices de Trabajo de Grado, he verificado que los estudiantes indicados arriba han implementado todas las correcciones que los Jurados del Proyecto de Grado definieron que se efectuaran, como consta en el Acta de Calificación correspondiente.

Firma de Director(a) del Proyecto de Grado

Calle 18 No. 118 – 250, Piso 2, Facultad de Ingeniería y Ciencias – *Santiago de Cali, Colombia*
Teléfono +57 (2) 3218200 Ext. 85

Nota de Aceptación

Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar el título de Ingeniero Electrónico.

Dr. Hernán Camilo Rocha
Decano de la Facultad de Ingeniería

Dr. Luis Eduardo Tobón Llano
Director Carrera Ingeniería Electronica.

Dr. Dimas Mavares Terán
Director(a) Trabajo

Dr. Luis Eduardo Tobón Llano
Jurado 1

Dr. Andrés Felipe Amador
Jurado 2

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería
Ingeniería Electrónica
Trabajo de Grado

Conformado de Haz para MIMO Masivo usando Algoritmos de Optimización Híbridos

Juan José Mora Brito

Director: Dr. Dimas Mavares Terán

1 de febrero de 2022

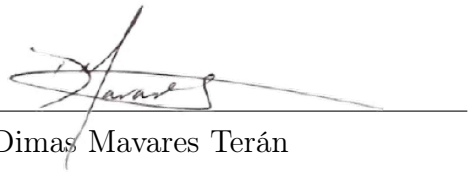
Santiago de Cali, 1 de febrero de 2022.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Luis Eduardo Tobón LLano
Director Carrera de Ingeniería Electrónica.
Cali.

Cordial Saludo.

Por medio de la presente me permito informarle que el estudiante de Ingeniería Electrónica Juan José Mora Brito (cod: 8935995) trabaja trabajan bajo mi dirección en el proyecto de grado titulado "Conformado de Haz para MIMO Masivo usando Algoritmos de Optimización Híbridos", y que este se encuentra terminado y listo para sustentación.

Atentamente,



Dr. Dimas Mavares Terán

Santiago de Cali, 1 de febrero de 2022.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Luis Eduardo Tobón LLano
Director Carrera de Ingeniería Electrónica
Cali.

Cordial Saludo.

Me permito presentar a su consideración el trabajo de grado titulado "Conformado de Haz para MIMO Masivo usando Algoritmos de Optimización Híbridos" con el fin de cumplir con los requisitos exigidos por la Universidad para optar al título de Ingeniero Electrónico.

Al firmar aquí, doy fe que entiendo y conozco las directrices para la presentación de trabajos de grado de la Facultad de Ingeniería aprobadas el 26 de Noviembre de 2009, donde se establecen los plazos y normas para el desarrollo del trabajo de grado.

Atentamente,

Juan José Mora Brito
Código: 8935995

Dedicatoria

Este trabajo va dedicado a mis padres, sin ustedes nada de esto habría sido posible. A Stephania por creer en mi de manera incondicional en estos últimos diez años. A Hannah y Juan Andrés por todo su cariño y apoyo fraternal. A Carlos, Alejandro, Abrahán y Toro por haberme acompañado y ayudado a crecer en diferentes etapas de mi vida.

Agradecimientos

A la Universidad Javeriana Cali, a la Universidad de los Andes (Mérida, Venezuela), al Dr. Dimas Mavares y a cada uno de las personas que de alguna manera me han ayudado en este proceso.

Resumen

En este trabajo se propone el diseño y aplicación de un algoritmo metaheurístico híbrido sobre el proceso de conformación de haces en un plano de antenas MIMO Masivo 8x8. Este algoritmo se ejecutó y se evaluó mediante una serie de experimentos desarrollados en MATLAB en los que se consideraron diferentes tipos y números de usuarios. El desempeño del nuevo algoritmo se evaluó en función del tiempo de ejecución, el número de iteraciones y la potencia promedio recibida por los usuarios en cada uno de los escenarios. Finalmente, se realizó una comparación entre el desempeño de los algoritmos metaheurísticos individuales con el nuevo algoritmo híbrido y se encontró que un algoritmo diseñado utilizando una estrategia de hibridación es capaz de dirigir mayor potencia a los usuarios en cada uno de los escenarios.

Palabras Clave: Beamforming, MIMO MASIVO, Algoritmo Metaheurístico, Estrategia de Hibridación.

Abstract

This investigation proposes the design and application of a hybrid metaheuristic algorithm on the beamforming process in a Massive 8x8 MIMO antenna plane. This algorithm was executed and evaluated through a series of experiments developed in MATLAB in which different types and numbers of users were considered. The performance of the new algorithm was evaluated based on the execution time, the number of iterations and the average power received by the users in each one of the scenarios. Finally, a comparison was made between the performance of the individual metaheuristic algorithms with the new hybrid algorithm and it was found that an algorithm designed using a hybridization strategy is capable of directing greater power to users in each of the scenarios.

Keywords: Beamforming, MASSIVE MIMO, Metaheuristic Algorithm, Hybridization Strategy.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Preliminares

1.1. Introducción

Los sistemas de comunicación han sido muy importantes para la humanidad y su desarrollo. Desde sus orígenes, el hombre ha buscado comunicarse y junto con el avance de la tecnología hemos visto como el desarrollo de estas permiten cada vez mejorar la forma en que nos comunicamos. Desde la última mitad del siglo XX hasta la actualidad, los sistemas de comunicación han tenido un crecimiento exponencial y en estos momentos nos encontramos a las puertas de la quinta generación de telefonía celular (5G).

Se espera que el 5G provea una comunicación inalámbrica más completa que las generaciones anteriores, brindando una red de baja latencia que sea capaz de transmitir grandes cantidades de datos (en el orden de gigabits por segundos) y al mismo tiempo tener un menor consumo energético. Esta nueva generación de comunicaciones celulares estará fundamentada en tecnologías que produzcan mejoramientos importantes en la tasa de transferencia efectiva de las celdas.[8]

En los últimos años, varios estudios se han centrado en sistemas masivos de múltiples entradas y múltiples salidas (MIMO), ya que se considera que juegan un papel importante en el despliegue del 5G. [9] Los sistemas MIMO masivo son sistemas en los cuales cada estación base contienen numerosas antenas. Este alto número de antenas permite alcanzar una eficiencia espectral más alta y una mejor eficiencia energética [9]. La tecnología multiple-input multiple-output (MIMO) ha sido ampliamente estudiada durante las últimas dos décadas y se ha aplicado a muchos estándares inalámbricos. El uso de estas tecnologías puede mejorar significativamente la capacidad y confiabilidad de dichos sistemas inalambricos, así como su eficiencia energética y eficiencia espectral[10]. Más allá de que sean temas que han sido estudiados por más de algunas décadas, actualmente continúan teniendo gran interés académico. Esto se debe a los beneficios que pueden aportar al mundo de las telecomunicaciones y el deseo de los investigadores de desarrollar redes cada vez más resilientes, eficientes y confiables.

Una de las técnicas más utilizadas en los arreglos de antenas MIMO masivos es el beamforming o conformación de haces. Desarrollada a principios de los años 90, el beamforming

es un tipo de procesamiento de señal utilizado en arreglos múltiples de antenas, ya sea del lado del transmisor o del receptor, con el fin de detectar de manera simultánea múltiples señales y así aumentar la capacidad del sistema y mejorar el rendimiento del sistema. Más allá de que el beamforming no sea un proceso del todo nuevo, esta técnica se presenta como una de las principales tecnologías que implementadas en conjunto con los arreglos de antenas MIMO masivo serán fundamentales para la consolidación de las redes 5G [4].

En está investigación se consideró un conjunto de algoritmos metaheurísticos previamente desarrollados los cuales fueron utilizados para diseñar un nuevo algoritmo híbrido. Este nuevo algoritmo se destinó a optimizar el proceso de beamforming en un arreglo de antenas 8X8 MIMO masivo y los resultados que se obtuvieron tras su aplicación son contrastados con la aplicación de cada uno de los algoritmos metaheurísticos por separado. Finalmente se analizaron los resultados obtenidos y se concluyó si la combinación de estos algoritmos representaba una mejora para el problema de conformación de haces propuesto.

1.2. Planteamiento del Problema

Desde sus inicios, el hombre ha tenido la necesidad de comunicarse. Con el tiempo se ha evidenciado como dicha necesidad es cada vez más y más exigente y como cada día se busca poder transmitir cantidades más grandes de información en periodos de tiempo cada vez más cortos. El desarrollo de las redes de telecomunicaciones ha sido fundamental para poder alcanzar estos objetivos, sin embargo, la cantidad de usuarios de dichas redes es cada vez más alta y se espera que esta siga creciendo de manera exponencial. Para el año 2009 se contaba con aproximadamente 900 millones de dispositivos conectados a la internet. Para el año 2020 son aproximadamente 20 mil millones [11].

En cuanto a Colombia, para el primer trimestre del año 2020 se tienen aproximadamente un número de 66,5 millones de abonados suscritos a la red de telefonía y una penetración del 130,6 dispositivos por cada 100 habitantes [8]. Comparando estas cifras con las del mismo periodo para el año 2013 tenemos que ha habido un aumento de mas de 20 millones en el número de usuarios (46,375,000) y la penetración también ha sufrido un aumento de aproximadamente 30 usuarios nuevos por cada 100 habitantes[12]. Como podemos ver el aumento en el numero de usuarios es cada vez mayor y se espera que este continué aumentado en el futuro.

Con alto número de usuarios es lógico que surjan problemas al momento en que todos estos dispositivos intenten recibir y transmitir datos. Cuando muchos dispositivos envían y

reciben datos se produce un congestionamiento de la red la cual conlleva a un throughput más pequeño o limitado [13].

Para afrontar este tipo de adversidades que presentan las redes con alto congestionamiento, se han desarrollado diversas propuestas que permitan optimizar el uso del espectro y por ende disminuir la tasa de transferencia efectiva[14] [15]. Una de las estrategias utilizadas para alcanzar estas características consiste en utilizar la misma frecuencia en enlaces de múltiples antenas (tanto en el transmisor como en el receptor) con el fin de mejorar la calidad y confiabilidad del enlace y sobreponerse a problemas de propagación e interferencia. A este concepto se le conoce como diversidad espacial[16]. Para poder lograr que esta técnica funcione de manera óptima se requieren antenas directivas que permitan dirigir la energía hacia el lugar donde se encuentren los usuarios. Si consideramos que los usuarios son estáticos pues no es más que utilizar antenas con alta directividad, sin embargo, con usuarios con una ubicación cambiante, ya sea de forma abrupta (usuarios nómada) o que cambien su ubicación de forma constante (usuarios en movimiento), es necesario utilizar otras estrategias para manejar el problema. A raíz de esto, surgen diversas técnicas dedicadas a resolver este problema, entre ellas una de las más destacadas es el uso del beamforming o conformación de haces en sistemas MIMO masivo [4].

El beamforming utiliza múltiples antenas con el fin de controlar la dirección del frente de onda controlando la magnitud y fase de las señales de antena individuales en una matriz de múltiples antenas. En otras palabras, la misma señal se envía desde múltiples antenas que tienen suficiente espacio entre ellas y así en cualquier ubicación dada, el receptor recibirá múltiples copias de la misma señal [17][18].

El uso del beamforming en sistemas MIMO masivo representa una posibilidad para mejorar la calidad de servicio, la eficiencia espectral, el área de cobertura, y las tasas de transmisiones de datos. La implementación de estos sistemas será fundamental en el desarrollo y despliegue de los sistemas 5G. Sin embargo, el beamforming en este tipo de sistemas presenta ciertas limitaciones, las cuales han hecho que estas tecnologías terminen siendo poco atractivas para el mercado. Una de las mayores limitaciones que se presentan es que se consideran un mayor número de elementos tanto en las antenas transmisoras como en las antenas receptoras lo cual se relaciona con los elevados costos de implementación. La segunda limitación es la complejidad matemática que representa la técnica del beamforming lo cual termina siendo muy demandante en términos de potencia y recursos computacionales.[19]

Una alternativa para solucionar estos retos que se presentan en la implementación de dichos sistemas es buscar una forma en que el procesamiento computacional no fuese tan in-

tensivo. En trabajos previamente realizados se ha encontrado que la aplicación de algoritmos de optimización inspirados en los mejores rasgos presentes en la naturaleza[20] pueden ayudar a optimizar dichos procesos [21], a este tipo de algoritmos se le conoce como algoritmos metaheurísticos[21]. Los algoritmos metaheurísticos son metodologías de alto nivel utilizadas para resolver problemas de optimización específicos[20]. El nombre de estos algoritmos deriva de la palabra heurístico que significa buscar y meta que hace referencia a alto nivel[21]. Estos algoritmos se caracterizan por tener componentes de exploración y explotación[20] los cuales se basan en procesos que típicamente se pueden encontrar en la naturaleza y ciertas abstracciones de esta, tales como el comportamiento de una colonia de hormigas, el recocido de materiales, el vuelo de un ave o algoritmos genéticos[22] [23].

Para intentar resolver el problema de los altos costos computacionales del beamforming, Polanco (2019) desarrolló una investigación donde analiza una colección definida de Algoritmos Metaheurísticos y determina cuales son los más adecuados para el Beamforming sobre un arreglo de este tipo. En esta investigación se considera la geometría física del arreglo, los tipos de usuarios y de acuerdo con la información obtenida por cada algoritmo se evalúa su tiempo de solución, cantidad de iteraciones, potencia en decibeles y diagramas de radiación optimizados. Posteriormente, utilizando esta información se plantea un análisis de los resultados obtenidos por cada algoritmo metaheurístico para determinar el desempeño de los algoritmos en los distintos escenarios planteados. Por último, se realiza una validación directa entre los esquemas de radiación obtenidos y los esquemas de radiación de software especializado para antenas.

Este proyecto se planteó como una continuación al trabajo de Polanco en donde se tienen en cuenta las mismas consideraciones y se utilizan los algoritmos con mejor desempeño para desarrollar un nuevo algoritmo híbrido. El nuevo algoritmo híbrido es aplicado sobre el mismo arreglo de antenas (MIMO masivo 8x8) utilizado por Polanco.

1.2.1. Formulación

¿La aplicación de un algoritmo metaheurístico híbrido presentará un mejor desempeño en la optimización del beamforming en un arreglo MIMO masivo 8x8 que la aplicación de varios algoritmos metaheurísticos de manera separada?

1.2.2. Sistematización

- ¿Cuál es el estado del arte del Beamforming con algoritmos metaheurísticos y de los algoritmos metaheurísticos híbridos ?

- ¿Qué aplicación han tenido los algoritmos metaheurísticos híbridos tanto en esta área como en otras?
- ¿Cuáles son los algoritmos metaheurísticos que presentan mejor desempeño para cada situación considerada en esta investigación (Número y tipo de usuarios)?
- ¿Cuáles son los pasos a seguir para poder desarrollar o diseñar un algoritmo metaheurístico híbrido ?
- ¿Cuál es una manera óptima de combinar los algoritmos seleccionados para desarrollar un nuevo algoritmos híbrido?
- ¿Cómo se comparan los resultados obtenidos para el nuevo algoritmo, con los resultados obtenidos de la evaluación de cada uno de los algoritmos de manera individual?
- ¿Qué información relevante y conclusiones se pueden extraer de los resultados obtenidos dentro del contexto del beamforming aplicado sobre un arreglo MIMO masivo 8x8 ?

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar un Algoritmo Metaheurístico híbrido y evaluarlo en diferentes escenarios de Beamforming, empleando antenas 8x8 MIMO masivo.

1.3.2. Objetivos Específicos

- Realizar una revisión de la bibliografía relacionada con el diseño de algoritmos metaheurísticos híbridos.
- Diseñar y aplicar una serie de situaciones en los cuales se varíe el número y tipo de usuarios con el fin de obtener los diagramas de radiación y algunas características adicionales, los cuales servirán como entrada al algoritmo a diseñar.
- Desarrollar un nuevo algoritmo que haga uso y combine los Algoritmos Metaheurísticos ya establecidos.
- Realizar pruebas en las que se compare el rendimiento del nuevo algoritmo con cada uno de los algoritmos metaheurísticos por separado en las diferentes situaciones previamente diseñadas.
- Analizar los resultados obtenidos y concluir cual de las dos alternativas funciona mejor y en que casos es útil este nuevo algoritmo.

1.4. Justificación

En este proyecto se planteó como objetivo principal desarrollar un Algoritmo Metaheurístico Híbrido y evaluarlo en diferentes escenarios de Beamforming sobre un plano de antenas 8x8 MIMO masivo. Estos algoritmos sobre los cuales se basa el nuevo algoritmo híbrido están diseñados con el objetivo de mejorar el proceso de conformación de haces y así reducir la interferencia entre celdas y aumentar la eficiencia espectral [24]. El desarrollo de este sistema continúa con el objetivo del proyecto anteriormente planteado y contribuye a resolver y mejorar los problemas que enfrenta el beamforming.

Tanto el beamforming como el uso de las antenas MIMO masivas han sido y seguirán siendo fundamentales en el desarrollo e implementación de los nuevas redes celulares 5G [10]. Adicionalmente en la actualidad el uso de las antenas MIMO se puede ver en sistemas como el 8x8 MIMO LTE outdoor CPE (customer-premises equipment) B3368 de la empresa China Huawei[25] o el sistema de antenas destinadas para el uso en transporte como lo son las LTE/WiFi MIMO antenna for transportation de Maxwave[26]. Esto demuestra un interés no solamente en el desarrollo de nuevos sistemas sino también una relevancia dentro de la industria de los dispositivos de comunicación.

Sin embargo el mayor impacto que podría tener este proyecto es contribuyendo al desarrollo de sistemas de comunicación. El objetivo de desarrollo sostenible No. 9 de las naciones unidas (Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación) establece como meta: Aumentar significativamente el acceso a la tecnología de la información y las comunicaciones y esforzarse por proporcionar acceso universal y asequible a Internet en los países menos adelantados de aquí al año 2030 [27]. Partiendo de esta idea, se ha encontrado que el uso de sistemas de MIMO masivo pueden ser utilizados con dispositivos que presenten alguna limitación o impedimento lo cual se relaciona con mejorar el acceso a las tecnologías de red. [28].

En secciones anteriores se habló acerca del incremento de dispositivos móviles en Colombia, sin embargo es interesante observar que la distribución de estos dispositivos no es del todo equitativa. Para el año 2018 según cifras del Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia [29] aproximadamente 23,8 millones de personas no cuentan con acceso a internet de ningún tipo. Según el Plan Nacional de Desarrollo para el periodo de 2018-2022 publicado por el gobierno colombiano se espera acelerar la inclusión social digital a través de incentivos al despliegue de redes para llegar a los hogares más necesitados [30]. Como podemos ver, los objetivos de este proyecto se alinean con estas propuestas y por ende un desarrollo de redes más eficientes y más resilientes puede significar mejores

sistemas de comunicación y la posibilidad de que una mayor cantidad de personas tengan acceso a redes móviles y sistemas de internet.

Este proyecto tendrá un impacto positivo desde el punto de vista social, industrial y académico y se realiza con la intención de que los resultados obtenidos puedan contribuir al desarrollo de sistemas de comunicación y al despliegue de las nuevas redes móviles.

1.5. Delimitaciones y Alcances

En el caso de este proyecto se diseñaron un número limitado de situaciones o escenarios en los cuales se observó el diagrama de radiación optimizado generado a partir del algoritmo híbrido. Considerando los altos costos de implementación, estos experimentos se llevaron a cabo utilizando únicamente el lenguaje de computación numérica matricial MATLAB. Partiendo de esto para cada experimento se tomaron en cuenta las siguientes consideraciones:

- Una misma banda de frecuencia
- Un número de algoritmos metaheurísticos menores o iguales a 5, los cuales se seleccionaron en función de su importancia o aceptación en el área.
- Un número menor a 4 usuarios receptores ubicados de manera equidistante.
- Dos tipos de usuarios. Se consideran usuarios que siempre mantienen la misma posición (Estáticos) y usuarios que presenten un cambio abrupto de posición y se mantienen en este estado por un periodo considerable de tiempo (Nómadas).
- No se considerarán usuarios que estén en constante movimiento.

Capítulo 2

Desarrollo del Proyecto

2.0.1. Marco Teórico

A continuación se presenta el marco teórico bajo el cual la teoría de este proyecto se encuentra fundamentada. Para comenzar es importante saber que son los arreglos planos de antenas y su relación con los sistemas de antenas MIMO Masivo.

2.0.1.1. Arreglos planos de Antenas

Los arreglos planos de antenas son un conjunto de antenas que se ubican sobre una geometría rectangular o plana. A diferencia de las antenas de un solo elemento, estos tipos de antenas proveen variables adicionales las cuales pueden ser usadas para controlar y modificar la forma del diagrama de radiación. Los arreglos planos son más versátiles y pueden proporcionar patrones más simétricos con lóbulos laterales inferiores. Al momento de hablar de arreglos planos es importante hablar de su factor de arreglo [1].

El factor de arreglo es el patrón de radiación de campo lejano para una matriz de N elementos isotópicos radiantes el cual se puede definir como una función de la geometría del arreglo y de la fase. Cada arreglo tiene su propio factor de arreglo y viene dado en función del número de elementos, su geometría, sus magnitudes relativas, sus fases relativas y la distancia entre cada elemento [1].

Considerando que para un arreglo lineal su factor de arreglo vendrá dado por

$$AF = \sum_{m=1}^{M} I_{m1} e^{j(m-1)(kd_x \sin\theta \cos\phi + \beta_x)} \tag{2.1}$$

donde son Im son los pesos complejos o coeficientes de excitación complejos de cada elemento m, dx es la distancia entre elementos en x, β_x es el cambio de fase entre los elementos y $k = \frac{2\pi}{\lambda}$ el número de onda[1]. Si se considera un arreglo plano compuesto por M elementos a lo largo del eje X y N elementos radiantes ubicados a lo largo del eje Y tal como se puede observar en la figura 2.1

El factor de arreglo vendrá dado por el producto entre el factor de arreglo lineal para el eje X y el factor de arreglo lineal correspondiente al eje y [1].

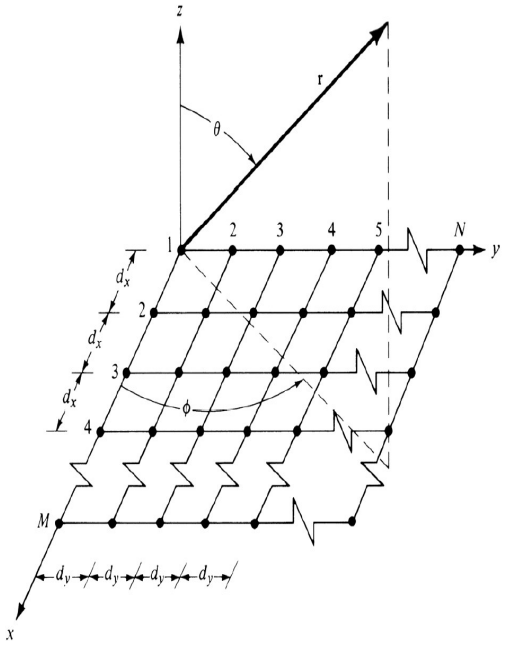


Figura 2.1: Geometría arreglo plano [1]

Con lo cual tendremos:

$$AF = AF_{xm}AF_{yn} = \sum_{m=1}^{M} a_{m1}e^{j(m-1)(kd_x \sin\theta\cos\phi+\beta_x)} \sum_{n=1}^{N} b_{n1}e^{j(n-1)(kd_y \sin\theta\sin\phi+\beta_y)} \tag{2.2}$$

donde W_{mn} son los vectores complejos definidos como:

$$W_{mn} = a_{m1}b_{n1}e^{j[(m-1)\beta_x)+j(n-1)\beta_y]} \tag{2.3}$$

y por lo tanto se obtiene que el vector del factor de arreglo vendrá dado por la ecuación:

$$AF = \sum_{m=1}^{M}\sum_{n=1}^{N} W_{mn}e^{j[(m-1)(kd_x \sin\theta\cos\phi)+(n-1)(kd_y \sin\theta\sin\phi)]} \tag{2.4}$$

Las cambios de fase β_x y β_y son independientes entre si y se pueden ajustar de tal manera que el lóbulo principal producido por AF_{xm} sea diferente al producido por AF_{yn}. Sin embargo, en diversas aplicaciones practicas se puede requerir que estos lóbulos principales apunten en la misma dirección. Si se desea tener un solo lóbulo principal los cambios de fase progresivos β_x y β_y deben cumplir la relación.[1]

$$\beta_x = kd_x \sin\theta\cos\phi \tag{2.5}$$

$$\beta_y = kd_y \sin\theta \sin\phi \tag{2.6}$$

2.0.1.2. MIMO Masivo y Beamforming o Conformación de Haces

Para realizar este proyecto hemos propuesto el uso de antenas MIMO Masivo 8x8 las cuales se usarán para llevar a cabo los experimentos que se desean implementar. La selección de este tipo de antenas se basa en la relevancia que están teniendo en la actualidad y las aplicaciones que se le pueden dar en la quinta generación de redes celulares y el interés académico que existe por el tema [10].

Tomando esto en cuenta los sistemas MIMO Masivo se pueden definir como sistemas de múltiples antenas (arreglo plano de antenas) en los cuales cada estación base contiene numerosas antenas. Este alto número de antenas permite alcanzar una eficiencia espectral más alta y una mejor eficiencia energética [2]. Este tipo de antenas permite modificar el diagrama de radiación lo cual quiere decir que permite dirigir la potencia irradiada en cualquier dirección deseada.

A continuación se presenta una idea de como se puede modelar el canal de subida (uplink) y el canal de bajada (downlink) de un sistema de MIMO masivo considerando que la señal recibida en la estación base j será y_j y en donde n_j es el ruido aditivo Gaussiano, $s_l k$ es la información enviada transmitida y h_{lk}^j es el vector que representa el canal entre la estación base j y el usuario k [2].

$$y_j = \sum_{l=1}^{L} \sum_{K=1}^{Kl} h_{lk}^j s_{lk} + n_j \tag{2.7}$$

el siguiente dibujo a continuación ilustra de manera adecuada el enlace de subida.

en cuanto al enlace de bajada la estación base en la celda l transmite la señal donde ζli es la señal de datos y W_{li} es el vector de peso que determina la directividad espacial de la transmisión [2].

$$x_l = \sum_{i=1}^{Kl} W_{li} \zeta li \tag{2.8}$$

Tomando en cuenta la directividad que se pretende alcanzar con este tipo de antenas se plantea la idea del beamforming. El beamforming se puede definir como la conformación del haz de la onda electromagnética, en la cual se busca dirigir el haz generado por el arreglo hacia usuarios puntuales, aumentando así la potencia recibida y mejoras de la eficiencia espectral en todo el sistema [4]. A continuación en la figura 2.4 se aprecia una imagen que ilustra

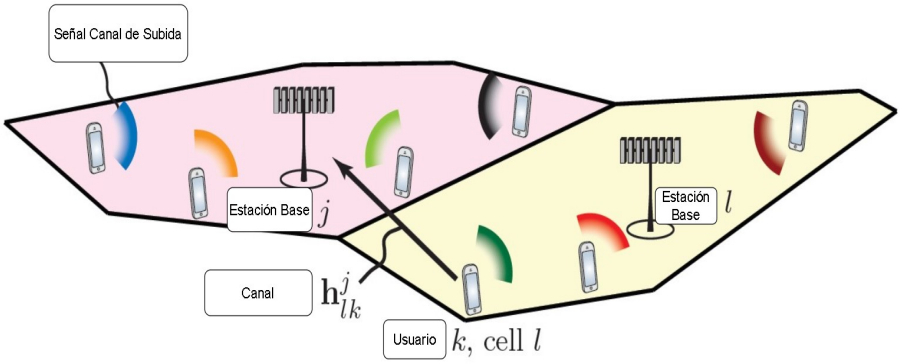


Figura 2.2: Uplink sistema MIMO masivo [2]

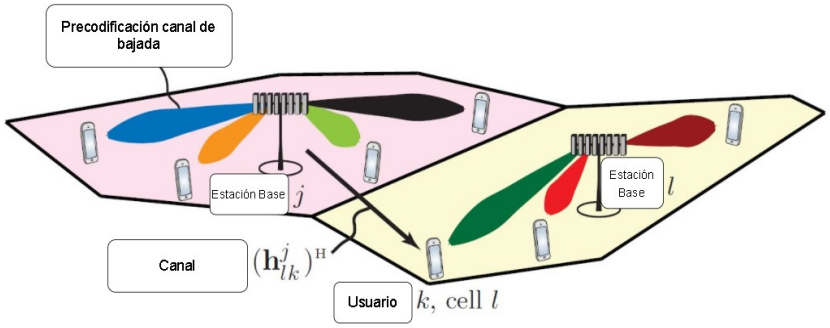


Figura 2.3: Downlink de sistema MIMO Masivo [2]

la comparación entre el beamforming en sistemas mimo masivo con los sistemas celulares tradicionales:

Las técnicas de beamforming son diversas y entre estas se pueden destacar los esquemas de switch y los esquemas adaptativos. Los esquemas de switch son aquellos que dependen de una red de beamforming conformada por rayos predeterminados. Mientras que los esquemas de beamforming adaptativo se destacan por tener la capacidad de enviar un haz directo a cada usuario. Esta característica se logra mediante el uso de vectores de peso que se aplican a las señales detectadas con el objetivo de controlar el ensanchamiento de la amplitud y los cambios de fase entre los elementos de la antena. En esta técnica de beamforming es posible crear formas específicas, y direcciones especificas para dirigir el haz hacia una estación móvil o usuario preferido [4].

En los arreglos de antenas planos las señales recibidas en cada elemento de la antena son construidas para mejorar la eficiencia del sistema de comunicación. Las señales detectadas en las diferentes antenas de la estación base pasan a través de un proceso de multiplica-

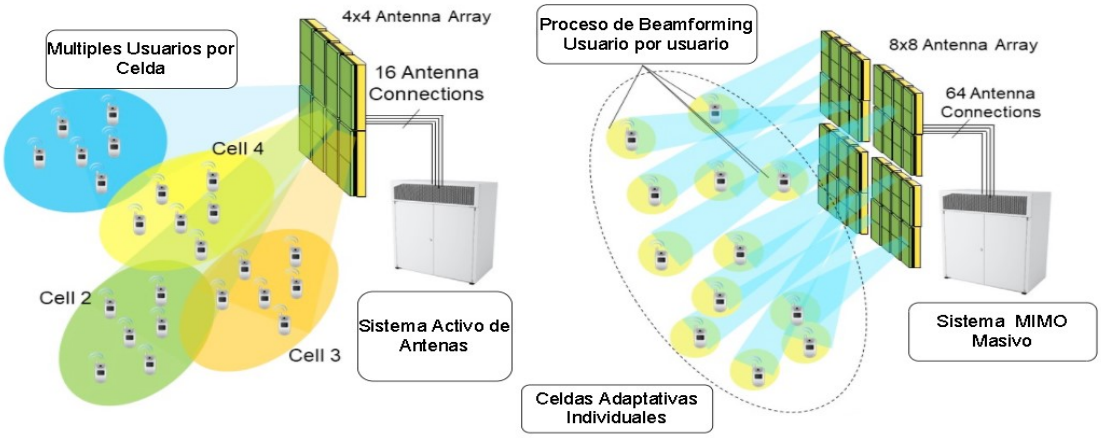


Figura 2.4: Beamforming MIMO Masivo [3]

ción con los pesos complejos y posteriormente son sumados. En los arreglos de beamforming adaptativo dependiendo de la dirección de llegada (DOA:direction of arrival) el haz estimado puede ser dirigido en la dirección en la que se quiere que se irradie la señal mientras que al mismo tiempo se aplica un proceso de cancelación a la señales que van en las direcciones no deseadas. Las antenas inteligente son capaces de estimar las direcciones de llegada y luego utilizando un algoritmo aplicado al beamforming el haz de la antena puede ser dirigido en la dirección deseada [4].

Estos algoritmos se pueden clasificar en dos tipos principales: algoritmos adaptativos ciegos y algoritmos adaptativos no ciegos. Los algoritmos adaptativos no ciegos requieren estadísticas conocidas de la señal transmitida, para poder determinar una ruta de viaje ponderada. Este objetivo se puede lograr típicamente usando una señal de entrenamiento que se transmite a través del enlace de comunicación a los terminales para apoyar la detección de los usuarios. Por el contrario, los algoritmos adaptativos ciegos no requieren ningún conocimiento estadístico para su funcionamiento; esto se expresa claramente con el término çiego". El enfoque de los algoritmos ciegos es maximizar la señal al terminal deseado y minimizar la interferencia de otros terminales sin tener ningún tipo de información acerca de los usuarios [4]. La siguiente ecuación ilustra la salida de la matriz adaptativa $y(t)$ que viene dada por:

$$y(t) = w^H x(t) \tag{2.9}$$

donde w^H es el complejo conjugado transpuesto del vector de pesos complejos y $x(t)$ es la señal de entrada. Los pesos son calculados de manera iterativa basados en el arreglo $y(t)$. En los algoritmos tanto ciegos como no ciegos, muchas veces una señal de referencia es utilizada

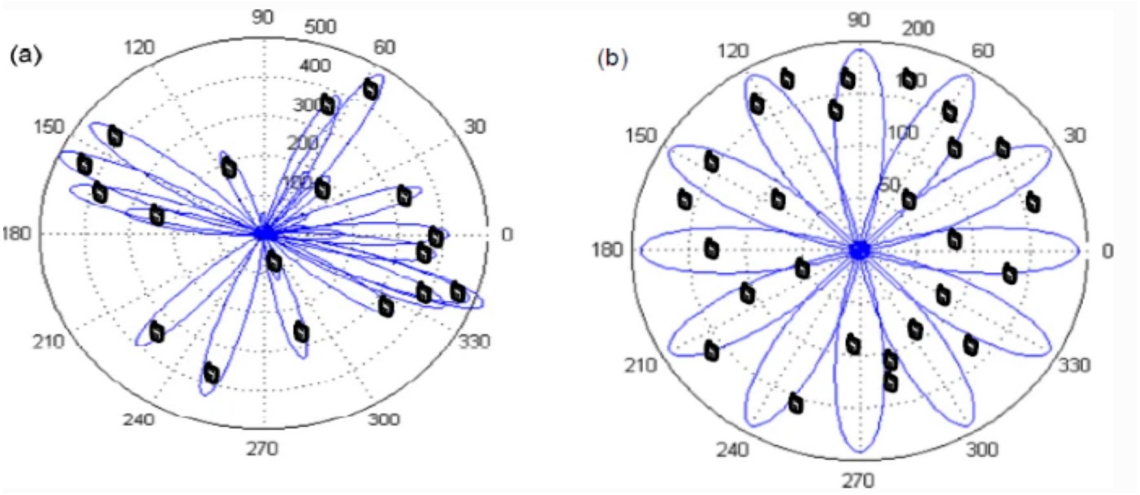


Figura 2.5: Beamforming Adaptativo y Switch Beamforming [4]

para adaptar el arreglo de pesos complejos de manera continua, es decir, la respuesta de los pesos al final de cada iteración se compara con la señal de referencia y la señal de error producida es utilizada para ajustar el algoritmo. La señal de error viene dada por [4].

$$e(t) = d(t) - w^H x(t) \tag{2.10}$$

donde $d(t)$ es la señal de referencia con la cual el algoritmo busca minimizar el error. En el caso de este proyecto se trabajó con técnicas de beamforming adaptativo ya que se realizó un control sobre el direccionamiento del haz y un algoritmo adaptativo ciego ya que no necesitamos alguna información estadística previa de los usuarios para poder determinar la dirección de la señal.

2.0.1.3. Normatividad de los sístemas MIMO Masivo en el 5G

Como ya hemos mencionado, las antenas MIMO masivo, en conjunto con el beamforming son pilares fundamentales en el despliegue de las tecnologías 5G. Sin embargo, estas deben dentro de las regulaciones establecidas por los diferentes entes reguladores de telecomunicaciones a nivel mundial. En caso de la unión europea, los puntos de acceso utilizados en el despliegue de esta nueva tecnología deben garantizar la protección de la salud y la seguridad de las personas por lo cual la potencia utilizada por estos equipos debe ser menor a 50 veces a lo que indica la evidencia científica internacional que puede tener efectos sobre la salud[31]. Las nuevas antenas deben ser menos visibles al ojo del público con un tamaño que no debe superar los 30 litros y ubicarse a una altura mínima de 4 metros en espacios cerrados [31].

Adicionalmente la norma EN 62232: 2017 se continua aplicando a todo tipo de estaciones base divididas en cinco clases de instalación correspondientes a diferentes límites de su potencia radiada isotrópica equivalente (EIRP). Siendo estas: de unos pocos milivatios (clase E0), 2 vatios (clase E2), 10 vatios (clase E10), 100 vatios (clase E100) y más de 100 vatios (clase E +) respectivamente. [31].

En cuanto a Estados Unidos la FCC (Federal Communication Commission) está trabajando sobre tres pilares fundamentales para el despliegue del 5G. Aumentar la oferta de espectro en el mercado y asignarla; actualizar la forma en que están estructuradas las políticas de y modernizar las regulaciones obsoletas. En cuanto al primer tema se establecieron los espectros para alta frecuencia como 24GHz, 37GHz, 39 GHz, 47GHz de 3.7 a 4.2 GHz para las bandas medias y 600 MHz, 800 MHz y 900 MHz. En cuanto a las nuevas políticas se busca implementar nuevas reglas que reduzcan los obstáculos legales para así lograr el despliegue de celdas pequeñas necesarias para el 5G de una manera mas rápida y efectiva; La modernización de las normas obsoletas se enfoca en garantizar la libertad y seguridad en las nuevas redes, actualizar las normas para que permitan el uso de postes de servicios públicos para implementar la nueva red de equipos que acelerará el proceso de implementación de 5G; Implementación de estrategias para incentivar la inversión en redes de fibra modernas por parte de las grandes empresas [32].

En el caso de Colombia se indica en el plan de despliegue de las tecnologías 5g elaborado por parte del Ministerio de Tecnologías de la Información y las Comunicaciones: “Con la tecnología 5G, se vienen grandes retos en la revisión y actualización de las medidas para la protección de los derechos de los usuarios y el régimen de calidad”[33]. Adicionalmente es importante destacar que el uso de estas nuevas tecnologías y sus efectos sobre la salud de los seres humanos no son conocidas a profundidad ya que los estudios acerca de los efectos a largo plazo aún se encuentran en desarrollo [34]. A pesar de esto los sistemas desarrollados en el proyecto van orientados a ser utilizados dentro de las regulaciones expuestas previamente.

2.0.1.4. Algoritmos metaheurísticos

A pesar de las ventajas que el beamforming pueda presentar, sus costos de implementación y en especial sus altos requerimientos computacionales son dos de las principales limitantes que esta tecnología presenta. Para poder afrontar este problema hemos decidido implementar algoritmos de optimización basados en fenómenos de la naturaleza.

Los algoritmos metaheurísticos son metodologías de alto nivel utilizadas para resolver problemas de optimización específicos[20]. El nombre de estos algoritmos deriva de la palabra heurístico que significa buscar y meta que hace referencia a alto nivel[21]. Estos algoritmos

se caracterizan por tener componentes de exploración y explotación[20] los cuales se basan en procesos que típicamente se pueden encontrar en la naturaleza y ciertas abstracciones de esta, tales como el comportamiento de una colonia de hormigas, el recocido de materiales, el vuelo de un ave o algoritmos genéticos [22] [23]. Dichos algoritmos han tenido diversas aplicaciones dentro del mundo de la optimización de procesos tales como en la identificación de fallas de un sistema de producción, diseño de unidades de procesamiento gráfico y predecir la capacidad resistiva de conexiones en estructuras compuestas[35].

Cuando se habla de algoritmos de optimización estos se pueden clasificar en dos categorías: algoritmos deterministas y algoritmos estocásticos. Los algoritmos deterministas siguen procedimientos estructurados y sus funciones son repetibles. Un ejemplo de esto podría ser subir una montaña, en este proceso siempre se tendrá el mismo inicio, se recorrerá el mismo camino y el resultado final será el mismo. A diferencia de los algoritmos deterministas, los algoritmos estocásticos siempre involucran algún grado de aleatoriedad, como sucede en el caso de los modelos que describen comportamientos de poblaciones, los cuales al ser simulados obtienen resultados ligeramente diferentes al final de cada ejecución [21].

Los algoritmos metaheurísticos presentan dos componentes: la intensificación (explotación) y diversificación (exploración). La diversificación hace referencia a la capacidad de generar diversas soluciones mientras se explora una escala global lo cual ayuda a evitar encontrar las mejores soluciones únicamente sobre los máximos locales. La intensificación se enfoca en la búsqueda de soluciones óptimas sobre regiones locales. Ambas componentes están relacionadas con la selección de las mejores soluciones las cuales serán aquellas que converjan de manera óptima. La combinación de todos estos elementos ayuda a que se puedan conseguir soluciones óptimas globales [21].

Algoritmos Utilizados por Polanco (2021)

En la investigación desarrollada por Polanco (2021) [24] titulada "Beamforming de arreglos planos MIMO Masivo usando optimización metaheurística". Se presenta una colección de algoritmos los cuales son utilizados para evaluar su comportamiento. A continuación se presentan de manera resumida los algoritmos utilizados por el autor, su inspiración, su origen, y su funcionamiento.

Recocido Simulado (SA) Uno de los primeros y más populares algoritmos metaheurísticos es el recocido simulado (SA). Este algoritmo es una técnica de búsqueda aleatoria basada en trayectorias para la optimización global. Este algoritmo está inspirado en el proceso de recocido presente en el procesamiento de materiales. Un ejemplo de esto es cuando

un metal se enfría y se congela en un estado cristalino con la energía mínima con el fin de reducir los defectos en las estructuras metálicas. El proceso de recocido implica el control cuidadoso de la temperatura y su velocidad de enfriamiento, a menudo llamado cronograma de recocido [21].

A diferencia de los métodos basados en gradientes y otros métodos de búsqueda deterministas que tienen la desventaja de quedar atrapados en mínimos locales, la principal ventaja de SA es su capacidad para evitar quedar atrapado en dichos mínimos. De hecho, se ha demostrado que el recocido simulado convergerá a su óptimo global si se utiliza un grado suficiente aleatoriedad en combinación con un enfriamiento muy lento. Esencialmente, SA es un algoritmo de búsqueda como una cadena de Markov, que converge en las condiciones adecuadas [21].

El pseudocódigo a continuación describe el proceso o los pasos que lleva a cabo el algoritmo. Para comenzar se requieren los parámetros que serán utilizados por el algoritmo y se inicializan las variables a utilizar. Posteriormente, se calcula cual es la mejor solución de el conjunto de soluciones iniciales y luego de esto se procede a generar nuevas posiciones que generarán nuevas soluciones. Seguidamente se evaluará si la mejor solución nueva cumple con el criterio de metrópolis o si la diferencia negativa entre la solución nueva y la mejor solución global es mayor a la tolerancia. En caso de que se cumpla alguno de estos dos criterios, esta será la nueva solución global. En base a la cantidad de aceptaciones que el algoritmo va teniendo, la temperatura irá disminuyendo según la siguiente ecuación 2.11 donde α es la constante de enfriamiento y T_0 es la temperatura inicial.

$$T = \alpha T_0 \tag{2.11}$$

Este proceso se repite de manera iterativa hasta que se alcanza alguno de los criterios de parada.

Cold Simulated Annealing (CSA): En SA la selección de la temperatura es importante ya que si la temperatura es muy alta, la probabilidad de aceptación P tenderá a 1 y todas las soluciones serán aceptadas, mientras que si la temperatura T es muy baja entonces la probabilidad P tenderá a cero y por ende se pierde diversidad en las soluciones aceptadas. En otras palabras, el caso en que T tiende a cero describe el comportamiento similar al de un algoritmo de gradiente ya que únicamente se aceptan las mejores soluciones. Sin embargo este comportamiento de tomar la mejor solución es de interés para nuestra investigación ya que representa una posibilidad de utilizar una estrategia determinista para refinar resultados. A raíz de esta idea se propone el uso del algoritmo Cold Simulated Annealing (CSA).

Algorithm 1 Simulated Annealing algorithm.

Require: Initial temperature, final temperature, cooling constant

Set initial position (y)
Set max number of accepted solutions ($MaxSol$)
Set max number of rejected solutions ($MaxRej$)
Set max number of executions ($MaxEx$)
Compute initial solutions matrix
Select best solution out of initial solutions matrix as best global solution
while error (e_i is larger than Tolerance) AND (rejects is smaller than max number of rejects) AND (temperature is smaller than final temperature) **do**
    Compute new position x
    Evaluate new solutions matrix
    Compute difference between new solution and best global solution
    if negative difference is larger than tolerance **then**
        Update best global solution with new solution
        Update position vector y
        Count accepted solution +1
    else if metropolis criteria is fufilled **then**
        Update best global solution with new solution
        Update position vector y
        Count accepted solution +1
    else
        Count Rejected solution +1
    if (Max execution is less than Execution Number) OR (Max accepted solutions is less than number of accepted solutions) **then**
        Reduce temperature
        Restart execution and solution count
    Compute error (($e_{i+1} = bestsolution - desiredsolution$)

La idea detrás CSA es el mismo algoritmo que el SA utilizado normalmente, su principal diferencia radica en su temperatura inicial. En el caso de esta investigación se utilizará el CSA con una temperatura 100 veces menor a la de SA con el fin de poder desarrollar etapas de refinamiento con comportamiento determinista.

Particle Swarm Optimization (PSO): Particle Swarm Optimization (PSO) o método de optimización de enjambre de partículas se define para funciones continuas no lineales. Es un algoritmo creado por Kennedy y Eberhart en el año de 1995, este método establece sus bases sobre modelos sociales de interacción como la cría de ciertas aves, escolarización de algunas especies de peces y la teoría de enjambres en general. Para la implementación, este algoritmo explora el espacio de la función objetivo realizando una métrica para ajustar las trayectorias de las partículas bajo dos componentes principales, una componente determinista donde cada partícula es atraída hacia la mejor posición por iteración *Lbest* consecuentemente hacia la mejor solución global *Gbest* y la componente estocástica que sucede al mismo tiempo con tendencia a moverse aleatoriamente. La posición de cada partícula tras una iteración viene descrita por la siguiente ecuación

$$V_i = \theta V_i + \alpha\varepsilon_1(Lbest - x_i) + \beta\varepsilon_2(Gbest - x_i) \tag{2.12}$$

$$x_i = x_i + V_i \tag{2.13}$$

donde x_i y v_i son la posición y la velocidad, ε_1 y ε_2 son valores aleatorios de entre cero y uno, θ es la constante de inercia, α es la constante de escalamiento de la mejor solución local y β es la constante asociada con la mejor solución global.

Cuando una partícula encuentra una posición mejor a *Lbest* automáticamente se actualiza como la mejor posición por iteración y se contrasta con la mejor solución global hasta el momento *Gbest* hasta que el objetivo no mejore o el criterio de parada le indique al algoritmo que se debe detener. A nivel computacional se implementa en pocas líneas de código buscando siempre la mejor posición histórica de la función objetivo tal como se puede observar en el siguiente pseudocódigo [21][24].

Busqueda de Cuckoo (Cuckoo Search): La búsqueda del cuckoo (CS) es uno de los mas recientes algoritmos metaheurísticos inspirados en la naturaleza, desarrollado en 2009 por Xin-She Yang de la Universidad de Cambridge y Suash Deb de C.V. Facultad de Ingeniería Raman. CS se basa en el parasitismo de cría de algunas especies de cuckoos. Además, este algoritmo se ve reforzado por los llamados vuelos de Lévy en lugar de simples caminatas aleatorias isotrópicas. Los cuckoos son una especie de aves que acostumbra a poner sus huevos en nidos de otras especies de aves en vez de nidos propios. Partiendo de este hecho,

Algorithm 2 PSO algorithm.

Require: Acceleration constants, Inertia, Tolerance

1: Set initial velocity (v_0)
2: Set initial solutions matrix
3: **if** Solution is out of bound **then**
4: Compute 2π module
5: Evaluate initial solutions matrix
6: Select best solution
7: **while** error (e_i is larger than Tolerance) AND (Stuck Solution count is smaller than max Stuck Solution) AND (iteration count is smaller than Max iteration) **do**
8: Compute new velocity $v_{i+1} = inercia \times vi + alpha \times rand \times (bestposition - positionvector) + beta \times rand. \times (Globalbest - positionvector)$
9: Compute new positions
10: Compute new solutions matrix
11: **if** Solution is out of bound **then**
12: Compute 2π module
13: Evaluate new solutions matrix
14: Compute error ($(e_{i+1} = bestsolution - desiredsolution)$

se presenta el siguiente modelo el cual sigue 3 reglas principales.

1. Cada cuckoo pone un huevo a la vez y lo deja en un nido elegido al azar [21][24].

2. Los mejores nidos con huevos de alta calidad se transferirán a las próximas generaciones [21][24].

3. El número de nidos disponibles para hospedar es huevos es fijo, y el ave hospedante descubre el huevo puesto por un cuckoo con una probabilidad (pa) entre 0 y 1. En este caso, el ave huésped puede deshacerse del huevo o simplemente abandonar el nido y construir un nido completamente nuevo[21][24].

Como una aproximación adicional, esta última suposición se puede aproximar por un reemplazo de una fracción pa de los n nidos hospederos con nidos nuevos (con nuevas soluciones aleatorias). Para un problema de maximización, la calidad o idoneidad de una solución puede ser simplemente proporcional al valor de la función objetivo. Desde el punto de vista de la implementación, podemos usar las siguientes representaciones simples de que cada huevo en un nido representa una solución, y cada cuckoo puede poner solo un huevo (lo que representa una solución). El objetivo es utilizar las nuevas y potencialmente mejores soluciones

(cuckoos) para reemplazar una solución no tan buena en los nidos. Aquí usamos el enfoque más simple, donde cada nido tiene un solo huevo . Las soluciones nuevas posiciones de los cuckoos vienen dadas por

$$x_i^{t+1} = x_i^t + \alpha L(s, \lambda) \tag{2.14}$$

donde $L(s, \lambda)$ representa los vuelos de Levy

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi s^{1+\lambda}} \tag{2.15}$$

Aquí α >0 es el factor de escala del tamaño del paso, que debe estar relacionado con las escalas del problema de interés.[21] En esencia, $L(\alpha)$ es un parámetro de tamaño de paso, más específicamente el tamaño de paso basado en vuelos de Lévy y $\Gamma(\alpha)$ es la función gamma estándar, y esta distribución es válida para pasos grandes s>0. [21]

El valor es $\beta = 3/2$ y la función gamma es:

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \tag{2.16}$$

La mejor forma de implementar de forma eficiente el vuelo de Lévy en el código del algoritmo del Cuco es a través del algoritmo de Mantegma (véase ecuación 2.17); este determina la distribución simétrica estable de Lévy, es decir que los pasos de la caminata aleatoria pueden ser positivos y negativos [21][24]. Entonces, la longitud del paso se define como:

$$s = \frac{u}{|v|^{1/\beta}} \tag{2.17}$$

Donde u, v provienen de distribuciones normales tales que $u \sim \mathcal{N}(0, \sigma_u^2)$ y $v \sim \mathcal{N}(0, \sigma_v^2)$ (véase ecuaciones 2.18 y 2.19), las varianzas se definen como:

$$\sigma_u^2 = \left\{ \frac{\Gamma(1+\beta) \sin(\pi\beta/2)}{\beta \Gamma\left[(1+\beta)/2\right] 2^{(\beta-1)/2}} \right\}^{1/\beta} \tag{2.18}$$

$$\sigma_v^2 = 1 \tag{2.19}$$

Por todo lo anterior, la ventaja adicional que tiene este algoritmo es que su búsqueda global utiliza los vuelos de Lévy en sus caminatas aleatorias, esto lleva al algoritmo a explorar la superficie de búsqueda de forma más eficiente que aquellos algoritmos que utilizan la caminata aleatoria estándar con distribución gaussiana. Esta ventaja combinada con una búsqueda local y convergencia global garantizada, hacen del Cuco un algoritmo muy eficiente

El Algoritmo (CUCKOO) tiene dos características importantes, relacionadas con las caminatas aleatorias una local y la otra global, para la local se define como:

$$x_i^{t+1} = x_i^t + \alpha\varepsilon \odot H(p_a - \varepsilon) \odot (g^* - x_k^t) \tag{2.20}$$

Donde g^* es la mejor solución por iteración y x_k^t es una solución seleccionada de forma aleatoria a través de permutación aleatoria, α el factor de escalamiento, H la función escalón de Heaviside que analiza la diferencia entre la probabilidad de abandono de un nido con un huevo cuco y valor aleatorio ε uniformemente distribuido. $\odot$ es el producto de Hadamard $u \odot v = u_{ij}v_{ij}$ [21][24].La global es la misma ecuación 2.14.

Este algoritmo utiliza una combinación equilibrada de un recorrido aleatorio local y un recorrido aleatorio exploratorio global, controlado por un parámetro de conmutación *pa*. Con esto el pseudocódigo vendrá dado por:

Algoritmo del Murciégalo (Bat Algorithm): En el 2010 el profesor Xin-She Yang de la Universidad de Cambridge, planteó el algoritmo del Murciélago, su inspiración biológica se basa en el comportamiento de eco-localización de los murciélagos. Los murciélagos son los únicos mamíferos con alas y con una gran capacidad de eco-localización, la cual sirve para detectar presas, evitar obstáculos y encontrar sus grietas en la oscuridad. Los murciélagos son capaces de emitir ondas sonoras de hasta 110 dB, los cuales rebotan en los objetos o cuerpos cercanos y luego son detectado por estos nuevamente. Estas ondas tienen diferentes características, dependiendo del tipo de murciélago los armónicos, ancho de banda y frecuencia de las ondas pueden variar. La mayoría de los murciélagos utilizan señales cortas moduladas en frecuencia. El rango de frecuencias típicas de los murciélagos se encuentra entre los 25KHz y los 150KHz, con ráfagas de pulsos que duran entre 5 y 20 ms. Se estima que hay más de 1000 especies, en una gama amplia de pesos y tamaños, y todos estos murciélagos utilizan la eco-localización.

El algoritmo está basado en el comportamiento biológico de los murciélagos. Para esto se define una población de murciélagos n los cuales utilizarán la eco-localización y volarán de manera aleatoria con velocidad y posiciones diferentes para detectar por ende para este algoritmo asumiremos que

1. Todos los murciélagos usan la eco-localización para detectar la distancia, y también son capaces de diferenciar entre comida / presa y barreras de fondo [21][24].

2. Los murciélagos vuelan aleatoriamente con velocidad vi en la posición x_i. Pueden ajustar automáticamente la frecuencia (o longitud de onda) de sus pulsos emitidos y ajustar

Algorithm 3 Cuckoo Algorithm.

Require: Nest Abandonment Probability pa ,Distribution index β

Set initial solutions matrix
if Solution is out of bound **then**
    Compute 2π module
Evaluate initial solutions matrix
while error (e_i is larger than Tolerance) AND (Stuck Solution count is smaller than max Stuck Solution) AND (iteration count is smaller than Max iteration) **do**
    Generate New Cuckoos using Levy Flight
    Evaluate new solution matrix
    Select cuckoos randomly from the new cuckoo group and insert them new into the original population if better
    if New Solution is out of bound **then**
        Compute 2π module
    Evaluate new solution matrix
    Sort Solutions
    Generate random cuckoos and Replace the worst solution with these new cuckoos
    Generate new solutions $x_i^{t+1} = x_i^t + \alpha s \odot H(p_a - \varepsilon) \odot (g^* - x_k^t)$ Ec. (2.20)
    if New Solution is out of bound **then**
        Compute 2π module
    Evaluate new solution matrix
    Sort Solutions
    Select best Solution
    Evaluate if stop criteria is met

la tasa de emisión de pulsos r entre 0 y 1, dependiendo de la proximidad de su objetivo [21][24].

3. Aunque la sonoridad de los pulsos puede variar de muchas formas, asumimos que la sonoridad varía desde un A0 grande (positivo) hasta un valor mínimo Amin[21][24].

El algoritmo del murciélago comparte ciertas características con el algoritmo de optimización de partículas (PSO). Para empezar, ambos de estos son algoritmos basados en poblaciones donde las poblaciones tienen comportamientos similares a los de un enjambre. En estos algoritmos los individuos se desplazan de tal manera que la trayectoria de cada una de las partículas del enjambre se ve determinada por componentes deterministas que la acercan hacia las óptimos locales y globales y componentes aleatorios que añaden cierto grado de aleatoriedad a las nuevas posiciones propuestas. La principal diferencia radica en que en el caso de BAT, a diferencia de PSO, no solo se toma en cuenta la velocidad y posición de cada partícula sino se toma en cuenta la frecuencia y tasa de pulso con que se emite el sonido para actualizar. La posición del murciélago vendrá dada tal como se ilustra en las siguientes ecuaciones:

$$V_i = \theta V_i + \beta\varepsilon(Gbest - x_i)fi \tag{2.21}$$

$$r = r_0(1 - e^{-\gamma k}) \tag{2.22}$$

$$A = \alpha A_0 \tag{2.23}$$

donde θ es la inercia, ε es un numero aleatorio, β es la constante de escalamiento y fi la frecuencia a la que el murciélago emite los pulsos. En el caso de la ecuación 2.22 y 2.23 tenemos la tasa de emisión de pulsos Y el volumen con el que se emiten los sonidos se ven ajustados en cada iteración por las constantes de variación de tasa de pulso γ y la constante de variación del volumen α donde A_0 y r_0 son el volumen y tasa de emisión de pulsos iniciales.

Al tomar en cuenta estas consideraciones adicionales, el algoritmo de murciélago utiliza los parámetros de emisión de tasa de pulso, volumen y frecuencia emitida para actualizar su posición y así buscar el nuevo óptimo a diferencia de PSO que únicamente depende de sus componentes deterministas y aleatorias.

Algoritmo Genetico (GA): El algoritmo genético, es un método de optimización basado en el principio de selección genética, creado por John Holland y sus colaboradores en 1970. Su inspiración biológica parte de la teoría de selección natural de Charles Darwin y la generación de cromosomas. En la naturaleza las especies que son capaces de adaptarse a entornos cambiantes son las que sobreviven. Estas características son generales de cada

Algorithm 4 BAT Algorithm.

Require: Volume, Frequency, Velocity, Acceleration, Inertia, Tolerance

Set initial pulserate (r)
Set initial position vector $x1$
Evaluate initial solutions matrix
while error (e_i is larger than Tolerance) AND (Stuck Solution count is smaller than max Stuck Solution) AND (iteration count is smaller than Max iteration) **do**
    Compute new frequency $fi_{i+1} = fmin + (fmax - fmin) \times rand$
    Compute new velocity $v_{i+1} = inercia \times vi + aceleration \times rand + (bestposition - initialposition) \times fi$
    Compute new position $x_2 = x_1 + v_{i+1}$
    if Position vector x1 is out of bounds **then**
        Compute 2π module
    Compute new solutions matrix with new position x2
    if Random number is smaller the pulse rate r **then**
        Update Position $x1$
    if Position vector x1 is out of bounds **then**
        Compute 2π module
    Compute new solutions matrix with updated position $x1$
    if (Loudness is smaller than random number) AND (random number is less than best solution) **then**
        Increase pulse rate r
        Decrease Volume
    Compare solutions matrix produced by $x1$ and $x2$
    Select best solution
    Compute error ($e_{i+1} = bestsolution - desiredsolution$)

especie, pero existen ciertas características únicas relacionadas con cada individuo y están determinadas con su contenido genético. Cada característica del individuo está controlada por una unidad básica llamada gen. Los conjuntos de características de control de genes forman los cromosomas, que son las claves para la supervivencia del individuo en un entorno competitivo. [21][24]

GA es un algoritmo de búsqueda global que puede asumir un amplio rango de problemas de optimización, lineales o no lineales,funciones estacionarias o no estacionarias, continuas o discontinuas. En este algoritmo cada cromosoma o conjuntos de cromosomas son representados por cadenas de bits en el caso binario o valores dentro del dominio de la función. En el caso de esta investigación, se trabajará con un algoritmo genético real ya que las soluciones tanto en las funciones de prueba como en los experimentos que llevaremos a cabo, están definidas en un dominio de números reales. El algoritmo genético se divide en cuatro operaciones principales, la creación de la población, la selección natural, cruce y mutación. La primera operación consiste simplemente en crear una población de n cromosomas de d dimensiones. En la selección natural cada cromosoma tiene información relacionada con la solución que personifica. En el cruce se considera una parte del cromosoma (alelos) de los padres dando paso a nuevos cromosomas, donde los débiles son descartados para remover las soluciones malas y en la mutación se busca que los agentes no se estanquen en óptimos locales, para lo cual se realizan cambios aleatorios cruzados en los alelos del cromosoma. Los cromosomas actúan de forma independiente y están asociados con una aptitud física que refleja cuán buena es. Los cromosomas pueden explorar el área de búsqueda de manera simultánea, lo que indica que entre mayor sea el valor de aptitud de un cromosoma, mayores serán sus posibilidades de supervivencia y reproducción y mayor será su representación en la siguiente iteración (generación) posterior [21] [35].

2.0.2. Funciones de Optimización de Referencia

Las funciones de Optimización de referencia son funciones utilizadas para validar y comparar el desempeño de los algoritmos. Dichas funciones nos permiten representar ciertos escenarios o casos artificiales en los que se puede observar el comportamiento del algoritmo con el fin de observar posibles errores y afirmar que su funcionamiento es el adecuado. Para esto consideramos las siguientes funciones de optimización de referencia de $X = (x_1, x_2) \in \mathbb{R}^2$, se las agrupa según sus similitudes en sus propiedades y formas, como son modalidad, separabilidad, valles, presencia de cuencas, etc. Buscando que las funciones de prueba utilizadas tengan propiedades diversas e imparciales, consideraremos tres tipos de superficies, aquellas que presenta un óptimo local y varios óptimos globales (Ackley y modified Schaffer #2),

Algorithm 5 Genetic Algorithm.

Require: Crossover Posibility, Mutation Posibility, Mutation Step

Set initial solutions matrix
if Solution is out of bound **then**
    Compute 2π module
Evaluate initial solutions matrix
while error (e_i is larger than Tolerance) AND (Stuck Solution count is smaller than max Stuck Solution) AND (iteration count is smaller than Max iteration) **do**
    Generate Random vector
    Crossover using Totem Pole Strategy
    if New Solution is out of bound **then**
        Compute 2π module
    Generate Random Vector
    Evaluate which elements of the Random Vector are smaller than the mutation probability
    Mutate the chromosomes in these positions generating a new solution
    if New Solution is out of bound **then**
        Compute 2π module
Sort Solutions
Select best Solution
Evaluate if stop criteria is met

aquellas que sus superficies son planas con óptimo global (Goldstein-Price y León) y aquellas que tienen varios óptimos globales (Bird, Holder table).

2.0.2.1. Óptimo Global Único

1. **Función Ackley 1:** Esta función continua, diferenciable, no separable, escalable y multimodal. Presenta una región externa casi plana con muchos óptimos locales y un gran agujero en el centro tal como se observa en la figura 2.6. Se selecciona por el riesgo que tienen los algoritmos de optimización(particularmente algoritmos escaladores) de ser atrapados en mínimos locales [5].

$$f(x_1, x_2) = 20\left(1 - e^{-0{,}2\sqrt{\frac{x_1^2+x_2^2}{2}}}\right) - e^{\frac{\cos(2\pi x_1)+\cos(2\pi x_2)}{2}} + e$$

Sujeta $-35 \leq x_i \leq 35$, el mínimo global se encuentra en $X^* = (0, 0)$, $f(X^*) = 0$.

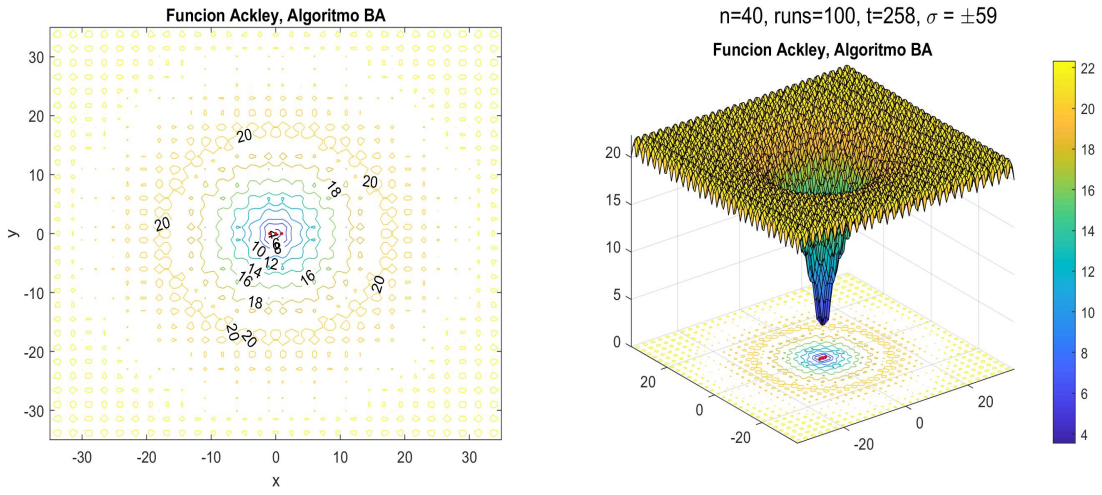


Figura 2.6: a) Funcion Ackley 2D. b) Funcion Ackley 3D [5]

2. **Función Modified Schaffer #2:** Esta función continua, diferenciable, no separable, escalable y multimodal. Es basada en la familia de las funciones Schaffer, presenta una pequeña área que contiene el óptimo global y a su alrededor muchos óptimos locales tal como se observa en la figura 2.7. Se selecciona por el riesgo que tienen los algoritmos de ser atrapados en mínimos locales y la necesidad explorar forma efectiva la superficie de búsqueda [5].

$$f(x_1, x_2) = 0{,}5 + \frac{\sin^2(x_1^2 - x_2^2) - 0{,}5}{(1 + 0{,}001(x_1^2 + x_2^2))^2}$$

Sujeta a $-100 \leq x_i \leq 100$, el mínimo global se encuentra en $X^* = (0, 0)$, $f(X^*) = 0$.

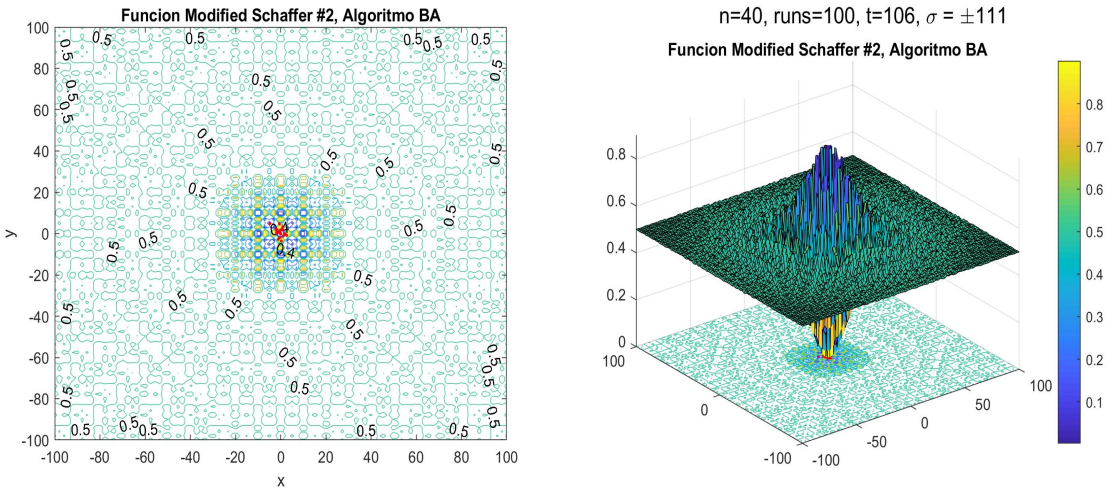


Figura 2.7: a) Función Modified Schaffer 2D. b) Función Modified Schaffer 3D [5]

.

3. **Función Egg Crate #7:** Esta función continua, diferenciable, separable, no escalable. Es basada en la forma de un contenedor en el cual normalmente los huevos son envasados tal como se observa en la figura 2.8. Presenta a lo largo de su área diversos máximos y mínimos que emulan la forma de dicho contenedor. En su centro se encuentra el mínimo global. Este algoritmo se selecciona ya que presenta diversos mínimos locales los cuales representan un reto para el algoritmo dado que este puede quedar atrapado ahí. [5].

$$f(x_1, x_2) = x_1^2 + x_2^2 + 25(\sin^2(x_1) + \sin^2(x_2))$$

Sujeta a $-5 \leq x_i \leq 5$, el mínimo global se encuentra en $X^* = (0, 0)$, $f(X^*) = 0$.

2.0.2.2. Superficies Planas

1. **Función Goldstein-Price:** Esta función continua, multimodal y no separable. Tiene forma de hoja doblada que genera un gran valle con muchos óptimos locales, con zonas

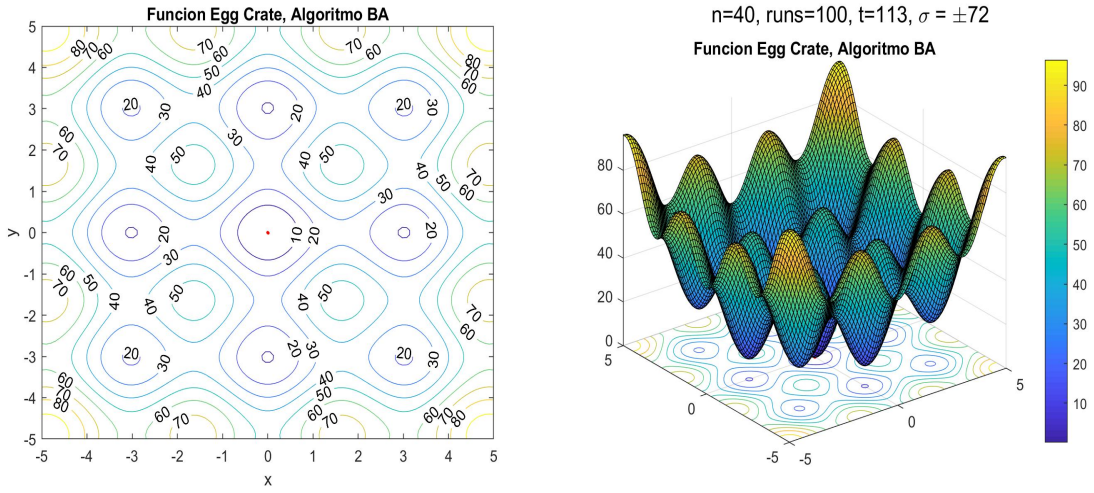


Figura 2.8: a) Función Egg Crate 2D. b) Función Egg Crate 3D. [5]

que presentan un fuerte descenso tal como se ilustra en la figura 2.9. Se selecciona porque el proceso de exploración puede ralentizarse considerablemente conforme el algoritmo encuentra el valle, además funciones con superficies planas presentan mayor dificultad para los algoritmos [5].

$$f(x_1, x_2) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \\ \left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$$

Sujeta a $-2 \leq x_i \leq 2$, el mínimo global se encuentra en $X^* = (0, -1)$, $f(X^*) = 3$.

2. **Función León:** Esta función continua, diferenciable, no separable, no escalable y unimodal. Tiene forma de hoja doblada que genera un gran valle con único óptimo global, con zonas que presentan un fuerte descenso según se ilustra en la figura 2.10. Se selecciona porque al igual que la función Goldstein-Price, el proceso de exploración puede ralentizarse considerablemente conforme el algoritmo encuentra el valle, además funciones con superficies planas presentan mayor dificultad para los algoritmos [5].

$$f(x_1, x_2) = 100(x_2 - x_1^3)^2 + (1 - x_1)^2$$

Sujeta a $-1{,}2 \leq x_i \leq 1{,}2$, el mínimo global se encuentra en $X^* = (1, 1)$, $f(X^*) = 0$.

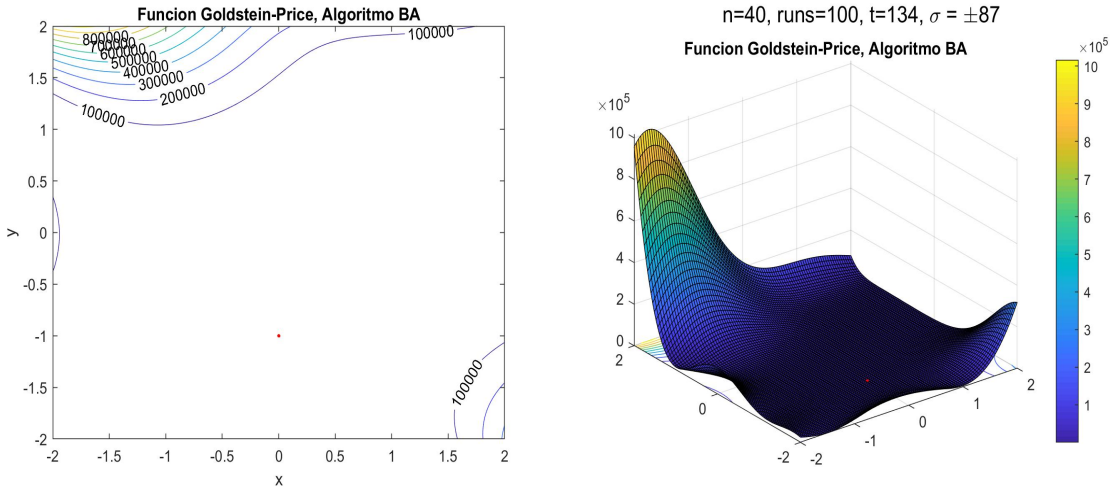


Figura 2.9: a) Función Goldstein Price b) 2D Función Goldstein Price 3D[5]

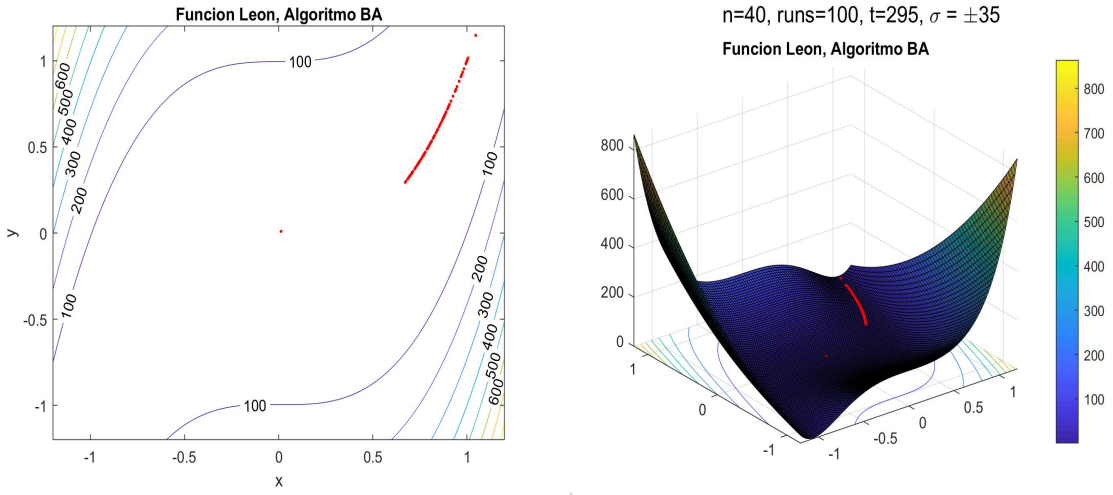


Figura 2.10: a) Función Leon 2D. b) Función Leon 3D[5].

2.0.2.3. Múltiples Óptimos Globales

1. **Función Bird:** Esta función continua, diferenciable, no separable, no escalable y multimodal. Presenta dos mínimos globales y dos locales como se puede observar en la figura 2.11, y mantiene un parecido a un panal de huevos. Se selecciona porque el algoritmo se puede estancar en óptimos locales y por su característica multimodal [5].

$$f(x_1, x_2) = \sin(x_1)e^{(1-\cos(x_2))^2} + \cos(x_2)e^{(1-\sin(x_1))^2} + (x_1 - x_2)^2$$

Sujeta a $-2\pi \leq x_i \leq 2\pi]$, sus dos mínimos globales se encuentran en $X^* = (4{,}70105, 3{,}15294)$, $(-1{,}58214, -3{,}13024)$, $f(X^*) = -106{,}76453$

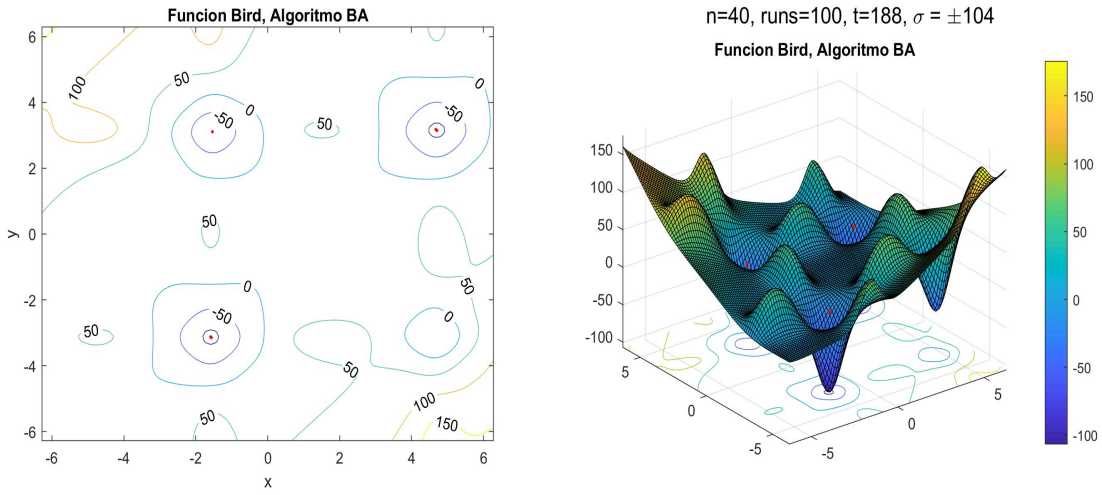


Figura 2.11: a) Función Bird 2D b) Función Bird 3D[5]

.

2. **Función Holder table 2:** Esta función continua, diferenciable, separable, no escalable y multimodal. presenta 4 óptimos globales simétricos en los extremos de la superficie formando una mesa, con múltiples óptimos locales como se puede notar en la figura 2.12. Se selecciona porque el algoritmo se puede estancarse en óptimos locales y por su característica multimodal [5].

$$f(x_1, x_2) = -\left|\sin(x_1)\cos(x_2)e^{\left|1-\sqrt{\frac{x_1^2+x_2^2}{\pi}}\right|}\right|$$

Sujeta a $-10 \leq x_i \leq 10$, el mínimo global se encuentra en $X^* = (\pm 8{,}05502, \pm 9{,}66459)$, $f(X^*) = -19{,}20850$

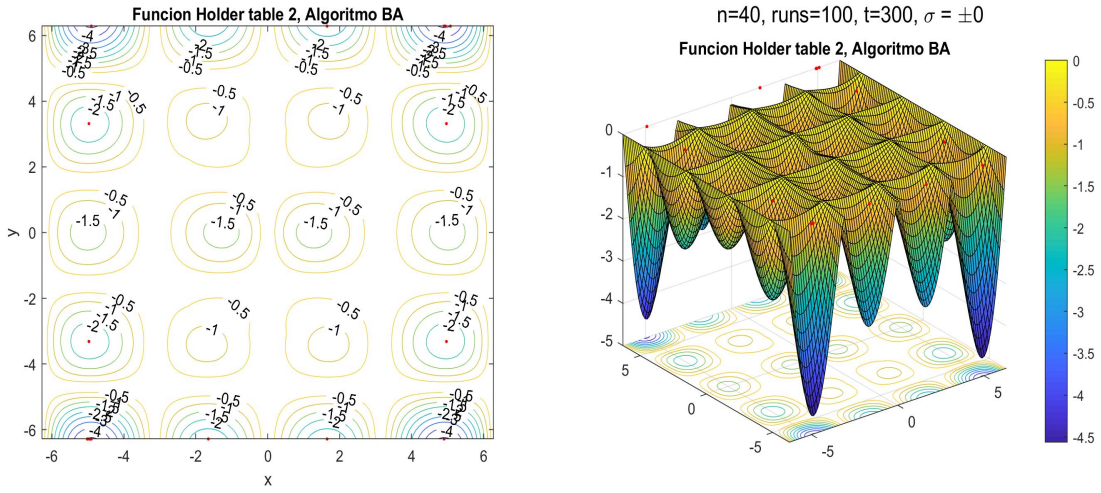


Figura 2.12: a) Función Table 2D b)Función Table 3D[5]

2.0.3. Algoritmos Metaheurísticos Híbridos

Tras haber repasado los algoritmos metaheurísticos que se tendrán en cuenta, podemos ver como pueden ser aplicados en diferentes problemas de optimización, sin embargo, se debe considerar que pueden existir problemas de optimización cuyos requerimientos no se puedan satisfacer con estos algoritmos. En diversas investigaciones se ha encontrado que la idea de combinar diferentes algoritmos, o las mejores cualidades de ellos, resulta en una optimización más eficiente y capaz de producir mejores resultados que el uso de cada algoritmo por separado [36] [37]. Partiendo de esta idea surge el concepto de algoritmos metaheurísticos híbridos. Los algoritmos metaheurísticos híbridos se pueden definir como: aquellos algoritmos metaheurísticos que combinan diferentes conceptos o componentes de mas de un algoritmo metaheurístico[38]. Este proyecto se centrará en la aplicación de algoritmos metaheurísticos híbridos desarrollados a partir del trabajo de investigación desarrollado por Polanco (2021) [24] sobre el beamforming de un sistema de antenas MIMO Masivo 8x8[21].

Cada día el número de algoritmos metaheurísticos presentes en la literatura aumenta y por ende las posibilidades de hibridación también. En el 2014 Xin-She Yang propuso que la cantidad de algoritmos metaheurísticos podría ser determinada por una simple ecuación combinatoria donde si se tiene un número de algoritmos N y $2 < k < N$ el número de combinaciones posibles de algoritmos vendrá dado por:

$$C_k^N = \frac{n!}{k!(n-k!)} \tag{2.24}$$

Esto quiere decir que si consideramos todos los valores posibles que K puede tomar en ese

rango tendremos que el número de algoritmos diferentes vendrá dado como si fuese una suma de coeficientes binomiales.

$$\sum_{k=0}^{N} \binom{n}{k} = 2^N \tag{2.25}$$

En otras palabras si se consideran unos 20 algoritmos podríamos tener una combinación de hasta 1,048,576 algoritmos híbridos diferentes. Sin embargo, más allá de que exista esta posibilidad de generar tantos algoritmos diferentes, no quiere decir que todos sean adecuados. Yang también propone que la combinación de algoritmos no debe realizarse de manera aleatoria. La combinación de algoritmos debe ser realizada tomando en cuenta las mejores cualidades de cada uno de los algoritmos y como se pueden complementar entre si para resolver el problema deseado [6].

La mayoría de los algoritmos sufren de dos problemas, convergencia prematura y convergencia tardía. En el caso de la convergencia prematura se obtiene una convergencia rápida tomando en cuenta el hecho de que el resultado puede estar lejos del valor deseado. Por otra parte, si un algoritmo tiene una convergencia tardía, generalmente su precisión será mejor y sus resultados estarán mas cercanos a los deseados pero su tiempo de convergencia será mayor. Adicionalmente también existe una relación entre la diversidad de las soluciones y el tiempo de convergencia que se puede observar en la figura 2.13. Normalmente la diversidad viene dada por la cantidad y calidad de las soluciones. A medida que un algoritmo se ejecuta, la diversidad de las soluciones ira disminuyendo ya que todas se irán aproximando hacia un mismo punto. Al utilizar algoritmos con mayor diversidad es posible encontrar mejores soluciones, sin embargo esto aumenta el tiempo de convergencia. Por otra parte, sucede el efecto inverso, entre menor sea la diversidad de una solución, mas rápido convergerá el algoritmo pero sus resultados no serán tan buenos como los del escenario anterior. Teniendo esto en cuenta, se presenta la idea de que para los algoritmo metaheuristicos híbridos se debe encontrar un punto de equilibrio que satisfaga la tasa de convergencia adecuada y la aproximación a resultados deseables en base a la diversidad de población utilizada [6].

2.0.3.1. Taxonomía o Clasificación de los algoritmos metaheurísticos

Como hemos podido observar, no es tarea fácil desarrollar un nuevo algoritmo híbrido. Teniendo en cuenta el gran número de combinaciones posibles, elegir una combinación adecuada no es sencillo y es aquí donde reside uno de los mayores retos de esta investigación. A raíz de esto es importante tomar en cuenta la taxonomía, normas de clasificación y criterios de combinación de los algoritmos metaheurísticos híbridos.

Al momento de hablar de taxonomía se hace referencia a la ciencia que se encarga de

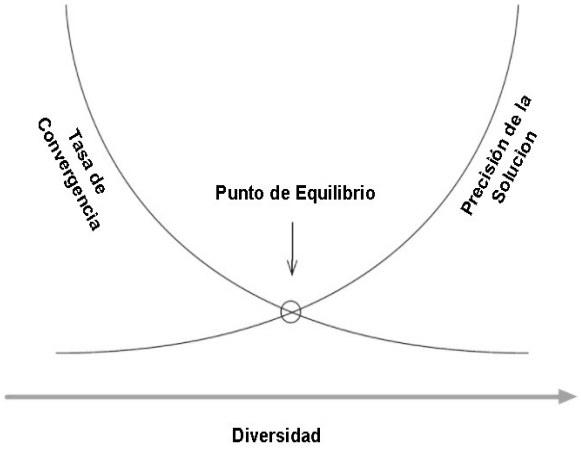


Figura 2.13: Punto de equilibrio entre convergencia y precisión [6]

organizar de manera jerarquizada y sistemática a cualquier conjunto de objetos o individuos. En el mundo de los algoritmos metaheurísticos se manejan dos tipos de clasificación o taxonomía en la literatura.

Taxonomía según Talbi La taxonomía desarrollada por Talbi que se observa en la figura 2.14 es una de las más utilizadas en la literatura. Esta taxonomía se compone de dos grupos principales: algoritmos organizados de forma jerárquica y algoritmos organizados de forma plana.

Dentro de la clasificación jerárquica tenemos las hibridaciones de alto y bajo nivel. Las hibridaciones de bajo nivel hacen referencia a métodos de optimización en los cuales una función especifica de algún algoritmo puede ser reemplazada por la de otro. En los algoritmos híbridos de alto nivel los algoritmos son cerrados y no se toma en cuenta su funcionamiento interno, esto quiere decir que únicamente se centran en las entradas y salidas de cada algoritmo.

Ambas de estos subgrupos se pueden clasificar en algoritmos de relevo o de trabajo en equipo. En la hibridación de relevo los algoritmos son ejecutados de manera secuencial, es decir se ejecutan uno tras otro donde la salida del primero es la entrada del segundo. Por otra parte la hibridación de trabajo en equipo representa modelos en los cuales varios agentes o algoritmos trabajan de manera paralela.

Los algoritmos de relevo de bajo nivel representan la clase de algoritmos que están em-

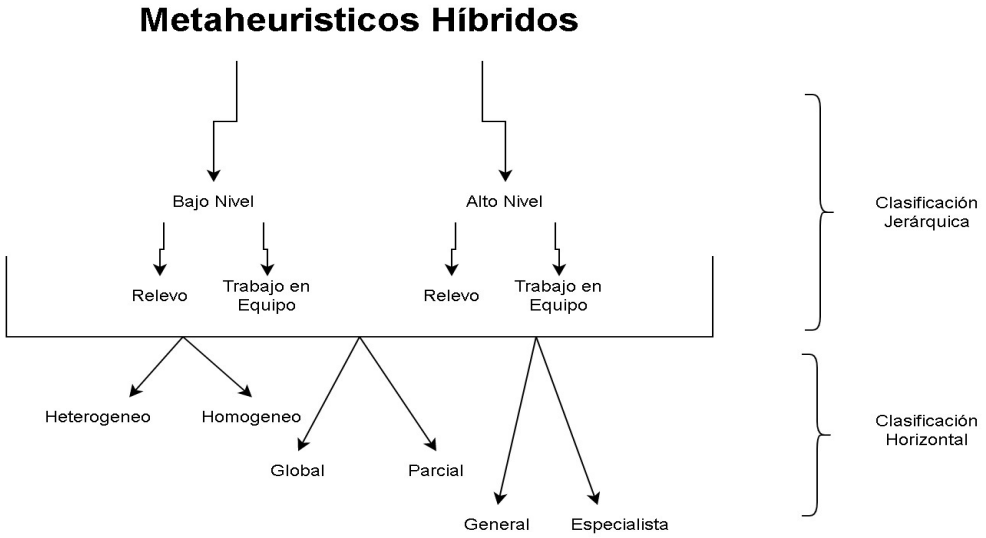


Figura 2.14: Taxonomía según Talbi [7]

bebidos dentro de otro. Los algoritmos de trabajo en equipo de bajo nivel toman en cuenta la explotación y exploración de óptimos como su principal meta o objetivo y por ende se utilizan dos algoritmos en paralelo. Generalmente se utiliza un algoritmo basado en población ya que son buenos para la exploración global y un algoritmo de búsqueda local considerando las ventajas que tienen estos para encontrar óptimos locales de manera mas precisa.

En cuanto a los algoritmos de alto nivel tenemos: Los algoritmos de relevo de alto nivel y Los algoritmos de trabajo en equipo de alto nivel. Los algoritmos de relevo de alto nivel son un conjunto de algoritmos contenidos en si mismos ejecutados de manera secuencial mientras que los algoritmos de trabajo de equipo de alto nivel son varios algoritmos contenidos en si mismos ejecutados de forma paralela.

En cuanto a la clasificación plana tenemos 3 subconjuntos, el primero de estos está compuesto por los algoritmos homogéneos y heterogéneos. Los algoritmos híbridos homogéneos son todos los algoritmos combinados que están inspirados en el mismo fenómeno de la naturaleza mientras que los heterogéneos hacen referencia a algoritmos conformados por diferentes tipos de algoritmos. El segundo conjunto se encuentra compuesto por los algoritmos parciales y los algoritmos globales. Los híbridos globales se encargan de resolver un único

problema de manera global, mientras que los híbridos parciales descomponen un problema en problemas mas pequeños. Finalmente tenemos los algoritmos generales y especialistas. Los Algoritmos generales son aquellos que buscan resolver el mismo problema de optimización mientras que los algoritmos especialistas combinan algoritmos que resuelven diferentes tipos de problemas.[7]

Taxonomía según Raidl En este articulo se encarga de presentar una mirada más amplia hacia lo que pueden llegar a ser los algoritmos metaheurísticos híbridos. A diferencia de la taxonomía de Talbi que se encarga únicamente de considerar las características y estrategias de hibridizacion, Raidl presenta una clasificación mas completa que abarca ideas no solo de como se pueden combinar diferentes algoritmos sino también de como estas ideas se pueden enfocar más hacia la implementación de dichos algoritmos. Dicha taxonomía se ilustra en la figura 2.15

Esta taxonomía se encuentra dividida en 4 grupos principales donde el primero se encarga de clasificar a las estrategias de hibridación según qué se está combinando. Esto se refiere a si la hibridación se hace entre dos algoritmos metaheurísticos, un algoritmo metaheurístico con algún algoritmo en especifico, o un algoritmo metaheurístico con un algoritmo de inteligencia artificial u otro recurso operativo. El segundo grupo utilizado en esta clasificación se asimila a la clasificación utilizada en la taxonomía de Talbi, donde se considera si los algoritmos son de alto o bajo nivel. En este caso, los algoritmos se clasifican según su nivel de hibridizacion, con lo cual se refiere a si dichos algoritmos se hibridizan considerándose como algoritmos contenidos en si mismos o simplemente se combinan algunas partes de dichos algoritmos. El tercer grupo de clasificación se denomina clasificación según orden de ejecución. Hace referencia a la forma en que se ejecutan los algoritmos siendo estos de manera paralela, secuencial o una combinación de ambos métodos. Por ultimo, tenemos la hibridizacion según estrategia de control, la cual se puede dividir en dos grandes categorías: algoritmos colaborativos e integradores.

Los estrategias colaborativas son aquellas que se caracterizan por ser la combinación de dos o más algoritmos corriendo en manera secuencial o en paralelo. En los estrategias híbridas colaborativas multietapa el primer algoritmo actúa como un optimizador global y el segundo actúa como un buscador local. En las colaborativas Secuenciales ambos algoritmos se ejecutan de manera secuencial hasta que alguno de los dos encuentre un criterio de convergencia. Por ultimo, en las estrategias paralelas de este tipo se ejecutan dos algoritmos al mismo tiempo sobre una misma población. [6] [39]

En las estrategias integradoras un algoritmo se establece como el maestro y otro como el subordinado. En este caso el algoritmo secundario tiene un peso del 10-20 por ciento dentro del algoritmo principal. Un clásico ejemplo de esto es la configuración de isla.[39] [6] [39]

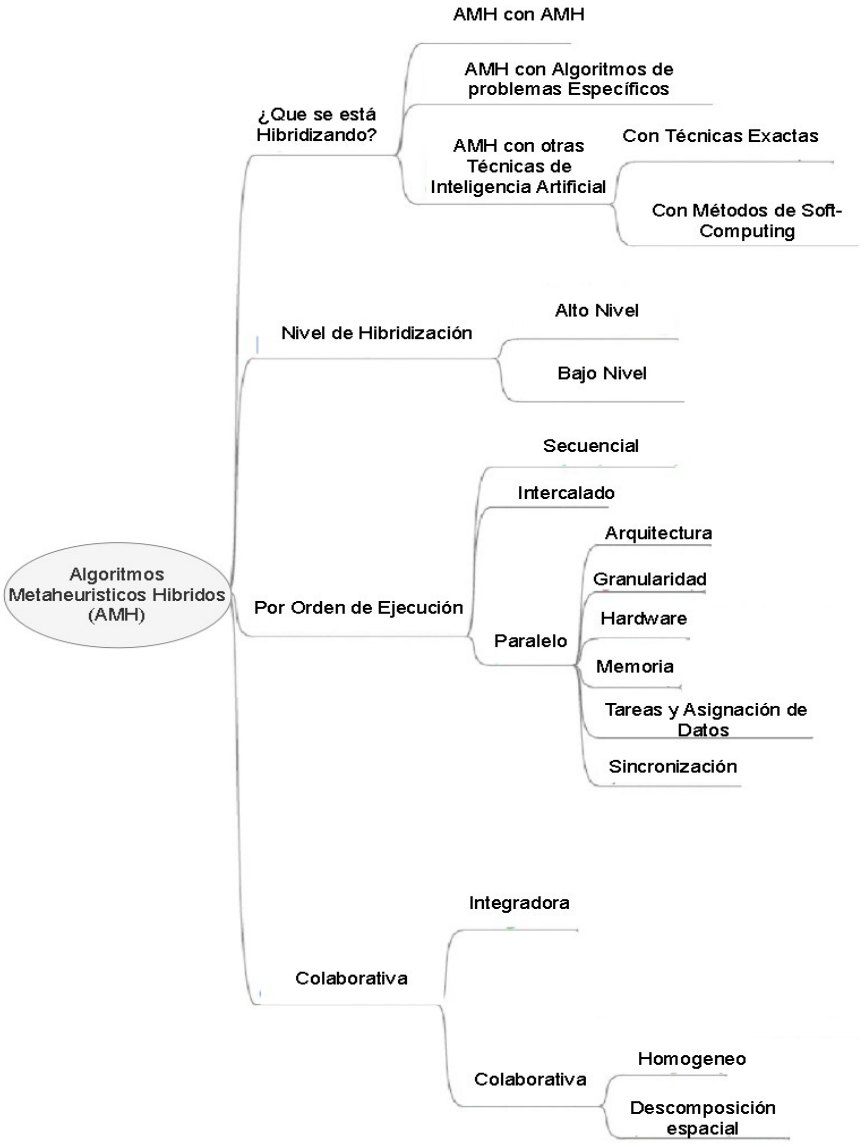


Figura 2.15: Clasificación de algoritmos metaheurísticos según Raidl [6]

2.1. Antecedentes

Para realizar el presente trabajo se realizó una revisión de estudios previos relacionados con el tema y que han sido un aporte en la realización de este trabajo. A continuación se muestran dichos estudios:

Para poder establecer un punto de partida para este proyecto, se utilizaron los siguientes libros y artículos con el fin de establecer una base teórica sobre la cual estará fundamentada

el resto de la investigación.

- **Beamforming de arreglos planos MIMO Masivo usando optimización metaheurística** Esta investigación se encuentra en desarrollo actualmente por Jan Polanco Velasco. En este trabajo de grado presentado por Polanco se propone realizar el Beamforming de una onda electromagnética en un arreglo plano de antenas MIMO Masivo usando una colección definida de Algoritmos Metaheurísticos. Se realizan experimentos en MATLAB en los cuales se considera la geometría física del arreglo y el tipo de usuario, y de acuerdo con la información obtenida por cada algoritmo (tiempo de solución aceptable, cantidad de iteraciones, energía recibida, potencia en decibeles, rango de ángulos, diagramas de radiación optimizados, tasa de convergencia, etc.) se analiza cuales Algoritmos Metaheurísticos son los más adecuados para el Beamforming.[24].

- **PSOGSA-Explore: A new hybrid metaheuristic approach for beampattern optimization in collaborative beamforming,** realizado por S. Jayaprakasam, S. Rahim y Chee Yen Leow en el año 2015. En este articulo se presenta un sistema de beamforming colaborativo el cual sufre de lóbulos laterales muy grandes producto del posicionamiento de los nodos. Partiendo de este problema se presenta un algoritmo híbrido de optimización metaheurística llamada Particle Swarm Optimization and Gravitational Search Algorithm-Explore (PSOGSA-E) con el fin de suprimir los niveles de picos de los lóbulos laterales (peak sidelobe level). El PSOGSA-E esta basado en el PSOGSA. El PSOGSA es el algoritmo obtenido de la combinación del algoritmo de búsqueda gravitacional con el particle swarm optimization (PSO) y permite la exploración con el fin de evitar convergencia prematura. Por ser un algoritmo que combina las funcionalidades de dos algoritmos ya existentes es considerado, según la taxonomía de Talbi [7],como un algoritmo híbrido de trabajo en equipo de bajo nivel(low-level teamwork hybrid algorithm). Las principales diferencias entre el PSOGSA y el PSOGSA-E propuesto en este artículo es que el segundo utiliza 1 parámetro de ajuste en vez de 3 que normalmente usa el PSOGSA lo cual reduce es espacio de búsqueda para parámetros ideales de un plano tridimensional a un plano unidimensional y por ende la complejidad de ajuste de parámetros se ve simplificada. Adicionalmente el PSOGSA-E no reduce la exploración a medida que aumenta el número de iteraciones, es decir el algoritmo propuesto aun mantiene la habilidad de continuar explorando y así evitar el problema de la convergencia prematura. Finalmente,las simulaciones del PSOGSA-E muestran que su desempeño es mejor que el de los algoritmos PSO, GSA y PSOGSA ya que se obtienen mejores resultados, de manera más rápida y una mejora de hasta 100 por ciento en la reducción del PSL (peak side lobe) mediante la estimación del mejor peso para cada nodo.[36]

- **Hybrid Metaheuristic Algorithms: Past, Present, and Future** realizado por

T.O. Ting, Xin-She Yang, Shi Cheng and Kaizhu Huang en el 2014. En este artículo se explica el rol que juegan los algoritmos híbridos en la mejora de la capacidad de búsqueda de los algoritmos. Adicionalmente se da a conocer que el objetivo de la hibridación es combinar las ventajas de cada algoritmo para formar un algoritmo híbrido el cual busca al mismo tiempo que intenta minimizar cualquier desventaja sustancial. En general, el resultado de la hibridación puede generar algunas mejoras en términos de velocidad o precisión computacional. Este articulo analiza los avances recientes en el área de la hibridación de diferentes algoritmos al igual que establece las formas y métodos de hibridación y por último se sugieren algunas recomendaciones cruciales para un mayor desarrollo de estos en el futuro. [6]

- **A Taxonomy of Hybrid Metaheuristics** Escrito por HG Talbi en el 2002, se presenta una taxonomía de los algoritmos metaheurísticos híbridos en un intento de proporcionar una terminología común y mecanismos de clasificación. Dentro de este articulo el autor establece los criterios que se deben tomar en cuenta al momento de diseñar un algoritmo híbrido, la clasificación que surge de estas consideraciones y las diferencias entre los principales grupos de algoritmos. Como ilustración de la utilidad de la taxonomía se proporciona una bibliografía anotada que clasifica un gran número de enfoques híbridos según la taxonomía. [7]

- **Performance Enhancement for Adaptive Beam-Forming Application Based Hybrid PSOGSA Algorithm** realizado por Ahmed Magdy, Osama M. EL-Ghandour, Hesham F. A. Hamed en el año 2015. En este artículo se presenta un algorítmo basado en la técnica PSOGSA (Combinación del Gravitational Search Algorithm y el Particle Swarm Optimization) con el fin de mejorar el desempeño de sistemas de antenas inteligentes usando N elementos para arreglos con geometría circular uniforme. En este caso el algoritmo PSOGSA es diseñado como si fuese un algoritmo híbrido co-evolutivo heterogéneo de bajo nivel (low-level co-evolutionary heterogeneous hybrid). Esto se debe a que el algoritmo combina la funcionalidad de ambos algoritmos (bajo nivel), los algoritmos se corren en paralelo y no de manera secuencial (co-evolutivo), y se usan dos algoritmos diferentes para producir un resultado final (heterogéneo). Finalmente, los resultados muestran una mejora en términos de la potencia recibida y una convergencia global más rápida y robusta del PSOGSA en comparación al GSA.[37]

- **Social Emotional Optimization Algorithm for Beamforming of Linear Antenna Arrays** realizado por Gopi Ram, P. S. Pal, D. Mandal, R. Kar y Sakti Prasad Ghosal en el año 2014. En este artículo se plantea un algoritmo metaheurístico basado en la interacción social de humanos social emotional optimization algorithm (SEOA) el cual se usará para determinar la mejor configuración de los pesos del factor de arreglo que maximicen el lóbulo principal de un arreglo plano de antenas. SEOA es un

algoritmo de optimización estocástico basado en el comportamiento de una población en la cual cada individuo simulado representa a una persona. Todos los individuos en este modelo se comunican entre si a través de cooperación y competencia para buscar aumentar su estatus social, el ganador (aquel con mayor estatus social) es la solución. Con este algoritmo se buscará reducir el nivel de los lóbulos laterales y la reducción del ancho del haz principal. Como estrategia se define la ecuación del factor de arreglo plano, se establecen las características que va a tener (SEOA) y se realiza el experimento para 10, 14 y 20 elementos del arreglo plano, posteriormente se contrasta el nivel de los lóbulos laterales con respecto al lóbulo principal para un rango de grados entre [-100,100]. Este estudio se lleva a cabo para determinar los mejores pesos óptimos de excitación de corriente y el espaciado óptimo entre elementos de los conjuntos de antenas lineales. Adicionalmente también se observa el efecto que tiene el SEOA sobre la reducción de ancho del haz entre los primeros nulos (FNBW) y la relación del lóbulo principal a secundario (SLL). Como resultado se obtuvo en este estudio que el uso del SEOA ofrece una reducción considerable en el SLL y una mejora en cuanto al FNBW en comparación a arreglos convencionales de antenas no optimizadas. Adicionalmente se encontró que este algoritmo requiere menos tiempo de ejecución para encontrar diseños óptimos de arreglos de antenas. [40].

- **An effective hybrid cuckoo search algorithm for constrained global optimization** realizado por W. Long, X. Liang, Y. Huang y Y. Chen. en el año 2014. En este articulo se propone un algoritmo híbrido basado en el algoritmo de búsqueda de cuckoo el cual se combina con el método del lagrangiano aumentado y el método de búsqueda local de Solis y Wets. Se utiliza la busqueda de cuckoo como optimizador global mientras que el método del langrangiano aumentado es implementado es para el manejo de restricciones y el método de Solis y Wets actúa como un optimizador de búsqueda local. Según la taxonomía de Talbi esta hibridación podría clasificarse como un algoritmo de relevo de bajo nivel. Tras una serie de pruebas y experimentos se concluye que la hibridación proporciona una compensación más efectiva entre la explotación y la exploración del espacio de búsqueda. El algoritmo propuesto en este documento demuestra ser más competitivo y eficiente en comparación con los demás. Sin embargo, también se destaca que una investigación más profunda debe ser llevada a cabo para así poder examinar las eficiencias del algoritmo propuesto en otros problemas de optimización del mundo real y de optimización a gran escala. [41]

- **Antenna Theory: Analysis and Design** escrito por Constantine A. Balanis en el año 2016. En el capítulo 6 de este libro se establece la información relacionada a los arreglos de antenas, sus ecuaciones de factor de arreglo y aspectos relevantes sobre el control del patrón de radiación como su geometría (linear, circular, rectangular), dis-

tancia entre los elementos del arreglo, amplitud de los elementos del arreglo, fase de los elementos del arreglo y patrón de radiación relativo de cada elemento. Adicionalmente, en el capitulo 16 se habla acerca de la teoría del arreglo de antenas, Beamforming Adaptativo, técnicas de Beamforming optimas, algoritmos de procesamiento digital adaptativo, redes móviles ad hoc, diseño de antenas, simulación e impacto del diseño de antenas, capacidad/rendimiento de la red y tasa de errores de bits.[1]

- **Engineering Optimization: An Introduction with Metaheuristic Applications,** realizado por Xin-She Yang en el año 2011. A lo largo de este libro se explican algunos de los diferentes tipos de algoritmos metaheurísticos que existen y los cuales fueron considerados en el trabajo de Polanco [24]. Entre estos tenemos (GA), Simulated Annealing(SA), Particle Swarm Optimization (PSO). Para cada algoritmo se hace una introducción, se explica el procedimiento básico, selección de parámetros, características especiales que pueda tener y como sería su implementación en MATLAB. [21]

- **Biologically Inspired Optimization of Antenna Arrays** Este artículo proporciona una introducción a los algoritmos genéticos (GA), la optimización de enjambres de partículas (PSO) y la optimización de colonias de hormigas (ACO). Se presentan varios ejemplos de optimización de redes de antenas para ilustrar el poder de estos algoritmos. Este artículo presenta tres aplicaciones de optimización de conjuntos de antenas que se resuelven utilizando tres algoritmos de búsqueda global diferentes. Estos algoritmos no garantizan la "mejor" solución, pero suelen encontrar muy buenas soluciones que cumplen con las especificaciones. Finalmente se concluye que solo los algoritmos de GA y PSO pueden ser utilizados para optimizar un arreglo de antenas sin embargo no se concluye cual de los dos es una mejor opción. [42]

- **Hybrid Metaheuristics An Emerging Approach to Optimization** realizado por Christian Blum, Maria Jose Blesa Aguilera, Andrea Roli and Michael Sampels en el año 2008. En el primer capítulo de este libro se realiza una breve introducción al tema y se establecen los conceptos claves para entender que son los Algoritmos Metaheurísticos híbridos tales como métodos de búsqueda local, la metaheurística, y algunos algoritmos básicos destacados. Adicionalmente, se plantea nuevamente que los algoritmos híbridos pueden ser clasificados como algoritmos integradores y algoritmos colaborativos [39]. En los demás capítulos del texto se explican casos muy específicos de estos algoritmos, los criterios que se utilizan al momento de elaborar estos algoritmos híbridos, y también se establecen ejemplos en los que estos algoritmos han sido aplicados. [38]

2.2. Diseño del algoritmo metaheurístico híbrido

Para desarrollar el algoritmo metaheurístico híbrido deseado se tomaron en cuenta tanto aspectos de la taxonomía de Talbi como de la taxonomía de Raidl. En cuanto a Talbi se propuso que los algoritmos únicamente fuesen de alto nivel ya que se considera a cada uno de los algoritmos de manera individual y su combinación se realizará utilizando la totalidad de cada algoritmo en vez de ciertos elementos de cada uno de ellos. Adicionalmente los algoritmos utilizados serán algoritmos heterogéneos ya que contamos con 5 algoritmos metaheurísticos y el objetivo de este trabajo es combinar los diversos algoritmos ya existentes en vez de diferentes variaciones de un mismo algoritmo. En cuanto a la taxonomía de Raidl, podemos decir que el algoritmo a diseñar es de hibridizacion de algoritmo metaheurístico con algoritmo metaheurístico, tiene un nivel de hibridizacion alto ya que combina algoritmos completos, y se utilizarán estrategias tanto integradoras como colaborativas. [7] [39]

Adicionalmente, es importante considerar la clasificación de los algoritmos metaheurísticos. Los algoritmos metaheurísticos se pueden dividir en dos grandes grupos, algoritmos basados en población y algoritmos de solución simple. Los algoritmos de solución simple manipulan y transforman una única solución durante la búsqueda, mientras que en los algoritmos basados en población desarrollan una población completa de soluciones. Los algoritmos de solución simple están orientados a la explotación, es decir tienen el poder de intensificar la búsqueda en las regiones locales mientras que los algoritmo basados en población están orientados a la exploración, es decir dichos algoritmos permiten una mejor diversificación ya que la búsqueda se genera sobre todo el espacio. En el caso de esta investigación, GA, BAT, PSO y CUCKOO se comportan como algoritmos basados en población y SA como un algoritmo de solución sencilla [7].Dichas características se establecen como punto de partida y dentro de ellas se considerarán las diferentes posibilidades que existen para combinar los algoritmos.

Como motivo de llevar a cabo esta experimentación o búsqueda de posibilidades de profundizar en la hibridizacion de cada algoritmo, a continuación se presentan dos propuestas de hibridizacion que siguen las clasificaciones de taxonomía ya expuestas mientras prueban variaciones o estrategias de hibridación nuevas.

2.2.1. Estrategia de Hibridación 1: Configuración de isla con Migración asíncrona por solicitud (MAS)

La primera propuesta de hibridación es utilizar una estrategia de combinación híbrida de trabajo en equipo de alto nivel. Esta estrategia se caracteriza por ser un modelo de optimización en los que muchos agentes cooperantes evolucionan de forma paralela y buscan cooperar entre si.

Un ejemplo utilizado de manera recurrente en este tipo de estrategia es la configuración de isla. En un modelo de isla, cada algoritmo mantiene su propia subpoblación, y dichos algoritmos trabajan en conjunto intercambiando periódicamente una parte de sus poblaciones en un proceso llamado migración. Esta migración depende de dos parámetros. El primero es el intervalo de migración. Este intervalo se refiere a el número de iteraciones entre una migración o el criterio utilizado para determinar cuando una migración se debe llevar a cabo. El segundo parámetro es el tamaño de la migración, lo cual indica el número de individuos de la población a migrar [7].

A diferencia del modelo de isla tradicional, en esta investigación se realizaron ciertos ajustes a dicho modelo. El primer ajuste consistió en evitar el uso de un número fijo de iteraciones que determine cuando se debe hacer cada migración. Para esto se utilizó un parámetro en el que cada vez que un algoritmo detecte que está estancado solicita ayuda a un algoritmo vecino y en caso de que este tenga un conjunto de soluciones que pueda ayudar al algoritmo estancado, estas serán enviadas como población migrante. A esto se le conoce como migración por solicitud.[43]

La forma en que funciona la migración por solicitud es bastante sencilla. Cada algoritmo se está ejecutando de manera paralela. Si algún algoritmo no consigue una mejor solución después de un número determinado de iteraciones, envía una señal de ayuda. Todos los demás algoritmos al recibirla evaluarán si su soluciones son mejores a las del algoritmo estancado, y en base a esto enviarán su población migrante para ayudarlo.

Al momento de realizar la migración debemos tener un criterio para poder decidir cuales soluciones son las que deseamos enviar al algoritmo estancado. Inicialmente se propuso enviar el 25 por ciento de las mejores soluciones. Sin embargo esta estrategia resultó muy elitista y por lo tanto con muy poca variabilidad entre las soluciones migrantes. Para tratar de enviar poblaciones migrantes mas diversas, el tamaño de población migrante será función de la variación de cada una de las soluciones con respecto

a la mejor solución global. En otras palabras, consideremos una matriz de soluciones de $[n, m]$ donde n es el tamaño de la población o el número de soluciones disponibles y m es la cantidad de elementos de la antena. Con esta matriz de soluciones buscaremos el valor de cada fila evaluada en la función de costo obteniendo asi n soluciones las cuales se compararán con la mejor solución global. Finalmente, las soluciones se clasificarán según su variabilidad con respecto a la mejor solución global, y un número determinado de estas serán migradas al algoritmo estancado. Como criterios de parada, los correspondientes a este modelo de hibridación son: a) un numero máximo de iteraciones predeterminado, b) el estancamiento de cada uno de los algoritmos y c) el error entre la mejor solución y la solución deseada.

MAS (Migración Asíncrona por solicitud) presenta una modificación adicional al modelo original y es la inclusión de una etapa de refinamiento tal como se ilustra en el esquema del modelo en la figura 2.16. En este caso los algoritmos utilizados en cada una de los bloques de la isla son algoritmos de población. Sin embargo, la etapa de refinamiento se utilizará un algoritmo de solución individual (SA). Esta decisión se lleva a cabo teniendo en cuenta que los algoritmos basados en población nos ayudan a explorar de manera mas completa el área del problema mientras que el algoritmo de solución individual utiliza su capacidad de explotación para obtener un resultado mas refinado. [7]

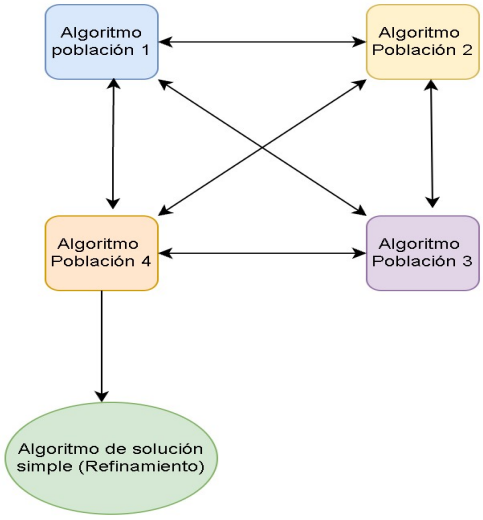


Figura 2.16: Configuración de isla: migración asíncrona por solicitud (MAS)

Algorithm 6 Island Configuration Excecution

Require: algorithm parameters, initial weight vector, incoming vectors form other blocks

```
initialize variables
while error ($e_i$ is larger than Tolerance) AND (Help signals for all algorithms are 0) AND (iteration count is smaller than Max iteration) do
    Execute algorithm block 1
    Execute algorithm block 2
    Execute algorithm block 3
    Execute algorithm block 4
Select Best solution
Execute Single Solution Algorithm
Compute error
Select best solution
compute error
```

2.2.2. Estrategia de Hibridación 2: Relevo y Búsqueda en Equipo (RBE)

La segunda propuesta de hibridación es una estrategia de combinación híbrida de relevo de alto nivel. Esta estrategia se caracteriza por ser un modelo de optimización en el que los diferentes algoritmos están conectados de manera secuencial. Por lo tanto, la salida del primer algoritmo es igual a la entrada al segundo. Adicionalmente, en esta configuración planteamos dividir la población de manera equitativa, de tal manera que cada algoritmo trabaje con una parte de la población total. De esta manera se divide el costo computacional del problema entre los algoritmos disponibles [7].

Continuando con la idea de la configuración anterior, en este caso también se utilizan algoritmos basados en población para hacer una búsqueda más global sobre la superficie. Posteriormente, cada algoritmo tendrá su propia etapa de refinamiento utilizando el algoritmo de Recocido de materiales frío (CSA). Con esta idea se propone utilizar inicialmente el algoritmo de CSA como una etapa de refinamiento determinista, y luego seleccionar a la mejor solución propuesta entre todos los algoritmos, para refinar por ultima vez este resultado con un algoritmo de solución individual y de búsqueda aleatoria como lo es SA tal como se ilustra en la figura 2.17. Como criterios de parada dentro de cada uno de los problemas se tiene a) un número máximo de iteraciones que debe ser alcanzado b) el error entre la solución deseada y la mejor solución y en el caso de SA y CSA c) haber alcanzado el número máximo de soluciones rechazadas [21].

Algorithm 7 Island Configuration Block

Require: algorithm parameters, initial weight vector, incoming vectors form other blocks

```
Set initial solutions matrix
if Migrating population availabe then
    organize solutions from best to worst
    replace worst solutions with incoming migrating population
if Algorithm 2 needs helpd then
    if Current best solution is better than Algorithm 2 best solution then
        calculate average value of each column from solution matrix
        subtract Best Global value with each solution
        sort solutions into groups according to how much they vary in comparisson to Best Global value
        select 2 solutions from each group and place them in an output vector.
if Algorithm 3 needs helpd then
    if Current best solution is better than Algorithm 3 best solution then
        calculate average value of each colum from solution matrix
        subtract Best Global value with each solution
        sort solutions into groups according to how much they vary in comparisson to Best Global value
        select 2 solutions from each group and place them in an output vector.
if Algorithm 4 needs helpd then
    if Current best solution is better than Algorithm 4 best solution then
        calculate average value of each colum from solution matrix
        subtract Best Global value with each solution
        sort solutions into groups according to how much they vary in comparisson to Best Global value
        select 2 solutions from each group and place them in an output vector.
Execute algorithm
Select best solution
Calculate error
if error is equal to previous error then
    stuckcount=stuckcount+1
if stuckcount is greater than maxstuckiteration then
    send help signal to other algorithms
```

Algorithm 8 Relay Configuration

Require: algorithm parameters, initial weight vector, incoming vectors form other blocks

initialize variables
while error (e_i is larger than Tolerance) AND (iteration count is smaller than Max iteration) **do**
    Execute algorithm 1
    Compute error
    Execute algorithm 2
    Compute error
    Execute algorithm 3
    Compute error
    Execute algorithm 4
    Compute error
Select Best solution for algorithm 1
Select Best solution for algorithm 2
Select Best solution for algorithm 3
Select Best solution for algorithm 4
Execute Cold SA for solution from algorithm 1
Execute Cold SA for solution from algorithm 2
Execute Cold SA for solution from algorithm 3
Execute Cold SA for solution from algorithm 4
Select best solution out of all Cold SA outputs
Execute SA for best global solution
compute error

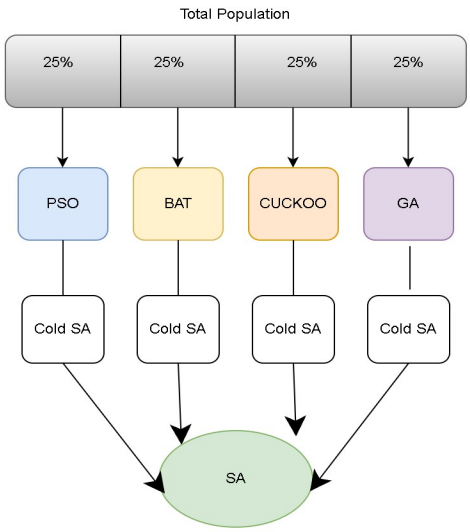


Figura 2.17: Configuracion de Relevo y Búsqueda (RBE)

Capítulo 3

Estructura del sistema

3.1. Plan de pruebas

A continuación se presenta la descripción de las pruebas y el plan de pruebas diseñado con el que se evaluaron las estrategias de hibridación. Las siguientes pruebas se diseñaron con la intención de validar el funcionamiento adecuado de los algoritmos antes de ejecutar los experimentos. Posteriormente, se detalla el funcionamiento y ejecución del experimento general que es la parte central de esta investigación.

3.1.1. Pruebas y Ajustes previos

Antes de llevar a cabo los experimentos principales de la investigación, se llevaron a cabo ciertas pruebas para verificar el funcionamiento adecuado de los algoritmos a utilizar.

3.1.1.1. Validación del funcionamiento de los Algoritmos Metaheurísticos con las funciones de Optimización de Referencia

En esta parte se buscó determinar que el comportamiento de los algoritmos fuese el adecuado y así al momento de probarlos con las funciones de Optimización de Referencia previamente definidas, poder verificar que efectivamente están operando según corresponde. En este caso se observó si los algoritmos eran capaces de acercarse a las soluciones teóricas, que tan efectivos son en términos de cuantas veces se aproximan a la solución, y por último verificar que su comportamiento concuerde con su pseudocódigo y descripción teórica.

3.1.1.2. Sintonización de algoritmos

Todos los algoritmos utilizados en esta investigación tienen como entrada una serie de parámetros o variables que determinan el comportamiento del algoritmo. En esta prueba se utilizó el experimento general, con todos los algoritmos individuales para encontrar cuales son los valores de dichos parámetros que permiten un mejor rendimiento. Para este ajuste se irán variando los parámetros de manera gradual para cada algoritmo con el fin de observar que efectos tienen estos sobre el número de estancamientos y minimización del error. Los valores obtenidos tras esta sintonización serán utilizados en los experimentos posteriores.

3.1.1.3. Ajuste de tamaños de grupos migratorios y tiempo de migración para estrategia MAS

Para la estrategia de hibridación en configuración de isla con migración asíncrona por solicitud (MAS) se tiene un proceso de migración entre los diversos algoritmos utilizados, sin embargo el tamaño de las poblaciones migrantes viene dado por la cantidad de individuos que se decida migrar. En esta prueba se determinó el número ideal de individuos y el número de iteraciones de estancamiento adecuado para poder llevar a cabo cada migración. Con esto se fijó un número de individuos migrantes que permitan que la ejecución del algoritmo sea lo suficientemente rápida y precisa.

3.1.2. Condiciones y descripción del experimento general

El experimento en general se llevó a cabo con la idea de que una antena o punto de acceso pueda estar ubicado en un espacio cerrado o abierto, sin obstáculos que puedan causar alguna interferencia sobre la señal y que esta pueda ser transmitida a través de un mecanismo de conformación de haces a los usuarios que se encuentran en dicho entorno tal como lo ilustra la figura 3.1.

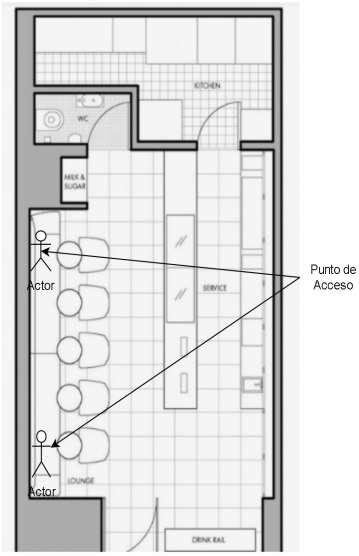


Figura 3.1: Diagrama Ilustrativo de escenario donde se desarrolla el experimento

El objetivo del experimento general es poner a prueba la capacidad que tienen los algoritmos metaheurísticos y las estrategias de hibridación para generar vectores de peso o un factor de arreglo con los cuales la potencia irradiada vaya dirigida hacia donde están ubicados los usuarios.

El algoritmo en principio es ciego y en ningún momento recibe como entrada un valor o ángulo que le indique donde se ubican los usuarios. Para saber donde están ubicados los usuarios se obtiene un diagrama de radiación de referencia tal como se ilustra en la figura 3.2. Tal como se puede observar este diagrama muestra con respecto a un ángulo theta en un rango de 0 a 180 grados, los ángulos donde la potencia deseada es mayor. Con esta información el algoritmo no sabe explícitamente donde está cada usuario, pero si sabe hacia que dirección debe conformar y apuntar los haces de la antena para tratar de igualar la potencia del diagrama de referencia.

Para realizar la representación de los usuarios se utilizan funciones gaussianas por su características geométricas. Las funciones gaussianas vienen dadas por 3 parámetros α (ancho) β (desplazamiento) y σ (amplitud), gracias a estas características, es posible generar una función que se asimile al haz emitido por una antena. Para poder establecer una relación entre las funciones gaussianas y los haces emitidos, utilizaremos el concepto de ancho de haz o beamwidth. El ancho de haz de un patrón de radiación se define como la separación angular entre los puntos idénticos a lados opuestos del patrón [1]. Para determinar el ancho a utilizar, se consideraron diversos tipos de antenas utilizadas para conexiones punto a punto en la frecuencia de 5Ghz. Para tener una referencia se buscaron antenas de corto, mediano y largo alcance:

- En el caso de corto alcance (0 a 5km) tendremos las antenas TL-WAZ510N de la compañía TPLINK con un ancho de haz de 60 grados en la dirección horizontal [44] .
- Para el caso de distancias medianas (hasta 10km) utilizamos como referencia una antena NANO Station Loco de la compañía Ubiquiti Networks que cuenta con un ancho de haz de 45 grados horizontales [45] .
- Por último, para distancias largas (mas de 15km) tenemos la antena CPE510 de Tplink que ofrece un ancho de haz de 30 grados horizontales [44].

Según se puede observar a medida que el alcance de una antena es mayor su ancho de haz disminuye. Teniendo en cuenta que la idea de este proyecto es tratar de generar haces lo más directivos posibles, consideraremos que los anchos de las funciones gaussianas utilizados para representar la ubicación de los usuarios serán 10 veces menores a los propuestos por las antenas existentes, en otras palabras serán de 6 grados para corto alcance, 4.5 para mediano alcance y 3 grados distancias largas.

La figura 3.2 ilustra un ejemplo de un diagrama de radiación de referencia donde se puede observar que la potencia deseada es mayor sobre los ángulos 20 y 50. Esto indica la presencia de usuarios en dichas posiciones. Tomando esto en cuenta, el algoritmo metaheurístico es

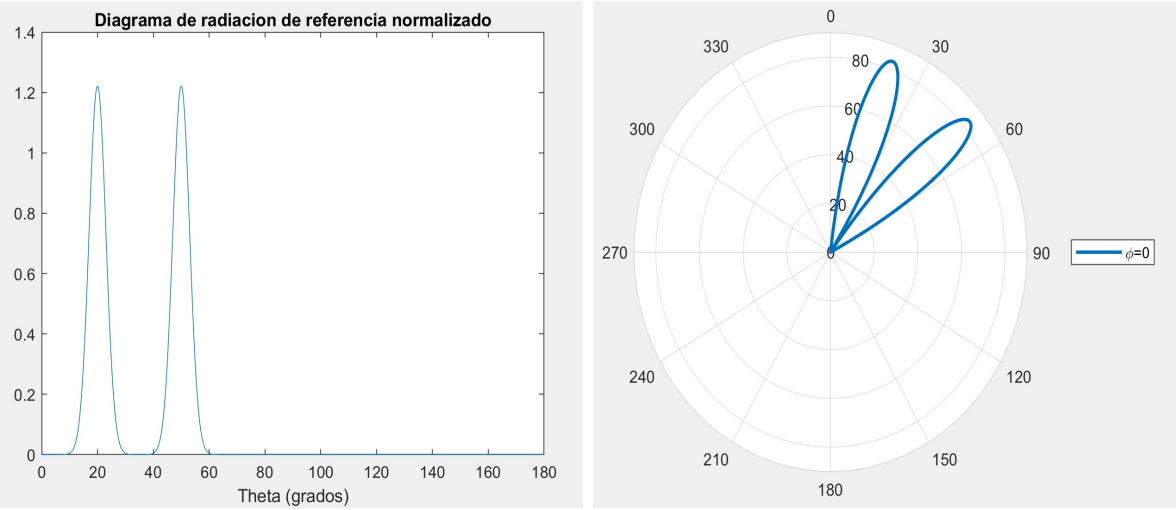


Figura 3.2: a)Diagrama de radiación de referencia normalizado b) Diagrama de radiación de referencia polar

utilizado para ir generando diversos vectores de pesos complejos que al evaluarlos en la función del factor de arreglo obtenemos un diagrama de radicación normalizado tal como se observa en la figura 3.3 a). Posteriormente este factor de arreglo se multiplica por el diagrama de radiación de referencia generando así una propuesta de solución, tal como se ilustra en en la figura 3.3 b). Dicha solución propuesta se comparará con el diagrama de radiación de referencia ilustrado en la figura 3.2 a), para evaluar si se aproxima lo suficiente a la solución deseada. En este experimento se tiene una función multiobjetivo que considera dos variables de optimización que se buscan mejorar en cada iteración. El primero es la maximización de la norma euclidiana de la solución obtenida por el producto del factor de arreglo normalizado y el diagrama de radiación de referencia. Esta variable busca que la norma euclidiana calculada para la solución propuesta sea cada vez mayor y se aproxime lo mas posible a la norma euclidiana del diagrama de radiación de referencia. En este caso la norma se calcula tal como lo indica la 3.1

$$\underset{Max}{fitness_1} = \|AF[w_i, \theta, 0] \cdot DR_{ref}\|_2 \tag{3.1}$$

Donde w_i es la matriz de pesos de la partícula i, $\theta \in [0, \pi]$ y DR_{ref} el diagrama de radiación de referencia.

Para el segundo objetivo de optimización, se busca la minimización de la diferencia entre los lóbulos del diagrama de radiación. Este segundo criterio hace referencia a que se desea

que la diferencia entre los puntos máximos de cada lóbulo sea lo mas pequeña posible con la intención de que los lóbulos sean lo mas similares posible en tamaño y toda la energía no sea irradiada por un solo lóbulo tal como se observa que sucede en la figura 3.3 a).

$$\underset{Min}{fitness_2} = \text{máx}\, L1\left(AF[w_i,\theta,0]\cdot DR_{ref}\Big|_{\theta_j=1,2,3}\right) - \text{máx}\, L2\left(AF[w_i,\theta,0]\cdot DR_{ref}\Big|_{\theta_j=1,2,3}\right) \tag{3.2}$$

Donde w_i es la matriz de pesos de la partícula i, $\theta_j = 1, 2, 3...$ los ángulos de los distintos usuarios y $L1$ el lóbulo 1 y $L2$ el lóbulo 2.

Tomando esto en cuenta, el algoritmo genera soluciones diferentes en cada iteración y estas son evaluadas bajo los criterios de optimización previamente mencionados. Finalmente, partiendo del hecho que se desea maximizar la potencia y minimizar la diferencia entre lóbulos, el problema se plantea como una maximización de $fitness_3$ tal como se muestra en la ecuación 3.3

$$\underset{Max}{fitness_3} = \frac{fitness_1}{fitness_2} \tag{3.3}$$

la solución propuesta que mejor optimice dicha variable es elegida como la mejor solución. Como salida, se obtiene el vector de pesos complejos o el factor de arreglo responsable de producir la mejor solución, y este es utilizado en la antena para dirigir la potencia en la dirección deseada.

Una vez obtenido el vector de pesos complejos correspondientes a la mejor solución propuesta por el algoritmo, se representa diagrama de radiación resultante sobre el plano polar. Tal como se planteo en el anteproyecto, se consideraron 3 tipos de situaciones donde los usuarios podrán ser estáticos, nómadas o mixtos. El tipo de antena utilizado es un arreglo rectangular antenas, con 64 elementos isotópicos 8×8, y con un espacio entre cada elemento de $d_x = d_y = 0{,}5\lambda$, la superficie horizontal de cobertura de la antena es de un rango entre 0 a 180 grados (Angulo Azimutal θ) y el número de individuos o partículas utilizadas en cada uno de los experimentos es de 40.

3.1.3. Experimento 1: Usuarios estáticos

3.1.3.1. Dos Usuarios estáticos

Se ejecuta el experimento general previamente descrito con 2 usuarios de ubicación estática. Estos dos usuarios son ubicados en las posiciones de 20 y 50 grados sobre el eje theta y se caracterizan por mantenerse siempre en la misma ubicación. La antena se ubica a una

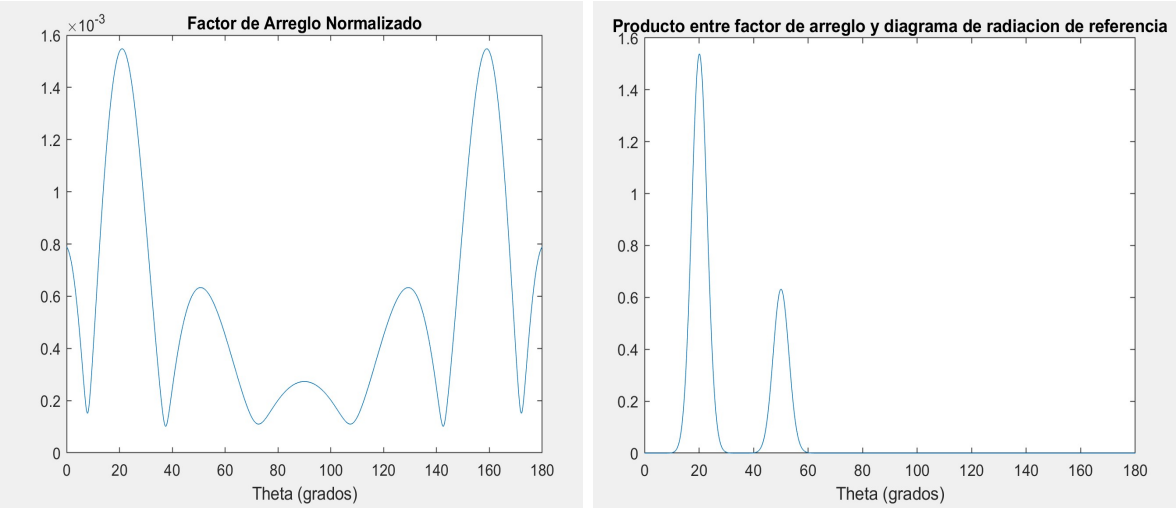


Figura 3.3: a) Factor de arreglo normalizado b) Producto del Factor de arreglo normalizado con el diagrama de radiación de referencia

altura constante, y se asume que los usuarios están ubicados de manera equidistante a la antena y el desplazamiento sobre el eje phi (vertical) no será tomado en cuenta.

Tal como se explicó previamente, en este tipo de experimentos el algoritmo va generando nuevas soluciones (vectores de peso) que al evaluarlos en la función de factor de arreglo permiten obtener un diagrama de radiación el cual se compara con el diagrama de radiación de referencia. A medida que dicha función se asemeje más a el diagrama de radiación deseado, el algoritmo irá convergiendo hasta obtener una solución adecuada.

Este experimento se diseña con la intención de estudiar como es el comportamiento del algoritmo en el caso que dos usuarios siempre mantengan su posición y así poder observar la potencia que cada uno de los usuarios recibe, el tiempo de ejecución y el número de iteraciones que necesarias para llegar al mejor resultado.

3.1.3.2. Tres Usuarios estáticos

Esta situación consiste en ejecutar nuevamente el mismo experimento general descrito pero con la intención de evaluar el rendimiento del algoritmo cuando se toma en cuenta un usuario adicional que en este caso estará ubicado a 80 grados junto con los otros dos ubicados en 20 y 50 grados respectivamente. En cuanto a la ejecución se realiza el mismo experimento solo que en este caso el diagrama de referencia tendrá 3 funciones gaussianas (Una utilizada para representar a cada usuario) con la intención de generar un lóbulo para cada usuario.

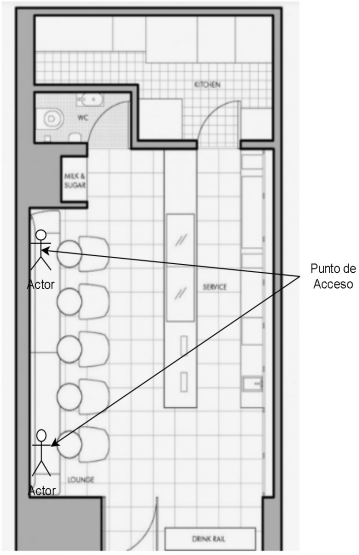


Figura 3.4: Situación con 2 Usuarios estáticos

Más allá de que este experimento se asimila al anterior también nos brinda información nueva. Al momento de considerar 3 usuarios, lógicamente se está contemplando un problema de mayor dificultad para el algoritmo. Esto nos permite evaluar si su desempeño es afectado gravemente por la adición de un usuario nuevo y también nos brinda nueva información sobre su desempeño en cuanto a su maximización de la función objetivo, potencia entregada a cada usuario, tiempo de ejecución y número de iteraciones. Adicionalmente, nos permite observar el proceso de formación de lóbulos en caso de que se agregue un nuevo usuario y comprobar si efectivamente el algoritmo agrega un nuevo lóbulo o simplemente agranda los lóbulos ya existentes.

3.1.4. Usuarios Nómadas

En este experimento se consideran los usuarios como si fuesen usuarios nómadas, esto se refiere a aquellos que se encuentran temporalmente estáticos y luego de un tiempo se realiza un desplazamiento a lo largo del eje theta hacia una nueva ubicación. Al hacer esto el algoritmo deberá nuevamente modificar su diagrama de radiación de referencia y deberá tratar nuevamente de encontrar el diagrama de radiación deseado.

Este escenario se realiza con la intención de simular la situación en la cual los usuarios estén en movimiento ya que en la mayoría de los contextos reales este es un caso posible. Sin

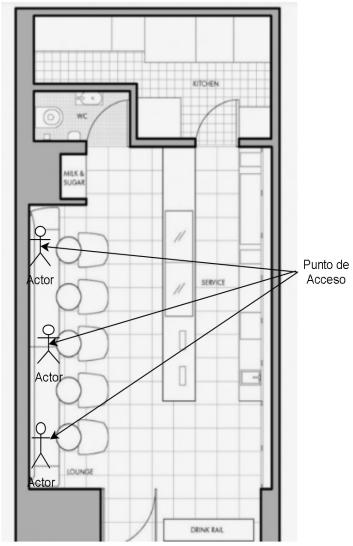


Figura 3.5: Situación con 3 usuarios fijos

embargo, para nuestro caso de estudio, de momento no existe ningún interés en realizar un seguimiento constante sobre los usuarios mientras estén en movimiento. El enfoque principal de este experimento es llevar las antenas a que puedan conformar un haz y dirigirlo a un usuario cada vez que este esté fijo en una posición nueva. En otras palabras, esto quiere decir que una vez los usuarios hayan encontrado su nueva ubicación, y el diagrama de radiación deseado vuelva a ser fijo el algoritmo o estrategia de hibridación procederá a ejecutarse. Para este experimento consideraremos dos usuarios. El primero tendrá una posición inicial de 20 grados y luego migrará hacia una posición de 30 grados mientras que el segundo se ubicará inicialmente en 50 grados y luego migrará a 80 grados.
Este experimento se realiza con la intención de tomar en cuenta que tan rápido puede responder el algoritmo y la capacidad que tiene para ajustarse a contextos nuevos más cercanos a la realidad. Al igual que en el experimento anterior los parámetros a observar en este experimento son la potencia irradiada en la ubicación de los usuarios, el tiempo de ejecución y el número de iteraciones necesarias para llegar al mejor resultado.

3.1.5. Usuarios Mixtos

Por último, para considerar el caso que más se aproxime a la realidad se tienen usuarios de ambos tipos. En esta situación tendremos usuarios que siempre permanecerán fijos y otros que cambiarán su ubicación de manera abrupta.

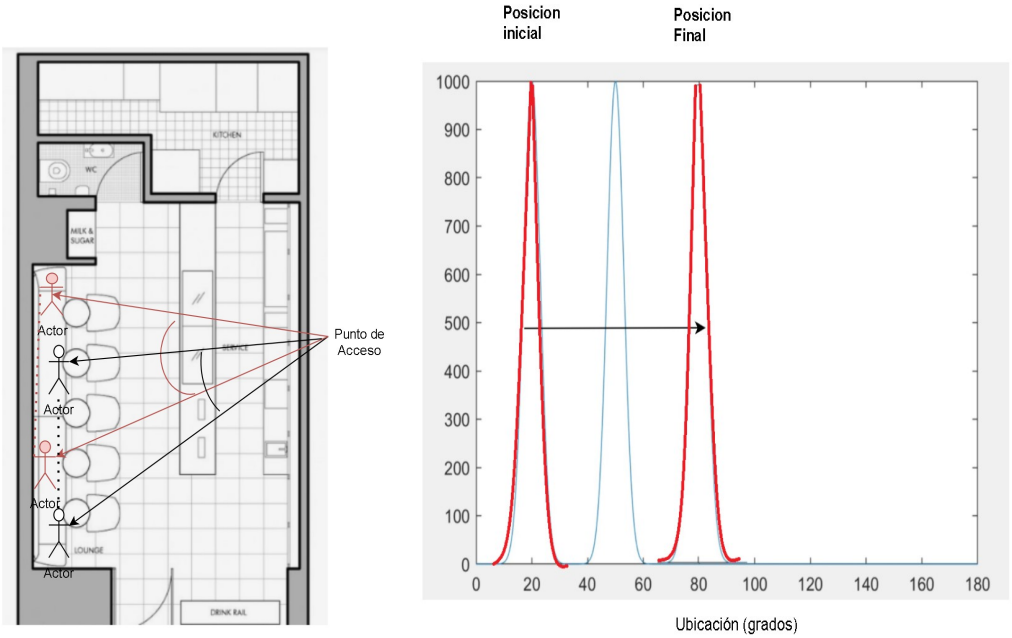


Figura 3.6: a) Situación con usuarios nómadas b) Diagrama de radiación de un usuario nómada.

Esta situación tiene la intención de recrear un escenario que combine todo tipo de usuarios ya que estos, solo en casos muy específicos serán todos de un mismo tipo. Adicionalmente nos permite observar el comportamiento del algoritmo en un contexto similar a la combinación de ambos experimentos previos. Para este último escenario se tendrán 3 usuarios donde uno es fijo y permanece siempre en 160 grados, mientras que los demás son nómadas y migran de 35 a 45 grados y 60 a 80 grados respectivamente. Los parámetros que se tienen en cuenta para este experimento son los mismos que en casos anteriores (Potencia recibida por cada usuario, Tiempo de ejecución, número de iteraciones).

3.1.6. Contextualización de Experimentos

Más allá que la sección anterior brinda una explicación acerca de cómo se llevarán a cabo los experimentos, a continuación, se presenta una breve descripción de escenarios en la vida real para los cuales los experimentos han sido diseñados. Para esto se describe el lugar o el área donde se pretende llevar a cabo los diferentes experimentos y a su vez, se presenta el comportamiento de los diferentes individuos que se encuentran en dicho ambiente. Esto se hace con la intención de aproximarse un poco más hacia lo que será la implementación y dar un contexto lo suficientemente descriptivo para que se pueda entender por qué cada experimento se plantea y se ejecuta así.

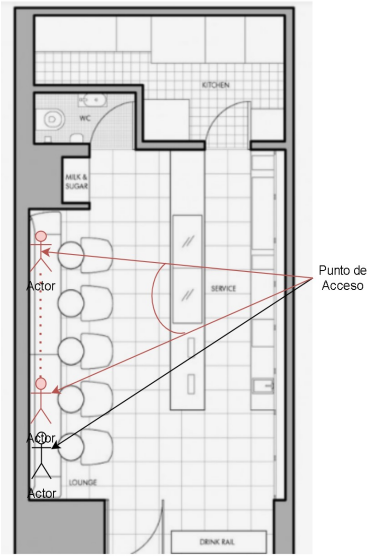


Figura 3.7: Situación con usuarios mixtos

3.1.6.1. Descripción del entorno

Se considera un restaurante pequeño o café similar a cualquiera de los que se podrían encontrar dentro del campus de la Pontificia Universidad Javeriana o zonas aledañas. Este hermoso y elegante restaurante tiene un espacio de 20 metros de largo por 10 metros de ancho donde sus comensales pueden acercarse a probar todo tipo de platos fuertes, ensaladas y postres. En cuanto a su distribución, posee a su izquierda una elegante y extensa barra donde se acostumbran a servir tragos, cócteles y bebidas de todo tipo. Hacia la parte derecha del restaurante se encuentra la cocina donde los exquisitos platillos del menú son preparados. En el centro del restaurante, se ubican 5 mesas distribuidas de manera equidistantes entre si y todas se encuentran ubicadas a la misma distancia de la entrada de dicho restaurante. Ligeramente por encima de la puerta de entrada se ubica un sistema de antenas de última tecnología que se encarga de realizar conformación de haces para que los diferentes usuarios puedan tener una buena conexión al momento de disfrutar de esta experiencia gastronómica.

3.1.6.2. Escenario 1: 2 Usuarios Fijos

En el caso de los usuarios estáticos consideraremos dos individuos. El primer individuo al cual llamaremos U1 estará ubicado en el área del bar, a unos 20 grados con respecto a la antena, y a su vez, tendremos un usuario U2 ubicado en la segunda mesa a aproximadamente unos 50 grados con respecto a la antena. Durante todo este tiempo ambos comensales permanecerán en dicha ubicación y el sistema de antenas irradiará la potencia en las direcciones

de cada uno de los usuarios.

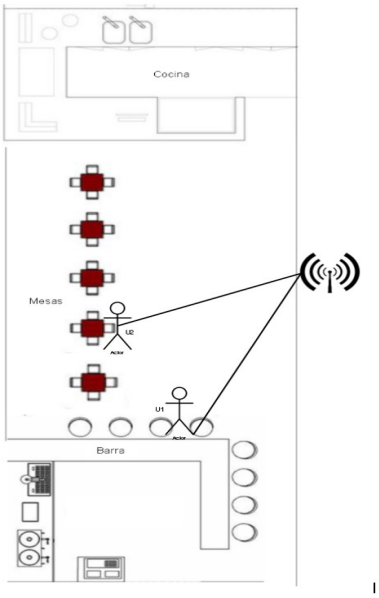


Figura 3.8: Contextualización de situación con dos usuarios fijos

3.1.6.3. Escenario 2: 3 Usuarios Fijos

Considerando el mismo escenario descrito en el ejemplo anterior, tendremos que luego de haber transcurrido unos quince minutos, un joven estudiante al cual llamaremos U3 se aproxima al restaurante y se sienta en la tercera mesa a trabajar en sus quehaceres de la universidad. Paralelo a esto el cliente U1 aun sigue en el bar mientras va por su tercer trago, y el usuario U2 apenas termina su platillo de entrada y está a la espera de plato principal. En este escenario, tendremos que nuevamente los tres usuarios continuarán manteniéndose en la misma posición por un periodo considerable de tiempo y mientras tanto el sistema de antenas buscará irradiar la potencia en las direcciones de 20,50 y 80 grados.

3.1.6.4. Escenario 3: 2 Usuarios Nómadas

Al día siguiente, el usuario U1 vuelve nuevamente al bar ubicándose así a 20 grados con respecto a la antena, sin embargo, esta vez tras un par de tragos decide que le gustaría degustar de la deliciosa gastronomía del lugar. Para esto el usuario U1 se desplaza de su posición inicial en 20 grados hasta la mesa más cercana a él que se ubica a unos 30 grados con respecto a la antena. Paralelo a todo esto un comensal U2 nuevamente ubicado en la segunda mesa a 50 grados nota que la luz solar que entra por la ventana está ligeramente intensa para

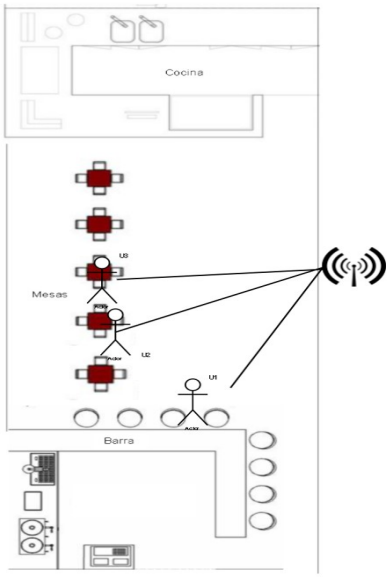


Figura 3.9: Contextualización de situación con tres usuarios fijos

su gusto y decide, justo en el mismo instante en que el comensal U1 se mueve, desplazarse hacia la siguiente mesa ubicada en 80 grados con respecto a la antena. Inicialmente la antena detectó cuando los usuarios se ubicaban en sus posiciones iniciales y dirigió la potencia en estas ubicaciones iniciales (U1=20) (U2=50). Una vez los usuarios se desplazaron hacia su nueva ubicación, la antena nuevamente busca la manera de irradiar la potencia en dichas direcciones (U1=30) (U2=80).

3.1.6.5. Escenario 4: 3 Usuarios Mixtos

Todos los lunes es el día en que se hacen los pedidos de los nuevos ingredientes que se utilizaran en el transcurso de la semana. Para esto el jefe de compras debe estar presente en la cocina para hacer un inventario y al mismo tiempo ponerse en contacto con los proveedores a través de una vídeo llamada. Para esto necesita una buena conexión durante todo este tiempo ya que es un proceso que puede durar horas y debe tener una buena calidad de señal para que en la vídeo llamada se logre observar con detalle los productos adquiridos. En otras palabras, el jefe de compras U3 necesita una conexión constante y por largos periodos de tiempo por lo cual es considerado un usuario fijo que estará en la ubicación de 160 grados. Paralelo a esto, el comensal U1 se ubica en la primera mesa cerca de la barra la cual se encuentra a 35 grados con respecto a la antena. Tras un rato de estar sentado allí nota que su silla se encuentra en un estado muy frágil y decide sentarse en la mesa siguiente que se ubica a unos 45 grados de la antena. Casi al mismo momento un cliente U2 ubicado en la

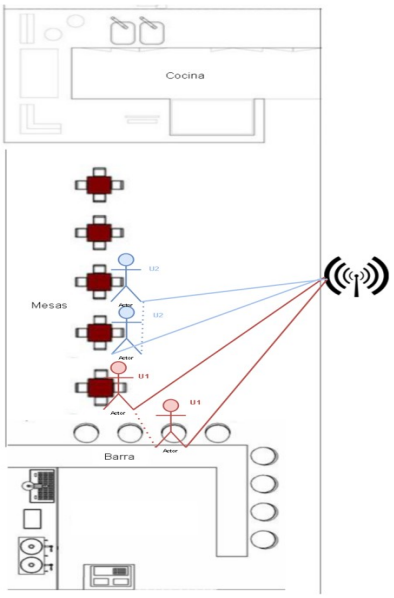


Figura 3.10: Contextualización de situación con dos usuarios Nomadas

tercera mesa a unos 60 grados nota que su mejor amigo de la universidad se acaba de sentar en la mesa de al lado. Por este motivo, decide acercase y preguntar si puede acompañarlo durante el almuerzo. Su amigo acepta, y la nueva ubicación del usuario U2, es de 80 grados con respecto a la antena. Finalmente en este escenario tendremos dos usuarios con posición cambiante (U1=35) a (U1=45) grados y (U2=60)a (U2=80) y el usuario U3 que todo el tiempo permanecerá en la cocina con la ubicación (U3=160).

3.1.7. Lista de Pruebas a llevar a cabo

Finalmente el listado de plan de pruebas queda de la siguiente manera

1. Validación del funcionamiento de los algoritmos metaheurísticos con las funciones de optimización de referencia
2. Sintonización de Algoritmos Individuales
3. Ajuste de poblacion migrante y tiempo de migración para Estrategia MAS
4. Experimento con dos usuarios estáticos
5. Experimento con tres usuarios estáticos
6. Experimento con dos usuarios Nómadas

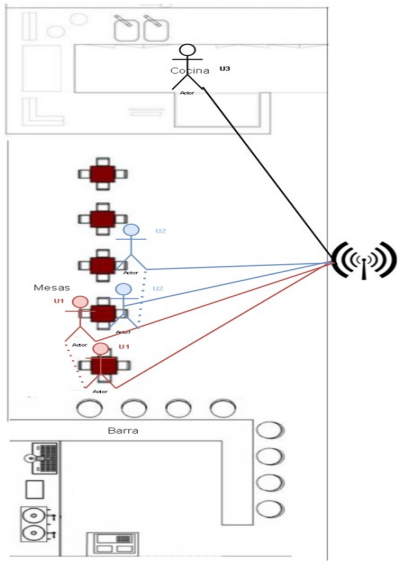


Figura 3.11: Contextualización de situación con dos usuarios Nomadas

7. Experimento con tres usuarios Mixtos

Capítulo 4

Resultados y Discusiones

4.1. Resultados

En este capítulo se muestran los resultados obtenidos tras ejecutar las pruebas y situaciones previamente descritas en secciones anteriores. Inicialmente se llevan a cabo a los experimentos de ajuste o de preparación. Estos experimentos son los encargados de ajustar los parámetros a utilizar por cada algoritmo en los experimentos principales, y a su vez también sirven como validación del funcionamiento de los algoritmos y estrategias de hibridación.

Posterior a estos experimentos se ejecutan tanto los algoritmos metaheurísiticos individuales como las estrategias de hibridación sobre los diferentes escenarios del experimento general. Finalmente se realiza una pequeña validación en la que se muestra como los resultados obtenidos por los algoritmos son validos en otro software utilizado para mostrar diagramas de radiación. Todas las simulaciones se realizaron en un computador portátil usando el software MATLAB R2019B. El computador cuenta con un procesador Intel 5Y70 de 2 Núcleos a 1.1 GHz, Memoria RAM 6 GB y Sistema Operativo Windows 8 Professional a 64 bits.

4.1.1. Sintonización de algoritmos

Considerando que los parámetros de cualquiera de los siguientes algoritmos puede tomar una gran cantidad de diferentes valores, es importante realizar una sintonización de dichos valores. En este caso, la palabra sintonización se refiere a encontrar un rango de valores para cada parámetro en el que este pueda funcionar de manera adecuada para el problema planteado. El primer paso en este proceso fue hacer una revisión de la literatura buscando cuales son los rangos en los que deben estar los valores de cada parámetro y con cuales de estos han obtenido buenos resultados en otras investigaciones. Tras haber determinado rangos de posible funcionamiento, se procedió a ejecutar un escenario del experimento general para evaluar el desempeño de cada algoritmo. Para esto se tomó el problema de conformación de haces con dos usuarios estáticos ubicados en 30 y 60 grados respectivamente y se ejecuto cada algoritmo sobre este problema probando todas las combinaciones posibles para los diferentes parámetros. Finalmente se observó cuales fueron los algoritmos que obtuvieron el menor error

(diferencia entre la norma del diagrama de radiación de referencia y la norma del diagrama de radiación de la mejor solución propuesta) y el menor estancamiento. En el caso de este segundo criterio, se considera como estancamiento a todo escenario en que el error actual e_k sea igual al error en la iteración anterior e_{k-1}. En el caso de este experimento se utilizó como criterio de parada una tolerancia del error (Valor mínimo deseado del error) de $1x10^{-3}$, un conteo de iteraciones estancadas de diez iteraciones y un número máximo de 200 iteraciones. En otras palabras esto quiere decir que el algoritmo se detendrá si el error producido es menor a $1x10^{-3}$, si alcanza las 200 ejecuciones o si tras diez iteraciones no ha habido una mejora o reducción en el error. Finalmente, los resultados son registrados en tablas donde se muestran los valores para cada parámetro, el error mas pequeño que dichos parámetros producen y el número máximo de iteraciones antes de detenerse por estancamiento.

4.1.1.1. Sintonización de PSO

En el caso de PSO se tienen 3 parámetros presentes tal como se puede observar en la ecuación 2.13 donde α es el parámetro asociado con la aceleración local, β es asociado con la aceleración global y θ es el parámetro de inercia. Según la literatura, los valores de dichos parámetros deben estar dentro del rango de 0.1 a 0.4 para el caso de α, θ debe estar entre 0.5 y 0.9 y β en el rango de 0.1 a 0.7 [21].

Según los resultados obtenidos en la tabla 4.1 podemos observar que los valores de la inercia están cercanos al valor máximo que este puede asumir,mientras que beta y alpha están mas cercanos hacia las cotas inferiores de los rangos establecidos. Un valor alto de la inercia significa que la velocidad en cada iteración será similar a la anterior ya que la inercia producirá que la velocidad actual sea 0.9 veces a la de la ejecución previa. Por otra parte, α y β adicionarán a dicha velocidad pequeñas variaciones cercanas al mejor valor local y mejor valor global. En otras palabras, esta combinación genera que el tamaño de paso, o la variación en la velocidad y posición generadas en cada iteración sean pequeñas. Estas pequeñas aproximaciones permiten que el algoritmo busque la solución deseada en las zonas o valores cercanos a la mejor solución actual.

Sintonización Parámetros PSO				
Parámetro	Inercia	Alpha	Beta	Mejor Resultado
Mejor Error	0.8	0.3	0.2	2.25
Menor Estancamiento	0.9	0.1	0.1	184/200

Cuadro 4.1: Sintonización Parámetros PSO

4.1.1.2. Sintonización de GA

En el caso de GA sus parámetros vienen dados por la probabilidad de cruce *pc* y la probabilidad de mutación *pm*. Tal como su nombre lo indica la probabilidad de cruce refleja que tan probable es que dos cromosomas (soluciones) se combinen intercambiando o mezclando partes de cada una de las ellas con el fin de ir convergiendo o aproximándose al valor óptimo deseado. Por otra parte, la probabilidad de mutación indica que tan posible es que ocurra un proceso que cambie alguna de las soluciones de manera aleatoria con el fin de aumentar la diversidad en la población existente. Normalmente la probabilidad de cruce es bastante alta estando entre los 0.7 y 1.0 mientras que la probabilidad de mutación se encuentra entre 0.001 y 0.05 siendo bastante baja con el fin de garantizar que las soluciones se crucen pero solo unas pocas participen en el proceso de mutación.

Según los resultados que se pueden observar en la tabla 4.2 se puede notar que se obtienen valores pequeños para la probabilidad de mutación (0.02 y 0.03) y valores altos para la probabilidad de cruce (0.90 y 0.95). Estos resultados describen una alta combinación e intercambio en las soluciones durante cada iteración así como una ligera aleatoriedad en las soluciones suficiente para poder mantener cierto grado de diversidad en la población. Adicionalmente, podemos notar que dichos parámetros ayudan a que el algoritmo converja y no se estanque ya que el criterio de parada alcanzado es el numero máximo de iteraciones y no el criterio de estancamiento. [21].

Sintonización Parámetros GA			
Parámetro	Pc	Pm	Mejor Resultado
Mejor Error	0.9	0.02	2.84
Menor Estancamiento	0.95	0.03	200/200

Cuadro 4.2: Sintonización Parámetros GA

4.1.1.3. Sintonización de BAT

Como se ha mencionado previamente, el algoritmo BAT comparte ciertas similitudes con el algoritmo PSO. BAT al igual que PSO poseen una componente de inercia θ y un factor de escalamiento global que permiten que en cada iteración la posición del murciélago o partícula venga dada por su velocidad. En el caso de BAT, tal como se puede observar en la ecuación 2.21, tendremos que la velocidad vendrá dada por la inercia θ, la constante global y la frecuencia fi. Dado que ambos son algoritmos inspirados en enjambres y la forma en que se determina su posición y velocidad es similar, utilizaremos los mismos resultados de

y θ obtenidos para PSO en el caso de BAT.

Por otra parte, las constantes de variación de la tasa de pulso y volumen, α y γ, serán las encargadas de ajustar el valor de la tasa de emisión de pulsos y el volumen con el que se emiten en cada iteración tal como se puede observar en las las ecuaciones 2.22, 2.23. Según lo encontrado en la literatura, tenemos que estos dos parámetros mantienen un valor constante de 0.9 los cuales ofrecen resultados satisfactorios para la mayoría de los problemas de optimización [21].

Tal como se puede observar en los resultados obtenidos en la tabla 4.3 podemos ver el buen desempeño que tiene en términos de estancamiento ya que logra hacer 92 % de las iteraciones deseadas hasta detenerse en la iteración número 184. El comportamiento del algoritmo con estos parámetros es similar al de PSO ya que al tener una inercia alta y un parámetro beta bajo, esto genera que en cada iteración la velocidad sea 0.9 veces igual a la anterior y la variación con respecto al mejor valor global sea pequeña. Con esto se tiene que en cada iteración el tamaño de paso o variación entre las soluciones serán pequeñas.

Sintonización Parámetros BAT					
Parámetro	Inercia	Gamma	Alpha	Beta	Mejor resultado
Mejor Error	0.8	0.9	0.9	0.3	4.34
Menor Estancamiento	0.8	0.9	0.9	0.3	184/200

Cuadro 4.3: Sintonización Parámetros BAT

4.1.1.4. Sintonización de SA

En el caso de SA se considera que la temperatura inicial será bastante alta. En este caso la temperatura se toma entre 5000 grados como temperatura inicial y 0 como temperatura final. La tasa de enfriamiento α tal como se puede observar en la ecuación 2.11, debe estar entre 0.7 y 0.99 para que de esta forma el proceso de enfriamiento pueda ser llevado a cabo de manera estable [21]. Analizando los resultados obtenidos en 4.4 podemos ver que una constante de enfriamiento mas cercana al limite superior (0.99) ofrece una mayor minimización del error mientras que una constante de enfriamiento ligeramente menor ofrece un menor estancamiento. A pesar de que los resultados presentados representan los mejores valores para cada criterio, no es difíciles notar que de todos los algoritmos SA es el que tiene el desempeño más limitado. Esto se debe a todos los demás son algoritmos basados en población los cuales manejan múltiples soluciones mientras que en el caso de SA se tiene una

única partícula, que dada la complejidad del problema presenta dificultades para presentar una solución aceptable [21].

Sintonización Parámetros SA		
Parámetro	Alpha	Mejor Valor
Mejor Error	0.91	5.68
Menor Estancamiento	0.76	134/200

Cuadro 4.4: Sintonización Parámetros SA

4.1.1.5. Sintonización de Cuckoo

En el caso del cuckoo hay tres parámetros a tomar en cuenta. Primero se considera α que es el parámetro de tamaño de paso, y Λ que es el exponente Levy utilizado para generar las caminatas aleatorias globales (vuelo de Levy) tal como lo indica la ecuación 2.15. En la literatura se ha encontrado que α y Λ generalmente tienen un valor de 1.5 y 0.01 respectivamente, por ende en esta investigación se mantendrán así [21]. Por ultimo la probabilidad de abandonamiento de nidos Pa es un porcentaje que varia de 0 a 1 el cual indica el porcentaje de soluciones en cada iteración que son reemplazadas por soluciones nuevas.

En el caso del CUCKOO los resultados obtenidos que se pueden observar en la tabla 4.5 muestran que una probabilidad entre 0.25 y 0.22 es la que produce menores estancamientos y una mayor minimización del error. Esto se interpreta como que aproximadamente 1 de cada 4 nidos (soluciones) serán reemplazadas por una nueva solución. Si Pa fuese un numero pequeño, no se generaría prácticamente ningún tipo de solución nueva, y poco a poco el algoritmo irá tendiendo hacia una misma solución en cada uno de sus individuos utilizados (Cuckoos). Por otra parte, cuando Pa tiene valores muy altos, las soluciones serán tan variadas y en cada iteración estarán cambiando tanto que al tener muchas soluciones diferentes el algoritmo difícilmente convergerá hacia un valor óptimo. En el caso que Pa es igual a 0.22-0.25 representa un grado en el que la mayoría de las soluciones se conservan pero sin embargo aproximadamente una cuarta parte de las soluciones es cambiada por nuevas soluciones aleatorias manteniendo así cierto nivel de diversidad en cada iteración. [21].

4.1.1.6. Validación del funcionamiento de las estrategias de hibridación con las funciones de optimización de referencia

Para validar el funcionamiento de las estrategias de hibridación se procede a utilizar nuevamente las funciones de optimización de referencia para verificar que dichas estrategias

Sintonización Parámetros CUCKOO		
Parámetro	Pa	Mejor Resultado
Mejor Error	0.25	3.21
Menor Estancamiento	0.22	165/200

Cuadro 4.5: Sintonización Parámetros CUCKOO

sean capaces de encontrar los mínimos en las funciones bidimensionales planteadas. En esta sección se observan tanto los resultados obtenidos con las estrategias de hibridación así como los resultados obtenidos por Polanco tras ejecutar los algoritmos de manera individual sobre dichas funciones. Está validación se realizó en MATLAB donde se ejecutó cada uno de los algoritmos 100 veces considerando una tolerancia de $tol = 10^{-5}$ y un numero de individuos $n = 40$ para cada uno de los algoritmos basados en población.

Tanto en la tabla **??** como en la tabla 4.7, se puede observar que se miden 3 parámetros para cada algoritmo. El primer parámetro a tomar en cuenta es el número promedio de ejecuciones que se necesita para que al menos una de las soluciones alcance el mínimo global deseado. Acompañando a este dato tenemos nuestro segundo parámetro siendo en este caso la desviación estándar. La desviación estándar nos indica el grado de dispersión que tienen las iteraciones de cada una de las diferentes soluciones con respecto al valor promedio de iteración. Por último encontramos la efectividad que refleja en cuantas de las 100 ejecuciones que realiza el algoritmo, logra que al menos uno de sus individuos o soluciones alcance el mínimo global deseado. Las funciones aparecen listadas por el siguiente orden:

1. Función Ackley
2. Función Modified Schaffer 2
3. Función Goldstein Price
4. Función Leon
5. Función Bird
6. Función Holder Table
7. Función Egg Crate

Según los resultados obtenidos reflejados en la tabla **??** podemos observar que para la funciones Ackley y Modified Schaffer todos los algoritmos, salvo SA en ambos escenarios, tienen tasas de éxito perfectas. En el caso de la función Goldstein Price podemos ver que

Fun	PSO	GA	SA	BA	CUCKOO
(1)	722 ± 64(100 %)	38 ± 2(100 %)	2017 ± 701(23 %)	200 ± 18(100 %)	319 ± 26(100 %)
(2)	377 ± 159(100 %)	244 ± 150(100 %)	1706 ± 557(0 %)	47 ± 12(100 %)	1656 ± 900(100 %)
(3)	268 ± 51(100 %)	24 ± 3(100 %)	1605 ± 398(100 %)	75 ± 21(98 %)	139 ± 17(100 %)
(4)	276 ± 231(69 %)	1258 ± 607(97 %)	1787 ± 582(8 %)	66 ± 23(100 %)	158 ± 40(100 %)
(5)	368 ± 173(88 %)	36 ± 15(100 %)	1670 ± 518(97 %)	96 ± 20(100 %)	196 ± 117(99 %)
(6)	303 ± 192(72 %)	34 ± 11(100 %)	1876 ± 556(60 %)	87 ± 21(94 %)	150 ± 29(100 %)
(7)	17 ± 14(100 %)	41 ± 41(100 %)	32443 ± 362(100 %)	7 ± 3(100 %)	44 ± 28(100 %)

Cuadro 4.6: Rendimiento: iteraciones, desviación estándar y tasa éxito

Función	MAS	RBE
(1)	540 ± 3(100 %)	239 ± 118(100 %)
(2)	652 ± 108(96 %)	297 ± 147(100 %)
(3)	526 ± 13(100 %)	123 ± 69(100 %)
(4)	725 ± 98(100 %)	273 ± 194(100 %)
(5)	535 ± 11(100 %)	154 ± 114(100 %)
(6)	853 ± 78(100 %)	433 ± 128(100 %)
(7)	522 ± 3(100 %)	130 ± 67(100 %)

Cuadro 4.7: Tabla Resumen de iteraciones, desviación estándar y tasa éxito para estrategias de hibridación.

salvo por dos agentes en la función BAT todas los algoritmos tienen tasas de éxito del 100 %. La función León al ser una función de superficie plana representa un reto mayor para los algoritmos ya que en este caso solo BAT y Cuckoo logran alcanzar el 100 % de la tasa de éxito. Para las funciones Bird y Table Holder nuevamente vemos que GA,BAT y Cuckoo tienen tasas de éxito casi perfectas mientras que PSO y SA presentan algunas dificultades. Finalmente para la función Egg Crate podemos observar que es la única para la cual todas las soluciones de todos los algoritmos logran converger a su mínimo global en todas las ejecuciones. En cuanto al numero de ejecuciones promedio necesitado por cada uno de los algoritmos podemos ver que GA y BAT son los que menos iteraciones necesitan para alcanzar el óptimo global, a diferencia de SA que para todas las funciones requiere más de 1600 iteraciones. En cuanto a la desviación estándar, podemos ver que nuevamente SA tiene la mayor dispersión en cuanto a sus números de ejecuciones mientras que de nuevo GA y BAT presentan un bajo grado de variación en cuanto a las iteraciones necesarias para alcanzar los valores óptimos.

En cuanto a los resultados obtenidos por las estrategias de hibridación reflejados en la tabla 4.7 es posible observar que salvo en 4 soluciones con la función Modified Schaffer por

parte de la estrategia MAS, la tasa de éxito con todas las funciones de referencia es del 100 %. Realizando una comparación con los resultados obtenidos por parte de los algoritmos individuales con las estrategias de hibridación, es evidente que no todos los algoritmos individuales tienen una tasa de éxito tan buena como las de las estrategias híbridas, y en el caso de los algoritmos de solución simple es bastante pobre en comparación con los resultados obtenidos por MAS y RBE. A pesar de esto, las estrategias individuales presentan una ventaja con respecto a los algoritmos MAS Y RBE y es que salvo en el caso de GA con la función León o CUCKOO con la función Modified Schaffer, estos siempre necesitan una menor cantidad de iteraciones que las estrategias híbridas.

4.1.1.7. Resultados de Ajuste de Estrategia de Hibridación MAS

A continuación se presentan los resultados obtenidos tras evaluar los procesos de migración en la estrategia de migración asíncrona por solicitud (MAS) presentada en la sección 2.2.1. Para poder determinar el parámetro del tamaño de población migrante se ejecutó nuevamente una versión del experimento general con dos usuarios estáticos en 30 y 60 grados, con la intención de observar como es la convergencia del error cuando se tienen uno, dos, o tres soluciones migrantes.

En este caso, el criterio de estancamiento es el mismo considerado en la sección 4.1.1 donde se indica que se considerará que hay una iteración de estancamiento si el error de la iteración anterior e_{k-1} es igual al error de la iteración actual e_{k-1}. Para interpretar los resultados se observó tanto el error inicial como el error final y la cantidad de mejoras que tuvo cada algoritmo sobre el error. A cada una de estas mejoras se le denomina reducción y entre más reducciones tenga el algoritmo, mejor será su capacidad de obtener mejores soluciones y evitar estancamientos.

Para este escenario que se puede observar en la figura 4.1 se obtuvo que: para 1 solución migrante, tenemos una mejora del error de aproximadamente 186 % y 5 reducciones, para el caso de 2 soluciones migrantes se observa una mejora del 458 % con 9 reducciones sobre el error y por último se tiene que para 3 soluciones migrantes la mejora del error es del 186 % y presenta 7 instancias donde este se reduce.

Por otra parte, para saber tras cuantos atascos se debe realizar el proceso de migración, se ejecutó nuevamente la estrategia MAS considerando que la migración se realizaría tras cinco, diez y quince iteraciones de Estancamiento. Según los resultados obtenidos para este caso, ilustrados en la figura 4.2 se encontró que para migraciones tras diez iteraciones el error se reduce un 240 % y tiene 4 reducciones. Por otra parte, para el caso en que ocurren migraciones tras cinco iteraciones encontramos que el error se reduce aproximadamente un

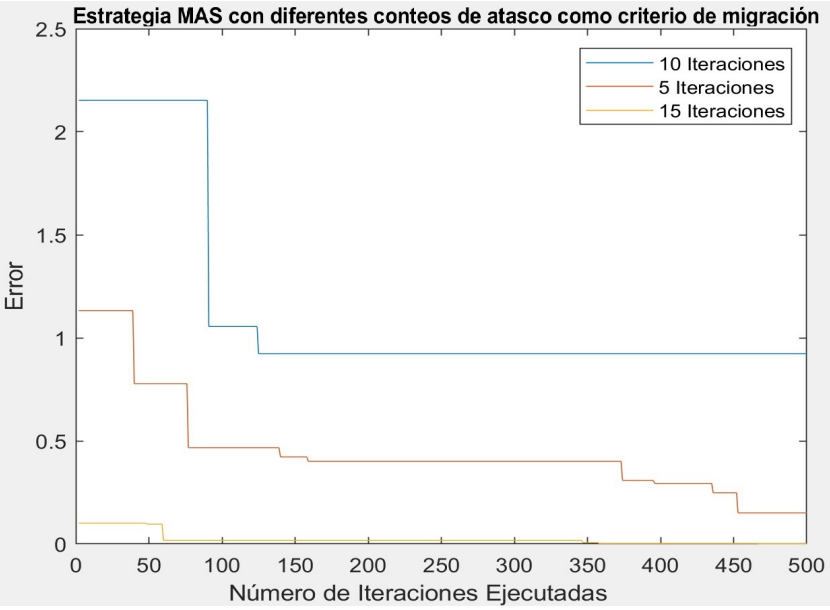


Figura 4.1: Convergencia del error usando la estrategia MAS con diferentes tamaños de grupos migratorios

793 % con´13 reducciones y por último, con 15 iteraciones se tiene una mejora del error del 785 % con 4 reducciones sobre el error.

Con estos resultados obtenidos se puede observar que la reducción del error será mejor cuando el tamaño de los grupos migrantes sea de dos individuos y la migración sea más frecuente, en este caso cada 5 iteraciones de estancamiento. Este resultado tiene sentido ya que representa un proceso más dinámico en el que si un algoritmo necesita ayuda, los demás migran sus soluciones sin dejar que hayan muchas iteraciones de por medio. Adicionalmente, demuestra que no es necesario tener grupos migratorios numerosos ya que con solo dos soluciones migrantes (para este tamaño de población) se obtienen buenos resultados.

4.1.2. Experimento usuarios fijos

4.1.2.1. Escenario 2 usuarios fijos

A continuación se presentan los resultados obtenidos para los casos en que hay dos usuarios fijos. Para la obtención de estos resultados, los usuarios fueron ubicados en 20 y 50 grados y se procedió a observar el comportamiento de diferentes algoritmos y estrategias de hibridación sobre el experimento. Se utilizó una cantidad máxima de 50 ejecuciones para cada algoritmo y como criterios de parada se consideraron bien 300 iteraciones en total o

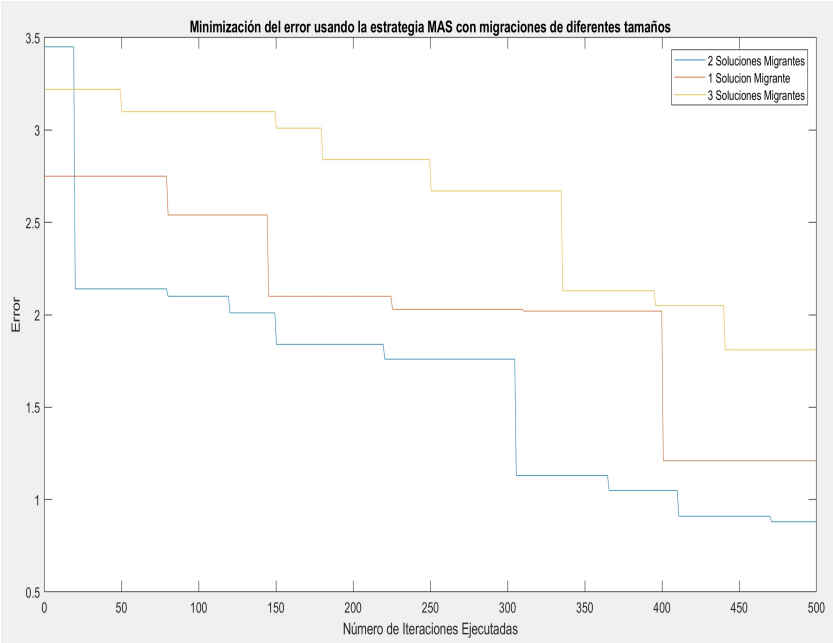


Figura 4.2: Convergencia del error usando la estrategia MAS con diferentes iteraciones de atasco como criterio migratorio

que el algoritmo se detuviera tras 50 iteraciones si $fitness3$ no mejoraba. Las imágenes y representaciones correspondientes pueden ser vistas en la sección A.1.2. Adicionalmente para cada uno de los algoritmos se obtuvo el tiempo de ejecución, el promedio y la desviación estándar de las iteraciones y la potencia promedio recibida por cada usuario respecto a la Isotrópica. Estos dos usuarios se consideraron tanto en el caso de distancias cercanas como el caso de distancias lejanas.

En la tabla 4.8 se presentan los resultados obtenidos para el caso de los dos usuarios en distancias lejanas. De acuerdo con estos resultados, se aprecia que las 50 iteraciones de GA son las que tienen un mayor tiempo de ejecución (246 segundos), seguido por la estrategia híbrida RBE con 241 y CUCKOO con 231 segundos. En cuarto y quinto lugar se encuentran PSO y MAS con 226 y 222 segundos, respectivamente. Con respecto a los algoritmos con la ejecución más rápida, BA con 208 segundos y SA con 179 lideran esta categorización. En este caso las estrategias híbridas y los algoritmos individuales tienen tiempos de ejecución bastante similares donde incluso algunos algoritmos individuales tienen tiempos de ejecución mayores a las estrategias híbridas.

En las columnas “Prom” y “Desv” se encuentra el número de iteraciones promedio de

las 50 ejecuciones de cada algoritmo o estrategia y su respectiva desviación estándar. Esta información es relevante ya que nos permite tener una idea en promedio de cuantas veces el algoritmo se aproxima al valor máximo de 300 iteraciones. Entre más cercano se encuentre el número promedio de iteraciones a 300, esto quiere decir que el algoritmo se estanca menos. En el caso del RBE y MAS se tiene que ambas tienen la mayor cantidad de iteraciones con 292 y 287 respectivamente. Ciertos algoritmos como CUCKOO, BA y SA alcanzan a hacer un número menor de iteraciones en comparación a los demás. Esto significa que los algoritmos obtienen la mayoría de sus mejoras al $fitness3$ en sus primeras iteraciones y luego el algoritmo se mantiene igual, o simplemente obtienen muy pocas mejoras en el proceso de convergencia.

	T(s)	Prom	Desv	Promedio Potencia (dB)			Desviación Potencia (dB)	
				U_1	U_2	Prom Por Usuario	U_1	U_2
PSO	226	192	29	4.02	2.07	3.04	2.67	2.76
GA	246	137	51	3.38	3.05	3.21	1.76	3.15
SA	179	117	7	5.25	0.57	2.91	3.49	6.05
BA	208	108	30	5.09	1.04	3.02	3.26	4.17
CUCKOO	231	104	8	5.11	0.57	2.84	3.01	5.87
MAS	222	292	10	3.60	3.44	3.52	0.77	0.55
RBE	241	287	20	3.23	3.25	3.24	0.55	0.64

Cuadro 4.8: Tabla resumen de experimento de usuarios 2 Fijos ubicados a distancias lejanas

Las columnas restantes de la tabla 4.8 muestran la potencia promedio respecto a la isotrópica y su desviación estándar en decibeles. El algoritmo que mas suministró energía en dirección de los usuarios fue la estrategia MAS con 3.52 dB, seguido de la estrategia RBE con 3.24 dB y de GA con 3.21 dB de potencia promedio. Más allá de que la potencia promedio recibida por usuarios sea similar, si se observa la potencia que reciben los usuarios U1 y U2 es posible notar que los algoritmos individuales irradian más potencia hacia uno de los dos usuarios, mientras que las estrategias híbridas distribuyen la potencia de la misma manera a ambos usuarios.

Considerando este mismo escenario pero para los usuarios que se encuentran a distancias cercanas, los resultados se muestran en la tabla 4.9. En esta tabla se aprecia que las estrategias híbridas RBE y MAS son las que presentan los tiempos de ejecución más altos con 224 y 213 segundos, respectivamente. En este escenario, las estrategias de hibridación tienen tiempos de ejecución mayores a los algoritmos individuales. En cuanto a la potencia, MAS emite en promedio 3.25 dB a cada usuario, siendo este el que mayor potencia irradia, seguido por RBE que irradia 3.15 dB y BAT que transmite una potencia promedio de 3.00

dB a ambos usuarios. En este escenario se evidencia también la tendencia que presentan los algoritmos individuales, donde estos tienden a concentrar su potencia más hacia un usuario que al otro, a diferencia de las estrategias híbridas que irradian niveles de potencia similares a ambos. Adicionalmente, es posible observar que en el caso de SA la potencia que recibe U2 es negativa. Estos valores negativos en los algoritmos no significan que en dicho punto el algoritmo consume potencia en vez de irradiarla, sino que se tienen niveles muy bajos de potencia (existencia de un nulo) o existe un lóbulo en dicho punto cuya potencia es menor que el valor de la isotrópica. Finalmente, si se realiza una comparación entre el escenario cercano y lejano se puede ver que el comportamiento entre los algoritmos individuales y estrategias de hibridación es bastante similar. La principal diferencia que se observa es que la potencia recibida por usuarios que se encuentran lejos es mayor que la de los usuarios cercanos. Esto se debe a que para distancias lejanas los algoritmos deben conformar haces más directivos dado que la función gaussiana de referencia es más estrecha.

	T(s)	Prom	Desv	Promedio Potencia (dB)			Desviación Potencia (dB)	
				U_1	U_2	Prom por Usuario	U_1	U_2
PSO	205	136	27	5.13	0.80	2.96	1.47	2.99
GA	206	141	50	4.54	1.28	2.91	2.31	3.53
SA	154	109	16	1.49	-7.17	-2.84	6.40	4.31
BA	207	119	35	5.84	0.17	3.00	1.96	3.51
CUCKOO	210	196	15	4.53	1.13	2.83	1.71	2.79
MAS	213	294	17	3.36	3.14	3.25	0.87	2.05
RBE	224	169	27	3.23	3.15	3.19	1.52	0.41

Cuadro 4.9: Tabla resumen de experimento de 2 usuarios Fijos ubicados a distancias cercanas

4.1.2.2. Escenario 3 usuarios fijos

A continuación se presentan los resultados obtenidos para los casos en que existen tres usuarios fijos. En esta escenario, los usuarios se ubicaron en 20, 50 y 80 grados, respectivamente, y se procedió a observar el comportamiento de los diferentes algoritmos y estrategias híbridas. Se utilizó una cantidad máxima de 50 ejecuciones para cada algoritmo y como criterios de parada se consideraron 300 iteraciones en total o que el algoritmo se detuviera tras 50 iteraciones en el que $fitness3$ no mejorara. Las imágenes y representaciones correspondientes a esta sección se pueden observar en la sección A.1.2. Para cada uno de los algoritmos se obtuvo el tiempo de ejecución, el número promedio y la desviación estándar de las iteraciones y la potencia promedio recibida por cada usuario con respecto a la Isotrópica.

Los resultados de este experimento se muestran en la tabla 4.10. De acuerdo con la información de la tabla, se obtiene que las 50 iteraciones de RBE son las que tienen un mayor tiempo de ejecución (315 segundos), seguido por el algoritmo CUCKOO con 261 y la estrategia MAS con 245 segundos. En cuarto y quinto lugar se encuentran PSO y BAT con 215 y 207 segundos, respectivamente. Por último, los algoritmos con la ejecución más rápida son GA con 207 y SA con 116 segundos. A pesar de tener tiempos de ejecución en rangos similares, las estrategias híbridas requieren un poco más de tiempo que los algoritmos individuales. Este tiempo de ejecución más elevado normalmente está relacionado con el hecho de que los algoritmos híbridos son algoritmos con mayor diversidad y entre mayor sea la diversidad de un algoritmo mayor será su tiempo de ejecución. Con respecto a los promedios, RBE y MAS fueron las estrategias o algoritmos con mas iteraciones promedio (24 y 29 de desviación estándar, respectivamente). Adicionalmente, también se observa que algoritmos como PSO, BAT y SA alcanzan a hacer un número menor de iteraciones en comparación con el resto. Esto puede significar que los algoritmos obtienen la mayoría de sus mejoras al $fitness3$ en sus primeras iteraciones y luego el algoritmo se mantiene igual, o simplemente obtienen muy pocas mejoras en el proceso de convergencia. El algoritmo que en promedio suministró más energía en dirección de los usuarios fue RBE con un total de energía de 2.31 dB con respecto a la isotrópica, seguido por MAS con 1.93 dB y GA con 1.49 dB. Según lo que se puede observar en la tabla 4.10 todos los algoritmos individuales presentan dificultades para conformar o dirigir un lóbulo donde se encuentra el usuario U3 ya que incluso algunos tienen valores de potencia por debajo de la isotrópica. En el caso de las estrategias de hibridación este problema no sucede; sin embargo, el tercer usuario recibe menos potencia que los primeros dos. Esto demuestra que al agregar un usuario adicional la complejidad del problema aumenta ya que tanto los algoritmos como las estrategias no pueden conformar lóbulos con la misma facilidad que en el escenario de dos usuarios.

	T(s)	Prom	Desv	Promedio Potencia (dB)				Desviación Potencia (dB)		
				U_1	U_2	U_3	Prom por Usuarios	U_1	U_2	U_3
PSO	215	81	25	6.21	-0.02	-2.46	1.24	3.94	4.10	4.80
GA	207	139	57	6.29	-0.89	-0.92	1.49	3.59	4.70	3.01
SA	116	149	43	5.74	0.77	-2.07	1.48	2.83	2.32	1.71
BAT	207	89	21	4.24	2.08	-3.49	0.94	6.80	3.66	5.39
CUCKOO	261	80	14	6.08	0.08	-4.21	0.65	3.58	4.98	7.55
MAS	245	263	29	2.90	1.84	1.05	1.93	1.74	0.91	0.76
RBE	315	274	24	2.80	2.65	1.49	2.31	0.99	0.51	0.71

Cuadro 4.10: Tabla resumen de experimento de 3 usuarios estáticos ubicados a distancias lejanas

Por otra parte, los resultados considerando el mismo escenario pero con usuarios ubicados a distancias cercanas se presentan en la tabla 4.11. De acuerdo con la información reportada en la tabla, el algoritmo con mayor tiempo de Ejecución es RBE con 323 segundos de ejecución, seguido por MAS con 277 segundos. Nuevamente, los resultados indican que las estrategias híbridas requieren más tiempo de ejecución. En cuanto a la potencia recibida se aprecia que MAS entrega una potencia promedio de 1.74 dB seguido de RBE que entrega 1.64 dB y GA con 1.49 dB. Similarmente, para distancias cercanas se repite la tendencia observada en el caso de distancias lejanas donde los algoritmos individuales presentan dificultades para transmitir la potencia en dirección del usuario U3. Adicionalmente, se observa que al igual que en el caso de dos usuarios estáticos, en este escenario los usuarios ubicados a distancias lejanas reciben mayor potencia en promedio que los ubicados a distancias cercanas.

	T(s)	Prom	Desv	Promedio Potencia (dB)				Desviación Potencia (dB)		
				U_1	U_2	U_3	Prom por usuario	U_1	U_2	U_3
PSO	222	91	32	6.21	-0.02	-2.71	1.16	3.63	3.12	3.11
GA	212	146	50	5.87	-0.49	-0.92	1.49	1.59	5.70	1.21
SA	139	147	47	5.44	1.33	-2.44	1.44	3.97	4.22	4.79
BAT	228	99	31	3.45	2.14	-3.01	0.86	2.81	2.66	4.39
CUCKOO	269	84	19	4.71	0.39	-4.57	0.53	2.48	4.08	6.23
MAS	277	269	19	2.90	2.84	1.05	1.74	1.88	0.91	0.76
RBE	323	271	27	2.80	1.65	0.49	1.64	0.76	0.75	0.66

Cuadro 4.11: Tabla resumen de experimento de 3 usuarios estáticos ubicados a distancias cercanas

4.1.3. Experimento usuarios Nómadas

Para el caso de usuarios nómadas, tal como se presentó en secciones previas, se consideraron 2 usuarios con posiciones variables. Primeramente, se ubican los dos usuarios una posición inicial de 20 y 50 grados, respectivamente. Posteriormente, luego de que el algoritmo converge y se detiene, ambos usuarios migran a las ubicaciones de 30 y 80 grados, respectivamente. Para cada uno de los algoritmos se obtuvo el tiempo de ejecución, el promedio y la desviación estándar de las iteraciones y la potencia promedio recibida por cada usuario con respecto a la Isotrópica.

Las imágenes y representaciones correspondientes se muestran en la sección A.1.2 y los resultados en la tabla 4.12. De acuerdo con los resultados obtenidos, es posible observar que

para el caso de distancias lejanas las 50 iteraciones de RBE son las que tienen un mayor tiempo de ejecución (288 segundos), seguido por la estrategia MAS con 223 y la estrategia CUCKOO con 213 segundos. De igual manera, se puede observar que las estrategias híbridas requieren un poco más de tiempo que los algoritmos individuales. RBE y MAS fueron las estrategias o algoritmos con más iteraciones promedio (261 Y 242) junto con unas desviaciones estándar de 41 y 24, respectivamente. Todos los demás algoritmos presentan menores ejecuciones que estos escenarios lo cual significa que RBE y MAS tienen una mayor capacidad para obtener mejores valores de $fitness3$ dado que no se estancan con tanta facilidad como los algoritmos individuales. Por otra parte, MAS con 4.83 dB promedio con respecto a la isotrópica, fue el algoritmo que más suministró energía en dirección de los usuarios. Le siguen en orden RBE con 3.93 dB y GA con 3.41 dB. Según lo que se puede observar en la tabla 4.12 nuevamente se aprecia que los algoritmos individuales conforman la mayor parte de la potencia hacia un usuario mientras que las estrategias de hibridación presentan la cualidad de irradiar potencia de manera equitativa a cada uno de los usuarios individuales. Adicionalmente, se destaca en este experimento la capacidad de tanto las estrategias de hibridación como de los algoritmos individuales de irradiar la potencia a los usuarios una vez estos se estos se desplazan. Esto demuestra que tanto los algoritmos como las estrategias de hibridación son capaces de identificar las nuevas ubicaciones de los usuarios, conformar haces y dirigirlos hacia las nuevas posiciones deseadas.

	T(s)	Prom	Desv	Promedio Potencia (dB)			Desviación Potencia (dB)	
				U_1	U_2	Prom por usuario	U_1	U_2
PSO	206	106	42	4.75	1.97	3.36	2.11	2.06
GA	208	132	14	4.68	2.13	3.41	1.17	2.84
SA	155	79	3	3.11	2.87	2.99	6.56	4.46
BA	209	88	25	6.01	0.14	3.08	2.61	2.48
CUCKOO	213	71	4	6.11	3.13	4.62	2.14	2.72
MAS	223	242	24	4.72	4.93	4.83	1.07	0.94
RBE	288	261	30	3.93	3.94	3.93	1.2	1.31

Cuadro 4.12: Tabla resumen de experimento de usuarios nómadas ubicados a distancias lejanas

Por otra parte, se procedió a considerar el mismo escenario pero con los usuarios ubicados a distancias cercanas. Los resultados de este experimento se observan que en la tabla 4.13. De acuerdo con estos, los algoritmos con mayor tiempo de ejecución son MAS y RBE con 247 y 236 segundos de ejecución; todos los demás algoritmos nuevamente tienen tiempos de ejecución menores a estos. Adicionalmente, MAS, y RBE son los que mayor numero de itera-

ciones promedio, 246 y 202, respectivamente. Con esto se observa que las estrategias híbridas son aquellas que más tiempo requieren, pero al mismo tiempo son las que más iteraciones ejecutan. Por último, con relación a la potencia promedio, MAS con 4.54 dB, RBE con 3.45 dB y GA con 3.29 dB son los que más potencia promedio entregan. Al igual que en escenarios y experimentos anteriores se evidencia nuevamente que los algoritmos individuales tanto para distancias lejanas como para distancias cercanas presentan dificultades para transmitir potencia de manera uniforme a ambos usuarios.

	T(s)	Prom	Desv	Promedio Potencia (dB)			Desviación Potencia (dB)	
				U_1	U_2	Prom por usuario	U_1	U_2
PSO	211	106	42	4.15	1.45	2.8	3.20	2.76
GA	212	132	14	4.43	2.14	3.29	0.91	2.73
SA	135	79	3	3.41	2.31	2.86	9.20	6.6
BA	214	88	25	4.51	1.11	2.91	3.60	3.18
CUCKOO	213	75	8	5.11	1.13	3.12	1.51	2.50
MAS	247	246	22	4.16	4.54	4.35	1.77	1.43
RBE	236	202	45	3.41	3.49	3.45	1.42	1.17

Cuadro 4.13: Tabla resumen de experimento de usuarios nómadas ubicados a distancias cercanas

4.1.4. Experimento usuarios Mixtos

Para el caso de usuarios mixtos se consideraron 3 usuarios de los cuales uno siempre estará fijo en la ubicación de 160 grados mientras que los otros migraran de 35 y 60 grados respectivamente a 45 y 80 grados. En este experimento se registra nuevamente el comportamiento de los algoritmos después de realizar el desplazamiento. Las imágenes y representaciones correspondientes a esta sección se pueden observar A.1.2 .

Para cada uno de los algoritmos se obtuvo nuevamente el tiempo de ejecución, el número de iteraciones promedio y desviación estándar de dichas iteraciones y la potencia promedio recibida por cada usuario con respecto a la Isotrópica. A continuación se puede observar los resultados obtenidos para el caso de usuarios a distancias lejanas.

En la tabla 4.14 obtenemos que las 50 iteraciones de RBE son las que tienen un mayor tiempo de ejecución siendo este de 292 segundos,seguido por la estrategia MAS con 210 segundos en tercer lugar BAT Y GA con 209 segundos. En este caso nuevamente las estrategias

híbridas RBE y MAS presentan un mayor tiempo de ejecución que los algoritmos individuales. En el caso de MAS y RBE se tiene que fueron las estrategias o algoritmos con más iteraciones promedio (269 y 259) junto con unas desviaciones de 24 y 35 respectivamente. Todos los demás algoritmos presentan menores ejecuciones que estos escenarios lo cual significan que RBE y MAS tienen una mayor capacidad para obtener mejores valores de $fitness3$.

Según lo que se puede observar en la tabla 4.14, el algoritmo que más suministró energía en dirección de los usuarios en promedio fue RBE con 2.53 dB promedio con respecto a la isotrópica, seguido de MAS con 1.60 dB y en tercer lugar CUCKOO con 1.11 dB. Al igual que en el escenario de 3 usuarios estáticos, se observa que los algoritmos individuales presentan dificultades para poder irradiar la potencia en cada una de las direcciones deseadas. Los algoritmos individuales se enfocan en únicamente dos usuarios ya que siempre el usuario U2 presenta una potencia promedio negativa. A diferencia de los algoritmos individuales, las estrategias de hibridación si son capaces de irradiar potencia hacia cada uno de los usuarios, sin embargo en este caso no siempre es posible que cada usuario reciba la misma potencia. Esto se debe a la complejidad que presenta el problema tanto para los algoritmos individuales como para las estrategias híbridas al momento de considerar tres usuarios.

Tanto las estrategias de hibridación como cada uno de los algoritmos individuales poseen la capacidad de irradiar la potencia a los usuarios una vez estos se hayan desplazado y adicionalmente es posible observar que la potencia recibida normalmente es mayor en el usuario estático (U3). Esto se debe a que como es el único usuario que mantiene la misma ubicación y por ende el algoritmo dedica más tiempo y más iteraciones a este y es capaz de dirigirle más potencia.

	T(s)	Prom	Desv	Promedio Potencia (dB)				Desviación Potencia (dB)		
				U_1	U_2	U_3	Prom por usuario	U_1	U_2	U_3
PSO	219	87	22	3.33	-9.82	6.01	-0.16	2.04	9.84	2.74
GA	209	139	48	3.03	-9.34	7.37	0.35	3.87	6.78	1.74
SA	155	153	21	-0.72	-2.07	5.01	1.22	6.46	4.57	2.98
BAT	209	87	34	2.62	-4.82	6.1	1.30	4.48	4.76	1.78
CUCKOO	215	89	18	1.82	-5.21	6.72	1.11	4.74	6.62	1.35
MAS	210	269	24	0.85	1.39	2.57	1.60	1.01	0.87	1.27
RBE	292	259	35	2.55	2.54	2.51	2.53	0.74	0.72	0.77

Cuadro 4.14: Tabla resumen de experimento de usuarios mixtos ubicados a distancias lejanas

Para este mismo escenario pero considerando que los usuarios se encuentran a distancias

cercanas podemos observar en la tabla 4.15 que nuevamente las estrategias híbridas son las que mayor tiempo de ejecución y numero de ejecuciones tienen. El algoritmo con mayor tiempo de ejecución es RBE con 298 segundos de ejecución seguido de MAS con 241 segundos. En cuanto al número de iteraciones promedio tenemos que MAS, y RBE son los que mayor numero de iteraciones promedio tienen con 291 y 274 respectivamente.

En cuanto a la potencia podemos observar que RBE entrega una potencia promedio de 2.40 dB seguido de MAS que entrega 1.15 dB y en tercer lugar encontramos a CUCKOO con 1.00 dB. Con esto se puede observar un comportamiento similar al del escenario de distancias lejanas ya que los algoritmos individuales nuevamente no son capaces de irradiar la potencia a cada uno de los usuarios y el usuario U3 sigue siendo el usuario que más potencia recibe. Adicionalmente al igual que en los escenarios con otros tipos de usuarios se evidencia que la potencia recibida para usuarios que se encuentran a distancias lejanas es mayor que para distancias cercanas.

	T(s)	Prom	Desv	Promedio Potencia (dB)				Desviación Potencia (dB)		
				U_1	U_2	U_3	Prom por usuario	U_1	U_2	U_3
PSO	203	94	30	1.53	-8.62	6.26	-0.27	6.4	5.4	3.4
GA	211	107	35	4.29	-12.24	2.30	-1.88	4.90	13.70	2.72
SA	121	168	12	-1.82	-8.44	3.01	-2.41	5.92	6.03	4.92
BAT	225	90	28	0.62	-10.48	6.60	-1.09	6.80	5.46	2.94
CUCKOO	215	89	18	0.82	-5.53	7.72	1.00	4.74	6.62	1.35
MAS	241	291	14	1.57	0.95	2.15	1.55	1.48	0.82	2.62
RBE	298	274	17	2.58	2.35	2.29	2.40	1.04	0.92	2.02

Cuadro 4.15: Tabla resumen de experimento de usuarios mixtos ubicados a distancias cercanas

4.1.5. Simetría

Según hemos podido observar, todos los diagramas de radiación presentados son representados de manera simétrica. Los diagramas reflejan la misma potencia de 90 a 269 grados que de 270 grados a 89. Esta simetría está producida por la manera en que se define el factor de arreglo. Nuevamente observando la función de arreglo 4.1 podemos ver que está definida en términos de sumatorias de exponenciales complejas donde W_{mn} es el vector de pesos complejos, θ es el ángulo azimutal, ϕ es el ángulo vertical, k es el numero de onda y d_x la distancia entre cada elemento sobre el eje x y d_y la distancia entre cada elemento sobre el eje y

$$AF = \sum_{m=1}^{M} \sum_{n=1}^{N} W_{mn} e^{j[(m-1)(kd_x \sin\theta\cos\phi)+(n-1)(kd_y \sin\theta\sin\phi)]} \tag{4.1}$$

Como bien se ha mencionado previamente, para este experimento el ángulo vertical ϕ será igual a cero. Al observar la ecuación 4.1 es posible notar que con esto para este valor de ϕ se obtiene que $\sin\phi = 0$ y $\cos\phi = 1$ y por lo tanto dicha expresión quedará reducida a:

$$AF = \sum_{m=1}^{M} \sum_{n=1}^{N} W_{mn} e^{j[(m-1)(kd_x \sin\theta)]} \tag{4.2}$$

con esto tendremos que k, d_y,y d_x son valores constantes mientras que $\sin\theta$ sera un vector resultante de evaluar θ en la función seno. Dado que en esta investigación *theta* está definido de $[0, \pi]$, es importante recordar que el seno de theta tiene el comportamiento de una función par en este rango ya que inicia en cero, alcanza su valor máximo en $\pi/2$ y luego tiene un valor de cero nuevamente en π. Si observamos nuevamente la ecuación 4.2 podemos ver que el factor de arreglo vendrá dado por una doble sumatoria de cada uno de los pesos complejos multiplicados por una exponencial compleja evaluada sobre un vector simétrico. En este caso tendremos que un vector simétrico es todo aquel vector $a = [a_1, a_2...a_k]$ de k elementos donde se cumple que $a_1 = a_k$, $a_2 = a_{k-1}$ y así sucesivamente cumpliendo siempre que $a_i = a_{k+1-i}$. Al tener una exponencial compleja evaluada en un vector simétrico, el resultado de esta también será un vector simétrico complejo. Finalmente cada iteración será una sumatoria de un peso complejo de forma $w_{nm} = a + jb$ multiplicado por un vector simétrico complejo, cuyo producto será un nuevamente un vector simétrico complejo. Al momento de graficar dicho vector, obtenemos la simetría que hemos podido notar en las diferentes representaciones del diagrama de radiación.

4.1.6. Validación de resultados Obtenidos

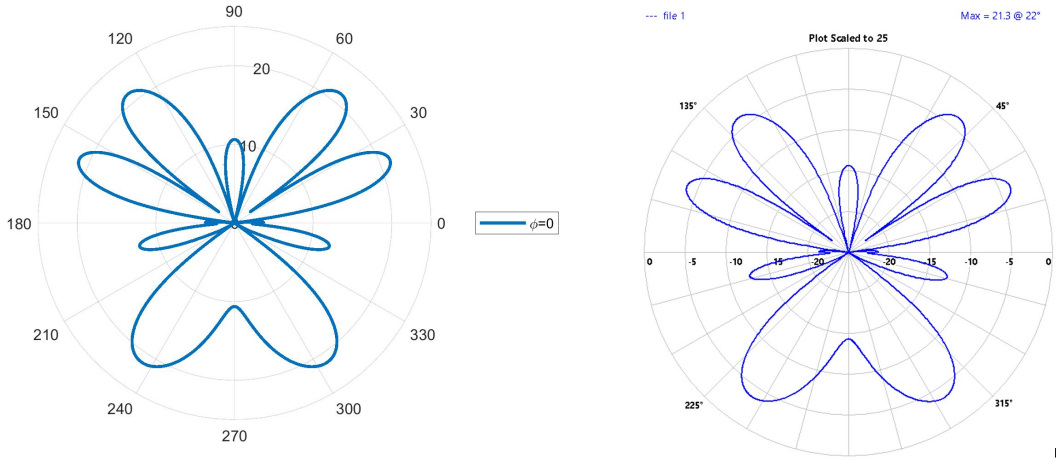
Más allá de que en esta investigación se ha demostrado cual ha sido el proceso para llegar a los resultados obtenidos, es importante ver si los resultados tienen alguna validez en un contexto o entorno ajeno o diferente al construido en esta investigación. Para poder realizar dicha validación se hace uso de la herramienta Antenna Radiation Diagram Plotter.

Antenna Radiation Diagram Plotter es una herramienta de software libre desarrollada por Roberto Padovani (Ex vicepresidente ejecutivo de Qualcomm) que permite graficar los diagramas de radiación de antenas. El desarrollo de esta herramienta surge como una respuesta de Padovani a la falta de herramientas practicas para graficar un diagrama de radiación de manera sencilla y directa. El software permite representar diagramas de radiación tanto

normalizados como sin normalizar haciendo uso de vectores que determinen el valor del diagrama en cada ángulo de 0 a 360 grados [46].

En el caso de esta investigación lo que se hizo para realizar la validación fue tomar los vectores de pesos complejos obtenidos de las estrategias de hibridación, evaluarlas en la función de arreglo utilizada y posteriormente expresar esos resultados en decibeles. Dichos resultados son graficados utilizando el software previamente mencionado. A continuación se presentan un par de ejemplos tanto para dos usuarios fijos ubicados en 20 y 50 grados como para el caso en que se tiene dos usuarios nómadas que inicialmente están ubicados en 20 y 50 grados y posteriormente migran a 30 y 80 grados respectivamente. Ambos ejemplos muestran el diagrama de radiación obtenido tanto por Matlab como el obtenido al evaluar los resultados en el Antenna Radiation Diagram Plotter.

La figura 4.3 ilustra la representación sobre el plano polar de el diagrama de radiación sobre el eje theta con corte en $\phi = 0$ correspóndete a dos usuarios fijos en 20 y 50 grados. Como podemos observar en ambos escenarios se conforman los haces en las direcciones deseadas y la representación es la misma para el caso de MATLAB como para el software Antenna Radiation Diagram Plotter.

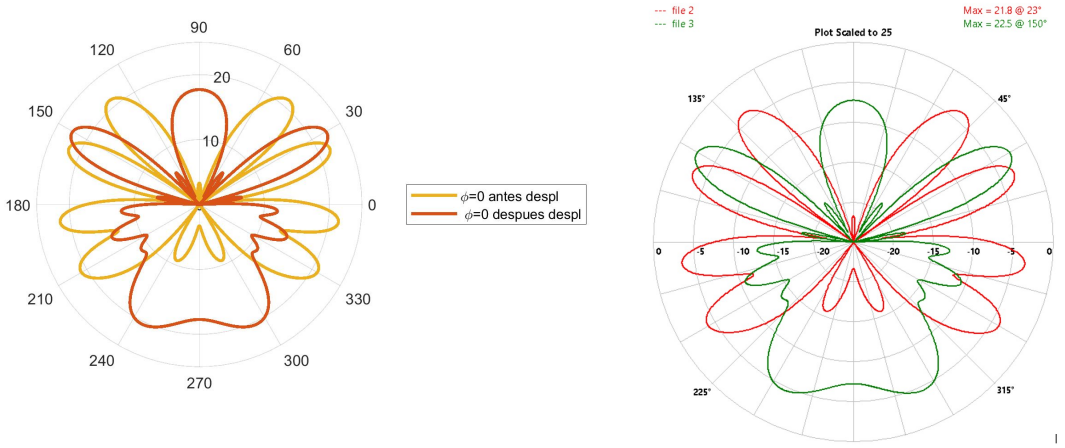


(a) MATLAB (b) Antenna Radiation Diagram Plot

Figura 4.3: Diagrama de Radiación 2 usuarios Fijos θ usando la estrategia MAS

La figura 4.4 ilustra la representación sobre el plano polar de el diagrama de radiación sobre el eje theta con corte en $\phi = 0$ correspondiente a dos usuarios nómadas. En este plano polar se ilustran dos diagramas de radiación el primero correspondiente a la posición inicial de los dos usuarios en 20 y 50 grados y la segunda a la posición obtenida después del

desplazamiento de 30 y 80 grados. En el caso de matlab se indica que la grafica color amarillo corresponde al eje theta antes del desplazamiento y el diagrama color naranja después del desplazamiento. Para el software Antenna Radiation Diagram Plotter el diagrama color rojo representa la potencia irradiada en las posiciones de 20 y 50 mientras que el verde ilustra el diagrama para los casos de 30 y 80 grados. En ambos escenarios nuevamente se conforman los haces en las direcciones deseadas y la representación es la misma para el caso de MATLAB como para el software Antenna Radiation Diagram Plotter.



(a) MATLAB (b) Antenna Radiation Diagram Plot

Figura 4.4: Diagrama de Radiación 2 usuarios Nómadas θ utilizando la estrategia MAS

Capítulo 5

Conclusiones

Esta investigación tiene por objetivo aplicar los algoritmos metaheurísticos desarrollados por Polanco (2021) [24] al problema de conformación de haces de arreglos de antenas MIMO masivo con el propósito de generar un algoritmo que presente un mejor rendimiento. La diferencia fundamental con respecto al trabajo de Polanco (2021) es que en este trabajo se usan algoritmos metaheuristicos híbridos que han sido combinados mediante diferentes estrategias de hibridación. Para llevar a cabo esta hibridación se procedió a hacer una revisión del estado del arte de las diferentes estrategias, las clasificaciones existentes y las cualidades que deben tener los algoritmos para que puedan ser combinables. Finalmente, la tarea de realizar la hibridación se llevó a cabo tomando en cuenta que, más allá de que existan infinitas posibilidades de combinar algoritmos metaheurísticos, no todas son lógicas o deben realizarse. En ese sentido, al momento de combinar cada algoritmo se deben considerar sus cualidades, funcionamiento y, en especial, fortalezas de manera que se pueda generar un nuevo algoritmo más eficaz y más eficiente.

Usando de la taxonomía de Talbi se encontró que se podían aprovechar las cualidades exploradoras de PSO, GA, CUCKOO, y BAT para poder examinar el problema de una manera mas extensa, dado que son algoritmos basados en población. Sin embargo, el funcionamiento de cada algoritmo es diferente. La manera en que exploran la superficie del problema es diferente y por ende es beneficioso tener varias poblaciones realizando procesos de búsqueda de manera distinta. Por otra parte, gracias a las cualidades de búsqueda individual que presentan SA y, su variación, CSA, estas ofrecen dos alternativas cuyas cualidades de búsqueda global son bastante limitadas, pero presentan una capacidad de explotación eficiente en valores cercanos al óptimo. Gracias a esta característica de SA y CSA, ambos algoritmos se utilizaron como etapas de refinamiento de resultados dentro de las estrategias de hibridación planteadas. Tomando esto en cuenta, se presentaron dos estrategias de hibridación: Migración asíncrona por solicitud (MAS) y relevo y búsqueda en equipo (RBE).

MAS se enfoca en utilizar diferentes algoritmos basados en población para realizar una búsqueda de la mejor solución donde los algoritmos son capaces de migrar sus soluciones y trabajar entre si. Por su parte, RBE utiliza varios algoritmos basados en población de forma paralela y posteriormente utiliza dos etapas de refinamiento, una determinista y la

otra metaheurística para obtener la mejor solución.

Las estrategias de hibridizacion propuestas se utilizaron en diversos contextos y escenarios para evaluar su desempeño. En el caso de usuarios estáticos, se encontró que las estrategias de hibridación y algoritmos individuales tienen un comportamiento similar cuando se consideran 2 usuarios. Sin embargo, al momento de considerar un usuario adicional las estrategias híbridas son capaces de dirigir la potencia hacia los tres usuarios mientras que los algoritmos individuales solo pueden dirigirla a dos de esos tres. En el caso de usuarios nómadas y mixtos, también se observa la misma tendencia que en el escenario anterior con respecto a la cantidad de usuarios y tipos de algoritmos. En el caso de los escenarios de usuarios mixtos, el usuario que recibe la mayor potencia siempre es el usuario que permanece estático.

De manera adicional, se pudo observar la capacidad que tienen las estrategias híbridas de dirigir cantidades similares de potencia hacia todos los usuarios. Los algoritmos individuales por su parte, suelen dirigir la potencia de manera más dispareja, donde unos usuarios reciben considerablemente más potencia que otros. Un último hecho a destacar es la baja desviación estándar que presentan las estrategias de hibridación en su potencia promedio en comparación con los algoritmos individuales, haciéndolos así menos variables con respecto a su valor medio.

La existencia del punto de equilibrio está nuevamente presente en estos resultados. Como bien se ha mencionado antes, todos los algoritmos sufren de dos problemas, convergencia prematura y convergencia tardía. Dichos problemas se relacionan directamente en este caso con la precisión de los algoritmos y su tiempo de ejecución. Adicionalmente, la diversidad de las soluciones también juega un rol importante. Como se pudo apreciar en los resultados, las estrategias de hibridación obtienen mejores resultados que los algoritmos ejecutados de manera individual y a su vez es notable que en algunos casos sus tiempos de ejecución son ligeramente mayores.

Estos resultados se relacionan directamente y concuerdan con lo planteado ya que tanto MAS como RBE son algoritmos con mayor diversidad que los demás algoritmos individuales. Al momento de diseñarse, se tomó en cuenta que estos algoritmos pudiesen ser lo suficientemente diversos para plantear una búsqueda más amplia pero con la intención de que su tiempo de convergencia fuese similar al de los algoritmos individuales. Su diversidad proviene del hecho de que el algoritmo MAS tiene 4 algoritmos con características exploradoras, cada uno con su propia población, que interactúan y migran entre si generando que las posibles soluciones que se consideren sean mas variadas. Por otra parte, RBE también goza de la ventaja de tener cuatro algoritmos basados en población trabajando de manera paralela y,

aunque en este escenario no interactúan directamente entre si, igual son cuatro algoritmos que intentan encontrar la mejor solución al problema de forma diferente.

Tal como se ha podido observar en los resultados obtenidos, los algoritmos metaheurísticos son capaces de dirigir mayor potencia de manera equitativa en las direcciones deseadas que los algoritmos individuales. Este rendimiento corresponde a que las estrategias híbridas, por la forma en que están diseñadas, son capaces de generar soluciones más diversas que los algoritmos individuales. La diversidad de las soluciones se relaciona directamente con la precisión de la solución así como su tiempo de convergencia. Entre más alta sea la diversidad, mayor es el tiempo de convergencia pero al mismo tiempo los resultados obtenidos son más cercanos a los deseados. Este es uno de los factores que se debe tener en cuanta al momento de elegir trabajar con los algoritmos metaheurísticos híbridos propuestos. Las estrategias híbridas planteadas en esta investigación son capaces de brindar mejores resultados sobre el problema de conformación de haces en sistemas 8x8 mimo masivo, a cambio de un numero de iteraciones mas elevado y un mayor tiempo de Ejecución. Si comparamos los resultados obtenidos por las estrategias híbridas con los algoritmos individuales es posible observar que la relación entre tiempo de ejecución y número de iteraciones es ligeramente más baja que en los algoritmos híbridos, sin embargo, el desempeño en términos de potencia dirigida a cada usuario y distribución de la potencia es notablemente mejor para el caso de las estrategias híbridas. Con esto se busca dar respuesta a la hipótesis originalmente planteada de que los algoritmos metaheurísticos híbridos tendrán un mejor desempeño que los algoritmos evaluados de manera individual. Mas allá de que los resultados lo demuestran, es importante establecer el contexto en que se justificará dicha respuesta, dado siempre existirá un intercambio entre tiempo de convergencia y precisión. En el caso de esta investigación se encontró que si se sacrifica ligeramente el tiempo de ejecución, para utilizar estrategias de hibridación en vez de ejecutar algoritmos individuales, se obtendrán resultados mejores. Para esta investigación es un costo razonable, sin embargo en otro tipo de proyectos puede no serlo.

Finalmente, con esto podemos ver que efectivamente los algoritmos metaheurísticos híbridos se presentan como una solución posible al problema de conformación de haces descrito para este tipo de antenas y queda abierto como un tema tentativo que podrá seguir siendo desarrollado e implementado.

Capítulo 6

Trabajo futuro

Como toda investigación en el mundo de la ciencia e ingeniería, este trabajo tiene aspectos en los que se puede mejorar y nuevas ideas que se pueden aplicar a el para continuar su desarrollo. En primer lugar, tenemos la consideración de algoritmos metaheurísticos diferentes a los que ya han sido utilizados. La existencia de un gran numero de algoritmos metaheurísticos nos permite pensar que puedan existir algunos que presenten características diferentes y que posiblemente ayuden a mejorar los tiempos de convergencia o calidad de la solución. Adicionalmente el uso de nuevos algoritmos también implica la posibilidad de utilizar nuevas estrategias de hibridizacion y por ende obtener aun mejores resultados.

Modificaciones sobre la función de costo utilizada actualmente ya que una función de costo que mejor se adapte o mejor describa el problema de conformación de haces puede obtener mejores resultados.

En caso de que se lograse mantener los resultados obtenidos por el algoritmo híbrido y mejorar su tiempo de convergencia se podría pensar en la implementación de usuarios móviles. Los usuarios móviles se pueden presentar como usuarios en constante movimiento y con un algoritmo lo suficientemente rápido y preciso se puede realizar un seguimiento de dichos usuarios. Para esta idea también serán útiles los resultados obtenidos de los usuarios nómadas, ya que si consideramos un usuario nómada que se mueve en distancias infinitesimalmente pequeñas en periodos de tiempos muy cortos tendría un comportamiento similar al de un usuario en constante movimiento.

En diversas partes del proyecto se menciona que algunas partes del algoritmo son ejecutados de manera paralela, sin embargo aun sigue existiendo cierta secuencialidad en estas operaciones. La idea de aplicar una paralelismo a través de hilos de ejecución se presenta como una alternativa que podría mejorar de manera notable el tiempo de ejecución del algoritmo.

La implementación física de este proyecto también será una parte fundamental de trabajos a futuro que se puedan llevar a cabo. La idea es buscar pasar de esta etapa de concepción y simulación a una etapa de implementación en un contexto real. Para esto será necesario

buscar el hardware y dispositivos que puedan ser utilizados para materializar esta idea y lógicamente buscar acoplar esto con el trabajo ya desarrollado. La parte de software también será importante ya que más allá de que MATLAB se presenta como la herramienta idónea para realizar las simulaciones, se debe tomar en cuenta que implementar dicho algoritmo en un lenguaje de programación compatible para programar antenas es de vital importancia para el funcionamiento del sistema físico.

Apéndice A

Anexos

A.1. Códigos

A.1.1. Códigos arreglos planos

A.1.1.1. Código de ejecución estrategia hibrida RBE

```
function [BestW,time,cont,L1,L2,L3]=RelayRun(PSOdata,GAdata,
    CUCKOOdata,BATdata)

global NormalizedDesiredSolution
global BestFun
global BestW
global numparticulas
global tol
global diselementos; %Distancia entre elementos de la antenna (
    Flotante)
global Nusuarios;           % Numero de usuarios
global U1;                    %   - U1          ubicaci n usuario 1
    en grados    (entero)
global U2;                    %   - U2          ubicaci n usuario 2
    en grados    (entero)
global U3                      %   - U3          ubicaci n usuario 3
     en grados    (entero)
global maxatasco
global maxit
global viPSO
global viBAT
global Loudness
global pulserate
global desiredsolution
global k
```

```
21 global initialpulserate
22 global pHistory
23 global FitHistory
24 global bar
25 global T
26 global acc
27 global PSObar
28 global GAbar
29 global BATglobaldiff
30 global CUCKOObar
31 global PSOglobaldiff
32 global GAglobaldiff
33 global CUCKOOglobaldiff
34 global pHistoryPSO
35 global FitHistoryPSO
36 global pHistoryCUCKOO
37 global FitHistoryCUCKOO
38 global pHistoryBAT
39 global FitHistoryBAT
40 global pHistoryGA
41 global FitHistoryGA
42 global BestFunGA
43 global BestFunBAT
44 global BestFunPSO
45 global BestFunCUCKOO
46 global rang
47
48
49
50
51 rang=6;
52
53
54
55
56
57
58
```

```
59 numparticulas=40; %- Numero de particulas (Entero)
60 antdim=8;                %Dimension antena (Entero)
61 NumAntElem=(antdim^2); %Numero de elementos de la antenna (Entero)
62 [wi]=initalize(NumAntElem,numparticulas); %inicializa y genera el
      vector de pesos wi tomando en cuenta el numero de elementos de
      la antena y el numero de particulas
63 [wi]=enrango(wi); %verifica que el vector de pesos wi este dentro
      del rango 0 a 2pi
64
65 %-------------------------------Inicializacion de variables
      ---------------------------------
66 tol=0.001;                %   - tolerancia  (flotante)
67 diselementos=0.5; %Distancia entre elementos de la antenna (
      Flotante)
68 Nusuarios=2;          % Numero de usuarios
69 U1=20;                    %   - U1          ubicaci n usuario 1 en
      grados    (entero)
70 U2=50;                    %   - U2          ubicaci n usuario 2 en
      grados    (entero)
71 U3=80;                      %   - U3          ubicaci n usuario 3 en
       grados    (entero)
72 errPSO=inf;                     % Error
73 errGA=inf;                    % Error
74 errBAT=inf;                     % Error
75 errCUCKOO=inf;                      % Error
76
77 maxatasco=20;
78 AtascoCountPSO=0;
79 AtascoCountGA=0;
80 AtascoCountBAT=0;
81 AtascoCountCUCKOO=0;
82
83
84 vi=zeros(numparticulas,NumAntElem); %   - Nusuarios         Numero
      de usuarios (entero)
85 maxit = 45;
86 go = 1;
87 k = 1;
```

```
88
89  alpha    = BATdata(1);                % Parametro alpha
90  gamma    = BATdata(2);                % Parametro gamma
91  aceleracion     = BATdata(3);                % Constante de
        Aceleracion
92  theta    = BATdata(4);                % Inercia
93 Amin     = 1;                    % Volumen minimo
94 Amax     = 2;                    % Volumen maximo
95 xmin     =0;             % Posicion Minima
96 xmax     = 2*pi;             % Posicion Maxima
97 viPSO=zeros(10,64); %   - Nusuarios        Numero de usuarios (
       entero)
98
99 viBAT=zeros(10,64);
100 %
101 %
102 %
103 Loudness        = (Amax-Amin).*rand(10,1)+Amin;  % Volumen inicial(
       Loudness)
104 initialpulserate       = 0.1.*rand(10,1);                % Tasa de
       emision de pulsos inicial cercana a cero
105 pulserate        = initialpulserate*(1-exp(-gamma*k));          %
       Aumenta cota r_0;
106 % %
107 %
108
109
110 %SA parameters
111 To=5000;                          % Temperatura Inicial
112 Tf=1e-1;                          % Temperatura final
113 itera=35;                        % Cantidad total de aceptados
114 iterr=250;                        % Cantidad total de
       rechazosejecuciones
115 alpha=0.9;                       % Proporcion temperatura
116 tol=0.1;                         % Accuracy o precision para error
117 % numparticulas=1; %- Numero de particulas (Entero)
118 % itertotal=200;
119 % Execution=0;                              % Ejecuciones
```

```
120 acc=0;                                  % Aceptados
121 rej=0;                                  % Rechazados
122 iter=0;
123 T=To;
124
125
126 FitHistoryPSO = zeros(1,maxit);   pHistoryPSO = zeros(size(wi,2),
        maxit);
127  FitHistoryBAT = zeros(1,maxit);   pHistoryBAT = zeros(size(wi,2),
        maxit);
128  FitHistoryCUCKOO = zeros(1,maxit); pHistoryCUCKOO = zeros(size(wi
        ,2),maxit);
129  FitHistoryGA = zeros(1,maxit);   pHistoryGA = zeros(size(wi,2),
        maxit);
130  FitHistorySA = zeros(1,maxit);   pHistorySA = zeros(size(wi,2),
        maxit);
131
132
133
134
135  FitHistory = zeros(1,maxit);   pHistory = zeros(size(wi,2),maxit);
136
137   % costfunction returns the fitness vector and the best solution
138 %   [fitness,best] = costfunction(pin);
139 tic
140 %-------------------------------------Algoritmo PSO
        ---------------------------------
141 [solutions,desiredsolution,NormalizedDesiredSolutionVector]=
        costfunction(Nusuarios,U1,U2,U3,numparticulas,wi); %evalua el
        vector de pesos wi en la funcion de costos
142 [BestW,BestFun,NormalizedDesiredSolution,I,bar]=choosebest(
        solutions,NormalizedDesiredSolutionVector,wi); %elige el
        conjunto de vectores de peso que pr
143 [diffini]=calcproduct(BestW, desiredsolution,Nusuarios,wi);
144 BestFun=BestFun/diffini;
145 pHistory = BestW ; FitHistory = BestFun;
146 FitHistoryPSO = FitHistory;   pHistoryPSO = BestW;
147  FitHistoryBAT = FitHistory;   pHistoryBAT =BestW;
```

```
148  FitHistoryCUCKOO = FitHistory; pHistoryCUCKOO = BestW;
149  FitHistoryGA = FitHistory;  pHistoryGA = BestW;
150  FitHistorySA = FitHistory;  pHistorySA = BestW;
151  PSOglobaldiff =2; GAglobaldiff =2; CUCKOOglobaldiff=2;
       BATglobaldiff=2;
152
153 %Plotting(antdim,diselementos, BestW )
154
155 BestFunGA=BestFun;
156 BestFunBAT=BestFun;
157 BestFunPSO=BestFun;
158 BestFunCUCKOO=BestFun;
159
160
161
162   errvGA(k)=inf;
163   errvCUCKOO(k)=inf;
164   errvPSO(k)=inf;
165   errvBAT(k)=inf;
166
167
168 itcount=1;
169 errv(k)=inf;
170 tic
171 select1=1;
172 select2=2;
173 select3=3;
174 select4=4;
175 s=1;
176 while go
177         k = k + 1;
178         [BestWGA,BestFunGA,BestWBAT,BestFunBAT,BestWPSO,BestFunPSO
              ,BestWCUCKOO,BestFunCUCKOO]=ParallelRelayblock(wi,
              select1, select2, select3, select4,PSOdata,GAdata,
              CUCKOOdata,BATdata);
179         [errGA,AtascoCountGA,k,go,~]=SimpleEnd(BestFunGA, errGA,
              NormalizedDesiredSolution,k,AtascoCountGA,maxit,
              maxatasco,s);
```

```
180             [errCUCKOO,AtascoCountCUCKOO,k,go,~]=SimpleEnd(
                    BestFunCUCKOO, errCUCKOO, NormalizedDesiredSolution,k,
                    AtascoCountCUCKOO,maxit,maxatasco,s);
181             [errPSO,AtascoCountPSO,k,go,~]=SimpleEnd(BestFunPSO,
                    errPSO, NormalizedDesiredSolution,k,AtascoCountPSO,
                    maxit,maxatasco,s);
182             [errBAT,AtascoCountBAT,k,go,~]=SimpleEnd(BestFunBAT,
                    errBAT, NormalizedDesiredSolution,k,AtascoCountBAT,
                    maxit,maxatasco,s);
183             errvGA(k)=errGA;
184             errvCUCKOO(k)=errCUCKOO;
185             errvPSO(k)=errPSO;
186             errvBAT(k)=errBAT;
187
188
189
190 end
191 time=toc;
192 go = 1;
193 k2 = 1;
194 AtascoCount=0;
195 maxatasco=45;
196
197 T=50;
198 maxit=200;
199
200 pHistoryGASA=pHistoryGA;
201 FitHistoryGASA=FitHistoryGA;
202 tolSA=tol;
203 GASAdiff=GAglobaldiff ;
204 % BestWSA=BestWPBlock;
205 % BestFunSA=BestFunPBlock;
206 numparticulas=1;
207
208 while go
209       [BestWGA,BestFunGA,FitHistoryGASA,pHistoryGASA,GASAdiff]=
          SAblock(BestWGA,alpha,tolSA,rej,itcount,BestFunGA,GASAdiff,
          FitHistoryGASA,pHistoryGASA);
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%[BestWSA,BestFunSA,SAFitHistory,SApHistory, SAglobaldiff]=SAblock(BestWSA,SAalpha,tol,rej,itcount,BestFunSA,SAglobaldiff,SAFitHistory,SApHistory);

    k2=k2+1;
    [errGA,AtascoCount,k2,go,~]=SimpleEnd(BestFunGA, errGA,NormalizedDesiredSolution,k2,AtascoCount,maxit,maxatasco,s);
    errvGA(k+k2-1)=errGA;
end
tam=size(FitHistoryGASA);
nor=norm(NormalizedDesiredSolution);
ndv=repmat(nor,1,tam(2));
%errvGA=abs(ndv-FitHistoryGASA);

go = 1;
k2 = 1;
AtascoCount=0;
maxatasco=45;

T=50;
maxit=200;


pHistoryPSOSA=pHistoryPSO;
FitHistoryPSOSA=FitHistoryPSO;
tolSA=tol;
PSOSAdiff=PSOglobaldiff ;


% BestWSA=BestWPBlock;
% BestFunSA=BestFunPBlock;
numparticulas=1;

while go
    [BestWPSO,BestFunPSO,FitHistoryPSOSA,pHistoryPSOSA,PSOSAdiff]=SAblock(BestWPSO,alpha,tolSA,rej,itcount,BestFunPSO,
```

```
        PSOSAdiff, FitHistoryPSOSA, pHistoryPSOSA);
    k2=k2+1;
    [errPSO, AtascoCount, k2, go, ~]=SimpleEnd(BestFunPSO, errPSO,
        NormalizedDesiredSolution, k2, AtascoCount, maxit, maxatasco, s)
        ;
    errvPSO(k+k2-1)=errPSO;
end
tam=size(FitHistoryPSOSA);
nor=norm(NormalizedDesiredSolution);
ndv=repmat(nor,1,tam(2));
%errvPSO=abs(ndv-FitHistoryPSOSA);
go = 1;
k2 = 1;
AtascoCount=0;
maxatasco=45;

T=50;
maxit=200;

pHistoryBATSA=pHistoryBAT;
FitHistoryBATSA=FitHistoryBAT;
tolSA=tol;
BATSAdiff=BATglobaldiff ;

% BestWSA=BestWPBlock;
% BestFunSA=BestFunPBlock;
numparticulas=1;

while go
    [BestWBAT, BestFunBAT, FitHistoryBATSA, pHistoryBATSA, BATSAdiff]=
        SAblock(BestWBAT, alpha, tolSA, rej, itcount, BestFunBAT,
        BATSAdiff, FitHistoryBATSA, pHistoryBATSA);
    k2=k2+1;
    [errBAT, AtascoCount, k2, go, ~]=SimpleEnd(BestFunBAT, errBAT,
        NormalizedDesiredSolution, k2, AtascoCount, maxit, maxatasco, s)
        ;
    errvBAT(k+k2-1)=errBAT;
end
```

```
273 tam=size(FitHistoryBATSA);
274 nor=norm(NormalizedDesiredSolution);
275 ndv=repmat(nor,1,tam(2));
276 %errvBAT=abs(ndv-FitHistoryBATSA);
277 go = 1;
278 k2 = 1;
279 AtascoCount=0;
280 maxatasco=45;
281
282 T=50;
283 maxit=200;
284
285 pHistoryCUCKOOSA=pHistoryCUCKOO;
286 FitHistoryCUCKOOSA=FitHistoryCUCKOO;
287 tolSA=tol;
288 CUCKOOSAdiff=CUCKOOglobaldiff ;
289
290
291 % BestWSA=BestWPBlock;
292 % BestFunSA=BestFunPBlock;
293 numparticulas=1;
294
295 while go
296     [BestWCUCKOO,BestFunCUCKOO,FitHistoryCUCKOOSA,pHistoryCUCKOOSA
        ,CUCKOOSAdiff]=SAblock(BestWCUCKOO,alpha,tolSA,rej,itcount,
        BestFunCUCKOO,CUCKOOSAdiff,FitHistoryCUCKOOSA,
        pHistoryCUCKOOSA);
297     k2=k2+1;
298     [errCUCKOO,AtascoCount,k2,go,~]=SimpleEnd(BestFunCUCKOO,
        errCUCKOO, NormalizedDesiredSolution,k2,AtascoCount,maxit,
        maxatasco,s);
299     errvCUCKOO(k+k2-1)=errCUCKOO;
300 end
301 tam=size(FitHistoryCUCKOOSA);
302 nor=norm(NormalizedDesiredSolution);
303 ndv=repmat(nor,1,tam(2));
304 %errvCUCKOO=abs(ndv-FitHistoryCUCKOOSA);
305 BestfunVect=[BestFunCUCKOO,BestFunBAT,BestFunGA,BestFunPSO];
```

```
306 BestWVect=[BestWCUCKOO;BestWBAT;BestWGA;BestWPSO];
307 % Barvect=[CUCKOObar, BATglobaldiff  ,GAglobaldiff  ,PSObar];
308 FinalFitHistory=FitHistoryCUCKOO;
309 FinalpHistory=pHistoryCUCKOO;
310 % FinalFitHistory=[FitHistoryCUCKOO;FitHistoryBAT;FitHistoryGA;
    FitHistoryPSO]
311 % FinalpHistory=[pHistoryCUCKOO;pHistoryBAT;pHistoryGA;pHistoryPSO
    ]
312 % ERRV=[PSOerrv;CUCKOOerrv;BATerrv;GAerrv;];
313 [BestfunVect,ind] = sort(BestfunVect,'descend');
314 select=ind(1);
315 switch select
316     case 1
317         FinalFitHistory=FitHistoryCUCKOOSA;
318         FinalpHistory=pHistoryCUCKOOSA;
319         Finalerrv=errvCUCKOO;
320         finaldiff=CUCKOOSAdiff;
321         long=length(Finalerrv);
322     case 2
323         FinalFitHistory=FitHistoryBATSA;
324         FinalpHistory=pHistoryBATSA;
325         Finalerrv=errvBAT;
326         finaldiff=BATSAdiff;
327         long=length(Finalerrv);
328     case 3
329         FinalFitHistory=FitHistoryGASA;
330         FinalpHistory=pHistoryGASA;
331         Finalerrv=errvGA;
332          finaldiff=GASAdiff;
333         long=length(Finalerrv);
334     case 4
335         FinalFitHistory=FitHistoryPSOSA;
336         FinalpHistory=pHistoryPSOSA;
337         Finalerrv=errvPSO;
338          finaldiff=PSOSAdiff;
339         long=length(Finalerrv);
340 end
341
```

```
342
343 % BestWVect= BestWVect(ind,:);
344 % Barvect=Barvect(ind);
345 % % FinalFitHistory= FinalFitHistory(ind,:);
346 % % FinalpHistory= FinalpHistory(ind,:);
347 % % ERRV= ERRV(ind,:);
348 BestWSA=BestWVect(1,:);
349 % % FinalFitHistory= FinalFitHistory(1,:);
350 % % FinalpHistory= FinalpHistory(1,:);
351 % finalbar=Barvect(1);
352 % % errvSA=[ERRV(1,:)];
353  BestFunSA=BestfunVect(1);
354
355
356 % Plotting(antdim,diselementos, BestWPBlock )
357 go = 1;
358 k2 = 1;
359 AtascoCount=0;
360 maxatasco=45;
361
362 err=errv(end);
363 maxit=200;
364
365 % BestWSA=BestWPBlock;
366 % BestFunSA=BestFunPBlock;
367 numparticulas=1;
368 %[solutions,DesiredSolution,NormalizedDesiredSolutionVector]=
      costfunction(Nusuarios,U1,U2,U3,numparticulas,BestWSA,bar);
369
370 T=5000;
371
372 while go
373       [BestWSA,BestFunSA,FinalFitHistory,FinalpHistory,finaldiff]=
          SAblock(BestWSA,alpha,tol,rej,itcount,BestFunSA,finaldiff,
          FinalFitHistory,FinalpHistory);
374       k2=k2+1;
375       [err,AtascoCount,k2,go,~]=SimpleEnd(BestFunSA, err,
          NormalizedDesiredSolution,k2,AtascoCount,maxit,maxatasco,s)
```

```
        ;
376     Finalerrv(long+k2-1)=err;
377 end
378 cont=long+k2;
379 time=toc;
380
381 k=1;
382 p=1;
383 [b,sizeFE]=size(Finalerrv);
384 errchangef=1;
385 for n=p:sizeFE
386     if k==1
387         previouserrorA=Finalerrv(1);
388     else
389         previouserrorA=Finalerrv(k-1);
390     end
391
392     err=Finalerrv(k);
393   if previouserrorA==err
394
395  else
396       errchangef(end+1)=k;
397       AtascoCount=0;
398   end
399 k=k+1;
400 end
401
402 BestWit=FinalpHistory(errchangef,:);
403 [tam,b]=size(BestWit)
404  k=0;
405  i=1;
406 for cycle=i:tam
407     k=k+1;
408     BestWitp=BestWit(k,:);
409 %       Plotting(antdim,diselementos, BestWitp ) %se grafican los
      resultados
410 %       BestWit=pHistory(end,:);
411 %       Plotting(antdim,diselementos, BestWit ) %se grafican los
```

```
    resultados
end


% Plotting(antdim,diselementos, BestWSA );
[L1,L2,L3]=Ganancia(BestWSA, U1,U2,U3,Nusuarios,wi,rang);

 %

end



function[diff]=calcproduct(Pout, desiredsolution,Nusuarios,wi)
rad=pi/180;
cant=1800;

NumAntElem=size(wi,2);
NumAntElem=sqrt(NumAntElem);
antdim=[NumAntElem NumAntElem];
diselementos=[0.5 0.5];

thetar=[0 180]*rad;
THETARAD=linspace(thetar(1),thetar(2),cant);
ArrayFactor=faf2(exp(1i*Pout),antdim,diselementos,THETARAD,0);%
    Patron de radiacion de la particula i
NormalizedAF=abs(ArrayFactor)/sum(abs(ArrayFactor));
prod=NormalizedAF.*desiredsolution;
%plot(prod)

% x = linspace(0,180,1800);
%
% plot(x,prod)
if Nusuarios==2
      [maxvalue1,pos1]=max(prod);
   if pos1>100
          posv1=[pos1-100:pos1+100];
          prod(1,posv1) = 0;
```

```
448       else
449           posv1=[0:pos1+100];
450           prod(1,posv1+1) = 0;
451
452       end
453
454
455
456       %plot(prod)
457
458       [maxvalue2,pos2]=max(prod);
459       if pos2>100
460           posv2=[pos2-100:pos2+100];
461           prod(1,posv2) = 0;
462       else
463           posv2=[0:pos2+100];
464           prod(1,posv2+1) = 0;
465
466       end
467
468
469       %plot(prod)
470
471       diff=abs(maxvalue1-maxvalue2);
472   elseif Nusuarios==3
473
474
475
476       [maxvalue1,pos1]=max(prod);
477         if pos1>100
478           posv1=[pos1-100:pos1+100];
479           prod(1,posv1) = 0;
480       else
481           posv1=[0:pos1+100];
482           prod(1,posv1+1) = 0;
483
484       end
485
```

```
%       plot(prod)

    [maxvalue2,pos2]=max(prod);
     if  pos2>100
          posv2=[pos2-100:pos2+100];
          prod(1,posv2) = 0;
     else
          posv2=[0:pos2+100];
          prod(1,posv2+1) = 0;

     end


%       plot(prod)

    [maxvalue3,pos3]=max(prod);
     if  pos3>100
          posv3=[pos3-100:pos3+100];
          prod(1,posv3) = 0;
     else
          posv3=[0:pos3+100];
          prod(1,posv3+1) = 0;

     end

   diff=abs((maxvalue1-maxvalue2)+(maxvalue1-maxvalue3));
end


end









function[err,AtascoCount,iter]=errorcalc(BestFun, err,
    NormalizedDesiredSolution,iter,AtascoCount)
```

```
523 previouserrorA=err;
524  err=sum(abs( BestFun-norm(NormalizedDesiredSolution)));
525  if previouserrorA==err
526       AtascoCount=AtascoCount+1;
527  else
528       AtascoCount=0;
529  end
530  iter=iter+1;
531 end
532 function [BestW,BestFun]=Compare(NewW,NewFun,
      NormalizedDesiredSolutionVector,BestW,BestFun)
533 if (NewFun>BestFun) && (NewFun<=norm(
      NormalizedDesiredSolutionVector))
534           BestFun=NewFun
535           BestW=NewW
536
537      end
538 end
539 function [wi]=NewPosition(vi,wi)
540  wi=wi+vi;
541 end
542
543 function [vi]=NewVelocity(alpha,beta,inercia,numparticulas,
      NumAntElem,vi,BestW,BestFun,wi)
544       vi=inercia*vi+alpha*rand(numparticulas,NumAntElem).*(repmat(
            BestW,numparticulas,1)-wi)+beta*rand(numparticulas,
            NumAntElem).*(repmat(BestFun,numparticulas,1)-wi);
545
546 % vi=inercia*vi+alpha*rand(n,NumAntElem).*(repmat(BestW,n,1)-wi)+
      beta*rand(n,NumAntElem).*(repmat(BestFun,n,1)-wi);
547 %    wi=wi+vi;
548 %       wi3=mod(wi,2*pi);
549 end
550 function [BestW,BestF,NormalizedDesiredSolution,I,bar]=choosebest(
      solutions,NormalizedDesiredSolutionVector,wi)
551
552 [~,I] = max(solutions);
553 NormalizedDesiredSolution=NormalizedDesiredSolutionVector(I,:);
```

```
554 BestW =wi(I,:);
555 BestF=solutions(I);
556 bar=max(NormalizedDesiredSolution);
557
558 end
559
560 function[solutions,DesiredSolution,NormalizedDesiredSolutionVector
    ]=costfunction(Nusuarios,U1,U2,U3,numparticulas,wi) %funcion
    que evalua el vector de pesos y genera un diagrama de solucion
    con los usuarios.
561 global rang
562 gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo; %
    funcion gaussiana
563 amp=1000; %amplitud
564 width=rang; %ancho (para distancias cercanas es 6, distancias
    medianas 4.5, distancias lejanas 6)
565 cant=1800; %cantidad de puntos
566 desplazamiento=0; %desplazamiento de las campanas
567 thetar=[0 180]; %rango de theta
568 THETA=linspace(thetar(1),thetar(2),cant); %theta en grados
569 rad=pi/180; %radianes
570 NumAntElem=size(wi,2);
571 NumAntElem=sqrt(NumAntElem); %numero de elementos de la antenna
572 antdim=[NumAntElem NumAntElem]; %dimensiones de la antea
573 diselementos=[0.5 0.5]; %distancia entre elementos en la antenna
574 N=[NumAntElem, NumAntElem];
575 thetar=[0 180]*rad; %theta en radianes
576 THETARAD=linspace(thetar(1),thetar(2),cant);     % 1 x cant va
    desde 0 a pi en 1800 pasos
577 k=2*pi;
578 d=diselementos;
579 if Nusuarios==2 %caso si se tiene 2 usuarios
580     usuarios=[U1 U2]; %ubicacion usuarios
581     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
        desplazamiento)+gaus(THETA,usuarios(2),width,amp,
        desplazamiento);%+gaus(THETA,t0(3),sig,amp,vo);
582      %se genera la solucion deseada que sera utilizada como
         referencia (funcion gaussiana)
```

```
583 elseif Nusuarios==3
584     usuarios=[U1 U2 U3] ; %ubicacion usuarios
585     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
            desplazamiento)+gaus(THETA,usuarios(2),width,amp,
            desplazamiento)+gaus(THETA,usuarios(3),width,amp,
            desplazamiento);
586     %se genera la solucion deseada que sera utilizada como
            referencia (funcion gaussiana)
587 end
588
589 Psix = (((1:N(1))-1).*(k*d(1)*sin(THETARAD).*cos(0))');
590 Psiy = (((1:N(2))-1).*(k*d(2)*sin(THETARAD).*sin(0))');
591 for i=1:N(1)
592     sumaPsi(N(1)*(i-1)+1:i*N(1),:)=Psix(i,:)+Psiy;
593 end
594 sumaPsi=exp(1i*sumaPsi); %64x1800 complex double;
595 ArrayFactor1            = exp(1i*wi)*sumaPsi;
596
597
598
599 %ArrayFactor1=faf2(exp(1i*wi),antdim,diselementos,THETARAD,0); %se
        calcula el factor de arreglo
600 NormalizedAF1=abs(ArrayFactor1); %valor absoluto del factor de
        arreglo
601  sumNAF1=sum(NormalizedAF1,2); %el valor total sumado de todos los
         elementos
602  NAF1=NormalizedAF1./sumNAF1; %Normalizacion del factor de arreglo
603  M = max(NAF1,[],2); %se encuentra el punto maximo del factor de
        arreglo normalizado
604   DesiredSolution1=repmat(DesiredSolution,numparticulas,1); %se
        genera matriz repetida de solucion deseada
605 NormalizedDesiredSolutionVector=DesiredSolution1.*M; %multiplico
        el punto mas alto por la solucion deseada(la solcucion deseada
        (desired solution) tiene una amplitud de mil y el punto maximo
        es como de 2x10-3, este producto se hace para obtener una
        solucion deseada que este en una misma escala que las
        soluciones que se van a ir propionendo)
606 prodv=NAF1.*DesiredSolution1; %multiplico la solucion deseada por
```

```
    el factor de arreglo normalizado
solutions=sqrt(sum(prodv.^2,2)); %calculo la norma del producto
    anterior


end



function [wi]=enrango(wi)
wi=mod(wi,2*pi);
end

function [wi]=initalize(NumAntElem,numparticulas)
LB=0*ones(1,NumAntElem);            % Limite inferior
UB=2*pi*ones(1,NumAntElem);         % Limite Superior
wmin=min(LB);                 % w minimo
wmax=max(UB);                 % w maximo



wi=(rand(numparticulas,NumAntElem))*(wmax-wmin)+wmin;
end

function Plotting(antdim,diselementos, BestW )
rad=pi/180;
antdim=[antdim antdim];
diselementos=[diselementos diselementos];

thetar=[0 180]*rad;
THETARAD=linspace(thetar(1),thetar(2),1800);     % 1 x cant va
    desde 0 a pi en 1800 pasos


AF=faf(exp(1i*BestW),antdim,diselementos,THETARAD,0);
AFnorma=abs(AF/sum(abs(AF)));
figure(9);
plot(AFnorma);
```

```
642
643
644 W=exp(1i*BestW);
645 Wbestpos=abs(W);                                    % Magnitudes
646 betabestpos=mod(angle(W)*180/pi,360);              % Angulos
647 nombre='PSO';
648
649 radtheta(W,antdim,diselementos,0,nombre)
650 % %----------------------Patron de Radiaci  n en Theta Polares
      ----------------------
651  radthetapol(W,antdim,diselementos,0,nombre)
652
653
654
655
656
657
658 end
659
660
661
662 function[L1,L2,L3]=Ganancia(Pout, U1,U2,U3,Nusuarios,wi,rang)
663 rad=pi/180;
664 cant=1800;
665 scale=-80;
666 cant=1800;
667 THETA=linspace(0,2*pi,cant);
668 % Ref=NormalizedDesiredSolution;
669 Ref= linspace(3.17,3.17,cant);
670 THETA1=linspace(0,180,cant);
671 L1=0;
672 L2=0;
673 L3=0;
674
675
676 NumAntElem=size(wi,2);
677 NumAntElem=sqrt(NumAntElem);
678 antdim=[NumAntElem NumAntElem];
```

```
679 diselementos=[0.5  0.5];
680
681 thetar=[0  180]*rad;
682 THETARAD=linspace(thetar(1),thetar(2),cant);
683 ArrayFactor=faf2(exp(1i*Pout),antdim,diselementos,THETARAD,0);%
      Patron de radiacion de la particula i
684 NormalizedAF=abs(ArrayFactor)/sum(abs(ArrayFactor));
685 Naf=NormalizedAF*1800;
686 % AFdBph1=10.*log10((Naf).^2);
687 % plot(AFdBph1)
688
689 AFdBph1=10.*log10((NormalizedAF).^2);
690 RefdB=10.*log10((Ref(1,:)).^2);
691 [AFdB] = polardB(AFdBph1,scale);
692
693 U1=U1*10;
694 U2=U2*10;
695 U3=U3*10;
696 rang=rang*10;
697
698 % plot(prod)
699
700 % x = linspace(0,180,1800);
701 %
702 % plot(x,prod)
703 if Nusuarios==2
704     L1=NormalizedAF(U1);
705 %       [~,largo]=size(AF1);
706 %       ref1=RefdB(:,1:largo);
707     %L1=sum(AF1);
708     L1=L1/(1/cant);
709    L1=10.*log10((L1).^2);
710
711
712    L2=NormalizedAF(U2);
713 %       [~,largo]=size(AF1);
714 %       ref1=RefdB(:,1:largo);
715     %L2=sum(AF2);
```

```
716       L2=L2/(1/cant);
717      L2=10.*log10((L2).^2);
718
719       L3=0;
720  elseif  Nusuarios==3
721
722      L1=NormalizedAF(U1);
723  %        [~,largo]=size(AF1);
724  %        ref1=RefdB(:,1:largo);
725       %L1=sum(AF1);
726      L1=L1/(1/cant);
727      L1=10.*log10((L1).^2);
728
729
730      L2=NormalizedAF(U2);
731  %        [~,largo]=size(AF1);
732  %        ref1=RefdB(:,1:largo);
733       %L2=sum(AF2);
734      L2=L2/(1/cant);
735      L2=10.*log10((L2).^2);
736
737
738
739          L3=NormalizedAF(U3);
740  %        [~,largo]=size(AF1);
741  %        ref1=RefdB(:,1:largo);
742  %      L3=sum(AF3);
743       L3=L3/(1/cant);
744      L3=10.*log10((L3).^2);
745
746
747
748  end
749
750  end
```

A.1.1.2. Codigo de estrategia híbrida RBE

```
1  function  [poutv,BestWPBlock,BestFunPBlock,FitHistory,pHistory]=
```

```
    Parallelblock(wi,select1, select2, select3, select4,PSOdata,
    GAdata,CUCKOOdata,BATdata)
global pulserate
global desiredsolution
global NormalizedDesiredSolution
global BestFun
global BestW
global numparticulas
global Loudness
global Nusuarios;          % Numero de usuarios
global U1;                   %   - U1          ubicacion usuario 1
    en grados   (entero)
global U2;                   %   - U2          ubicacion usuario 2
    en grados   (entero)
global U3                    %   - U3          ubicacion usuario 3
     en grados   (entero)
global k
global initialpulserate
global viPSO
global viBAT
global pHistory
global FitHistory
global bar



count=0;
if select1~= 0
    count=count+1;
end
if select2~= 0
    count=count+1;
end
if select3~= 0
    count=count+1;
end
if select4~= 0
    count=count+1;
```

```
35 end
36
37
38 wi1=wi(1:numparticulas/count,:);
39 wi2=wi((numparticulas/count)+1:2*numparticulas/count,:);
40 wi3=wi(1+(2*numparticulas/count):3*numparticulas/count,:);
41 wi4=wi(1+(3*numparticulas/count):4*numparticulas/count,:);
42
43
44 switch select1
45     case 1
46         %algdata=[pc,pm,mutstep]
47         pc =GAdata(1);
48         pm = GAdata(2);
49         mutstep = 0.3;
50         pout = SimpleGA(wi1,pc,pm,mutstep,Nusuarios,U1,U2,U3);
51
52
53     case 2
54         %algdata=[gamma,lambda,beta]
55
56         gamma = 1;
57         lambda = 1;
58         beta = 3/2;
59         pa = CUCKOOdata(1);
60         pout = SimpleCuckoo(wi1,pa,beta,Nusuarios,U1,U2,U3);
61
62         %       gamma = 1; lambda = 1; pa = ceil(0.90*size(wi,1));
               beta = 3/2;
63     case 3
64         %algdata=[alfa,beta,inercia]
65         alpha = PSOdata(1);  %   - alpha         constante
               aceleracion asociado con mejor solucion local   (entero
               )
66         beta = PSOdata(2);  %   - alpha         constante
               aceleracion asociado con mejor solucion local   (entero
               )
67         inercia = PSOdata(3); %   - inercia      constante de
```

```
            inercia  (entero)
68          pout = SimplePSO(wi1,alpha,beta,inercia,BestW,BestFun);
69
70      case 4
71          alpha    = BATdata(1);                % Parametro alpha
72          gamma    = BATdata(2);                % Parametro gamma
73          aceleracion      = BATdata(3);                % Constante de
               Aceleracion
74          theta    = BATdata(4);                 % Inercia
75          [pout] =  SimpleBAT(wi1,aceleracion,alpha,theta,BestW,
               BestFun,Nusuarios,U1,U2,U3,initialpulserate,k,gamma);
76
77  end
78  poutv=pout;
79  switch select2
80      case 1
81          %algdata=[pc,pm,mutstep]
82          pc =GAdata(1);
83          pm = GAdata(2);
84          mutstep = 0.3;
85          pout = SimpleGA(wi1,pc,pm,mutstep,Nusuarios,U1,U2,U3);
86
87
88      case 2
89          %algdata=[gamma,lambda,beta]
90          gamma = 1;
91          lambda = 1;
92          beta = 3/2;
93          pa = CUCKOOdata(1);
94          pout = SimpleCuckoo(wi1,pa,beta,Nusuarios,U1,U2,U3);
95
96          %       gamma = 1; lambda = 1; pa = ceil(0.90*size(wi,1));
               beta = 3/2;
97      case 3
98          %algdata=[alfa,beta,inercia]
99          alpha = PSOdata(1);  %   - alpha          constante
               aceleracion asociado con mejor solucion local   (entero
               )
```

```
100         beta = PSOdata(2);  %   - alpha         constante
               aceleracion asociado con mejor solucion local   (entero
               )
101         inercia = PSOdata(3); %   - inercia      constante de
               inercia  (entero)
102         pout = SimplePSO(wi1,alpha,beta,inercia,BestW,BestFun);
103
104     case 4
105         alpha    = BATdata(1);                % Parametro alpha
106         gamma    = BATdata(2);                % Parametro gamma
107         aceleracion      = BATdata(3);                % Constante de
               Aceleracion
108         theta    = BATdata(4);                 % Inercia
109         [pout] =  SimpleBAT(wi1,aceleracion,alpha,theta,BestW,
               BestFun,Nusuarios,U1,U2,U3,initialpulserate,k,gamma);
110
111 end
112
113 poutv=cat(1,poutv,pout);
114 switch select3
115     case 1
116         %algdata=[pc,pm,mutstep]
117         pc =GAdata(1);
118         pm = GAdata(2);
119         mutstep = 0.3;
120         pout = SimpleGA(wi1,pc,pm,mutstep,Nusuarios,U1,U2,U3);
121
122
123     case 2
124         %algdata=[gamma,lambda,beta]
125
126         gamma = 1;
127         lambda = 1;
128         beta = 3/2;
129         pa = CUCKOOdata(1);
130         pout = SimpleCuckoo(wi1,pa,beta,Nusuarios,U1,U2,U3);
131
132         %       gamma = 1; lambda = 1; pa = ceil(0.90*size(wi,1));
```

```
            beta = 3/2;
133     case 3
134         %algdata=[alfa,beta,inercia]
135         alpha = PSOdata(1);  %   - alpha         constante
               aceleracion asociado con mejor solucion local   (entero
               )
136         beta = PSOdata(2);  %   - alpha         constante
               aceleracion asociado con mejor solucion local   (entero
               )
137         inercia = PSOdata(3); %   - inercia      constante de
               inercia  (entero)
138         pout = SimplePSO(wi1,alpha,beta,inercia,BestW,BestFun);
139
140     case 4
141         alpha    = BATdata(1);                % Parametro alpha
142         gamma    = BATdata(2);                % Parametro gamma
143         aceleracion      = BATdata(3);                % Constante de
               Aceleracion
144         theta    = BATdata(4);                 % Inercia
145         [pout] =  SimpleBAT(wi1,aceleracion,alpha,theta,BestW,
               BestFun,Nusuarios,U1,U2,U3,initialpulserate,k,gamma);
146
147 end
148 poutv=cat(1,poutv,pout);
149 switch select4
150    case 1
151         %algdata=[pc,pm,mutstep]
152         pc =GAdata(1);
153         pm = GAdata(2);
154         mutstep = 0.3;
155         pout = SimpleGA(wi1,pc,pm,mutstep,Nusuarios,U1,U2,U3);
156
157
158     case 2
159         %algdata=[gamma,lambda,beta]
160
161         gamma = 1;
162         lambda = 1;
```

```
163         beta = 3/2;
164         pa = CUCKOOdata(1);
165         pout = SimpleCuckoo(wi1,pa,beta,Nusuarios,U1,U2,U3);
166
167         %       gamma = 1; lambda = 1; pa = ceil(0.90*size(wi,1));
                beta = 3/2;
168     case 3
169         %algdata=[alfa,beta,inercia]          %algdata=[alfa,beta,
                inercia]
170         alpha = PSOdata(1);  %   - alpha         constante
                aceleracion asociado con mejor solucion local   (entero
                )
171         beta = PSOdata(2);  %   - alpha         constante
                aceleracion asociado con mejor solucion local   (entero
                )
172         inercia = PSOdata(3); %   - inercia      constante de
                inercia  (entero)
173         pout = SimplePSO(wi1,alpha,beta,inercia,BestW,BestFun);
174
175     case 4
176         alpha    = BATdata(1);                  % Parametro alpha
177         gamma    = BATdata(2);                  % Parametro gamma
178         aceleracion      = BATdata(3);                  % Constante de
                Aceleracion
179         theta    = BATdata(4);                   % Inercia
180         [pout] =  SimpleBAT(wi1,aceleracion,alpha,theta,BestW,
                BestFun,Nusuarios,U1,U2,U3,initialpulserate,k,gamma);
181
182
183 end
184
185         poutv=cat(1,poutv,pout);
186
187         [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
                poutv); %evalua el vector de pesos wi en la funcion de
                costos
188
189         [cost2,ind] = sort(cost,'descend'); % sorting
```

```
190          pout= poutv(ind,:);
191
192         pout(end,:) = pHistory(k-1,:); %The former best sustitute
               now the current worst
193
194         [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
               pout); % Sort again to keep the current best
195         [cost,ind] = sort(cost,'descend');
196         pout= pout(ind,:);
197
198         Localbest=pout(1,:);
199
200
201         [diff]=calcproduct(Localbest, desiredsolution,
               NormalizedDesiredSolution,pout);
202
203
204         ee=isempty(find(diff>bar));
205
206         i=find(cost<=norm(NormalizedDesiredSolution));
207         select=i(1);
208
209         if (ee==1)
210              FitHistory = [FitHistory cost(select)] ;
211              pHistory = [pHistory;pout(select,:)];
212              bar=0.9*bar;
213         else
214              FitHistory = [FitHistory FitHistory(end)] ;
215              pHistory = [pHistory;pHistory(end,:)];
216         end
217
218
219
220         BestFunPBlock=FitHistory(end);
221         BestWPBlock=pHistory(end,:);
222
223
224
```

```
225
226 end
227
228
229 function [solutions,DesiredSolution,NormalizedDesiredSolutionVector
    ]=costfunction(Nusuarios,U1,U2,U3,numparticulas,wi)
230 gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo;
231 amp=1000;
232 width=3;
233 cant=1800;
234 desplazamiento=0;
235 thetar=[0 180];
236 THETA=linspace(thetar(1),thetar(2),cant);
237 rad=pi/180;
238 NumAntElem=size(wi,2);
239 NumAntElem=sqrt(NumAntElem);
240 antdim=[NumAntElem NumAntElem];
241 diselementos=[0.5 0.5];
242
243 thetar=[0 180]*rad;
244 THETARAD=linspace(thetar(1),thetar(2),cant);     % 1 x cant va
    desde 0 a pi en 1800 pasos
245
246 if Nusuarios==2
247     usuarios=[U1 U2];
248     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
        desplazamiento)+gaus(THETA,usuarios(2),width,amp,
        desplazamiento);%+gaus(THETA,t0(3),sig,amp,vo);
249 elseif Nusuarios==3
250      usuarios=[U1 U2 U3] ;
251      DesiredSolution = gaus(THETA,usuarios(1),width,amp,
         desplazamiento)+gaus(THETA,usuarios(2),width,amp,
         desplazamiento)+gaus(THETA,usuarios(3),width,amp,
         desplazamiento);
252 end
253
254
255 for i=1:numparticulas
```

```
256       ArrayFactor=faf(exp(1i*wi(i,:)),antdim,diselementos,THETARAD
              ,0);% Patron de radiacion de la particula i
257        NormalizedAF=abs(ArrayFactor)/sum(abs(ArrayFactor));
258        maximumAF=max(NormalizedAF);   %valor maximo en factor de
              arreglo
259        NormalizedDesiredSolution=DesiredSolution*maximumAF;
260        Product=NormalizedAF.*DesiredSolution;
261       NormalizedDesiredSolutionVector(i,:)=NormalizedDesiredSolution
              ;
262       solutions(i)=norm(Product);
263 end
264
265
266 end
267
268 function[diff]=calcproduct(Pout, desiredsolution,
      NormalizedDesiredSolution,wi)
269 rad=pi/180;
270 cant=1800;
271
272 NumAntElem=size(wi,2);
273 NumAntElem=sqrt(NumAntElem);
274 antdim=[NumAntElem NumAntElem];
275 diselementos=[0.5 0.5];
276
277 thetar=[0 180]*rad;
278 THETARAD=linspace(thetar(1),thetar(2),cant);
279 ArrayFactor=faf(exp(1i*Pout),antdim,diselementos,THETARAD,0);%
      Patron de radiacion de la particula i
280 NormalizedAF=abs(ArrayFactor)/sum(abs(ArrayFactor));
281 prod=NormalizedAF.*desiredsolution;
282 diff=abs(NormalizedDesiredSolution-prod);
283
284 end
```

A.1.1.3. Código de estrategia hibrida MAS

```
1 function [BestWSA,time,cont,L1,L2,L3]=IslandRunv2(GAdata,PSOdata,
      BATdata,CUCKOOdata,SAdata)
```

```
2  global PSOglobaldiff
3  global GAglobaldiff
4  global BATglobaldiff
5  global CUCKOOglobaldiff
6
7  global PSOalpha  %  - alpha        constante aceleracion asociado
       con mejor solucion local  (entero)
8  global PSObeta  %  - alpha         constante aceleracion asociado
     con mejor solucion GLOBAL  (entero)
9  global PSOinercia  %  - inercia     constante de inercia  (entero
     )
10 global NormalizedDesiredSolution
11 global BestFun
12 global BestW
13 global numparticulas
14 global tol
15 global diselementos; %Distancia entre elementos de la antenna (
     Flotante)
16 global Nusuarios;          % Numero de usuarios
17 global U1;                   %  - U1         ubicaci n usuario 1
     en grados  (entero)
18 global U2;                   %  - U2         ubicaci n usuario 2
     en grados  (entero)
19 global U3                    %  - U3         ubicaci n usuario 3
       en grados  (entero)
20 global maxatasco
21 global maxit
22 global vi
23 global PSOpHistory
24 global PSOFitHistory
25 global CUCKOOpHistory
26 global CUCKOOFitHistory
27 global BATpHistory
28 global BATFitHistory
29 global GApHistory
30 global GAFitHistory
31 global Loudness
32 global pulserate
```

```
33 global desiredsolution
34 global Migration4BAT
35 global Migration4PSO
36 global Migration4CUCKOO
37 global Migration4GA
38 global viPSO
39 global viBAT
40 global T
41 global acc
42 global rang
43 global BestWhist
44
45 rang=6;
46
47
48 numparticulas=40; %- Numero de particulas (Entero)
49 antdim=8;                  %Dimension antena (Entero)
50 NumAntElem=(antdim^2); %Numero de elementos de la antenna (Entero)
51 [wiPSO]=initalize(NumAntElem,numparticulas); %inicializa y genera
      el vector de pesos wi tomando en cuenta el numero de elementos
      de la antena y el numero de particulas
52 [wiPSO]=enrango(wiPSO); %verifica que el vector de pesos wi este
      dentro del rango 0 a 2pi
53 [wiGA]=initalize(NumAntElem,numparticulas); %inicializa y genera
      el vector de pesos wi tomando en cuenta el numero de elementos
      de la antena y el numero de particulas
54 [wiGA]=enrango(wiGA); %verifica que el vector de pesos wi este
      dentro del rango 0 a 2pi
55 [wiCUCKOO]=initalize(NumAntElem,numparticulas); %inicializa y
      genera el vector de pesos wi tomando en cuenta el numero de
      elementos de la antena y el numero de particulas
56 [wiCUCKOO]=enrango(wiCUCKOO); %verifica que el vector de pesos wi
      este dentro del rango 0 a 2pi
57 [wiBAT]=initalize(NumAntElem,numparticulas); %inicializa y genera
      el vector de pesos wi tomando en cuenta el numero de elementos
      de la antena y el numero de particulas
58 [wiBAT]=enrango(wiBAT); %verifica que el vector de pesos wi este
      dentro del rango 0 a 2pi
```

```
59
60
61
62 %-------------------------------Inicializacion de variables
       -------------------------------
63 tol=0.001;                  %   - tolerancia   (flotante)
64 diselementos=0.5; %Distancia entre elementos de la antenna (
       Flotante)
65 Nusuarios=2;            % Numero de usuarios
66 U1=20;                      %   - U1          ubicaci n usuario 1 en
       grados    (entero)
67 U2=50;                      %   - U2          ubicaci n usuario 2 en
       grados    (entero)
68 U3=80;                        %   - U3          ubicaci n usuario 3 en
        grados    (entero)
69 errPSO=inf;                       % Error
70 errBAT=inf;
71 errGA=inf;
72 errSA=inf;
73 errCUCKOO=inf;
74 maxatasco=5;
75 AtascoCountPSO=0;
76 AtascoCountBAT=0;
77 AtascoCountGA=0;
78 AtascoCountCUCKOO=0;
79
80 vi=zeros(numparticulas,NumAntElem); %   - Nusuarios        Numero
       de usuarios (entero)
81 maxit = 40;
82 go = 1;
83 k = 1;
84
85           %algdata=[pc,pm,mutstep]
86           pc = GAdata(1);
87           pm = GAdata(2);
88           mutstep = 0.3;
89
90           %algdata=[gamma,lambda,beta]
```

```
91          CUCKOOgamma = 1;
92          lambda = 1;
93          CUCKOObeta = 3/2;
94          pa = CUCKOOdata(1);
95
96          %       gamma = 1; lambda = 1; pa = ceil(0.90*size(wi,1));
                beta = 3/2;
97
98          %algdata=[alfa,beta,inercia]
99          PSOalpha = PSOdata(1);
100         PSObeta = PSOdata(2);
101         PSOinercia = PSOdata(3);
102
103         %algdata=[To,Tf,alpha]
104         To = SAdata(1);
105         Tf = SAdata(2);
106         SAalpha = SAdata(3);
107
108
109
110 %           To=5000;                                % Temperatura Inicial
111 %           Tf=1e-1;                                % Temperatura final
112         itera=35;                                % Cantidad total de
                aceptados
113         iterr=250;                               % Cantidad total de
                rechazosejecuciones
114 %           alpha=0.9;                              % Proporcion temperatura
115         %numparticulas=1; %- Numero de particulas (Entero)
116         itertotal=200;
117         Execution=0;                                    % Ejecuciones
118         acc=0;                                   % Aceptados
119         rej=0;                                   % Rechazados
120         iter=0;
121         T=To;
122
123         %0.9 0.9 0.2 0.5
124         %algdata=[alpha,gamma,aceleracion,theta]
125         BATalpha    = 0.9;                  % Parametro alpha
```

```
126         BATgamma    = 0.9;                  % Parametro gamma
127         aceleracion      = BATdata(1);                  % Constante de
                Aceleracion
128         BATtheta    = BATdata(2);                  % Inercia
129         Amin     = 1;                   % Volumen minimo
130         Amax     = 2;                   % Volumen maximo
131         xmin     =0;             % Posicion Minima
132         xmax     = 2*pi;              % Posicion Maxima
133         viBAT=zeros(numparticulas,NumAntElem);
134         viPSO=zeros(numparticulas,NumAntElem);
135 %
136 %
137 %
138         Loudness         = (Amax-Amin).*rand(numparticulas,1)+Amin;
                % Volumen inicial(Loudness)
139         initialpulserate       = 0.1.*rand(numparticulas,1);
                    % Tasa de emision de pulsos inicial
                cercana a cero
140         pulserate         = initialpulserate*(1-exp(-BATgamma*k));
                % Aumenta cota r_0;
141 % %
142 %
143
144
145  PSOFitHistory = zeros(1,maxit);   PSOpHistory = zeros(size(wiPSO
        ,2),maxit);
146  BATFitHistory = zeros(1,maxit);   BATpHistory = zeros(size(wiPSO
        ,2),maxit);
147  CUCKOOHistory = zeros(1,maxit);   CUCKOOpHistory = zeros(size(
        wiPSO,2),maxit);
148  GAFitHistory = zeros(1,maxit);   GApHistory = zeros(size(wiPSO,2),
        maxit);
149  SAFitHistory = zeros(1,maxit);   SApHistory = zeros(size(wiPSO,2),
        maxit);
150
151
152   % costfunction returns the fitness vector and the best solution
153 %   [fitness,best] = costfunction(pin);
```

```
154
155 %-------------------------------------Algoritmo PSO
    -------------------------------------
156 tic
157 [solutions,desiredsolution,NormalizedDesiredSolutionVector]=
    costfunction(Nusuarios,U1,U2,U3,numparticulas,wiPSO); %evalua
    el vector de pesos wi en la funcion de costos
158 [BestW,BestFun,NormalizedDesiredSolution,I,bar]=choosebest(
    solutions,NormalizedDesiredSolutionVector,wiPSO); %elige el
    conjunto de vectores de peso que pr
159 [diffini]=calcproduct(BestW, desiredsolution,Nusuarios,wiPSO);
160 BestFun=BestFun/diffini;
161 % x = linspace(0,180,1800);
162 % plot(x,NormalizedDesiredSolution)
163 % xlabel('Theta (grados)')
164 % title('Diagrama de radiacion de referencia normalizado')
165 BestWhist=BestW;
166 %Plotting(antdim,diselementos, BestW )
167 PSOglobaldiff=2; GAglobaldiff=2; CUCKOOglobaldiff=2; BATglobaldiff
    =2;
168 PSOpHistory = BestW ; PSOFitHistory = BestFun;
169 CUCKOOpHistory = BestW ; CUCKOOFitHistory = BestFun;
170 BATpHistory = BestW ; BATFitHistory = BestFun;
171 GApHistory = BestW ; GAFitHistory = BestFun;
172 % Plotting(antdim,diselementos, BestW)
173
174 PSOhistory = BestW ; PSOFitHistory = BestFun;
175 CUCKOOhistory = BestW ; CUCKOOFitHistory = BestFun;
176 BAThistory = BestW ; BATFitHistory = BestFun;
177 GAhistory = BestW ; GAFitHistory = BestFun;
178
179
180
181 i=1;
182 helpCUCKOO=0;
183 helpBAT=0;
184 helpGA=0;
185 helpPSO=0;
```

```
186 Migration4PSO=double.empty(0,64);
187 Migration4GA=double.empty(0,64);
188 Migration4BAT=double.empty(0,64);
189 Migration4CUCKOO=double.empty(0,64);
190 BestfunBAT=BestFun;
191 BestfunPSO=BestFun;
192 BestfunGA=BestFun;
193 BestfunCUCKOO=BestFun;
194 PSOerrv(k)=inf;
195 BATerrv(k)=inf;
196 GAerrv(k)=inf;
197 CUCKOOerrv(k)=inf;
198 tic
199 while (k<maxit) % || (helpCUCKOO && helpGA && helpPSO &&
    helpBAT)
200         k = k + 1
201          [BestfunBAT, BestBATWI , wiBAT, Migration4CUCKOO,
              Migration4GA, Migration4PSO, helpCUCKOO, helpGA, helpPSO,
              helpBAT, BATerrv, errBAT, AtascoCountBAT]=BATislandBlock2
              (wiBAT, aceleracion, BATalpha, BATtheta, BATgamma,
              helpCUCKOO, helpPSO, helpGA, initialpulserate, k, errBAT,
              AtascoCountBAT, BestfunCUCKOO, BestfunPSO, BestfunGA,
              BestfunBAT, BATerrv);
202         [BestfunPSO, BestPSOWI, wiPSO, Migration4CUCKOO,
              Migration4GA, Migration4BAT, helpCUCKOO, helpGA, helpPSO,
              helpBAT, PSOerrv, errPSO, AtascoCountPSO]=PSOislandBlock2(
              wiPSO, helpCUCKOO, helpBAT, helpGA, k, AtascoCountPSO, errPSO
              , PSOerrv, BestfunCUCKOO, BestfunBAT, BestfunGA, BestfunPSO)
              ;
203         [BestfunGA, BestGAWI, wiGA, Migration4CUCKOO, Migration4PSO,
              Migration4BAT, helpCUCKOO, helpGA, helpPSO, helpBAT, GAerrv
              , errGA, AtascoCountGA]=GAislandBlock2(wiGA, pc, pm, mutstep
              , helpCUCKOO, helpBAT, helpPSO, k, AtascoCountGA, errGA,
              GAerrv, BestfunCUCKOO, BestfunBAT, BestfunGA, BestfunPSO);
204         [BestfunCUCKOO, BestCUCKOOWI, wiCUCKOO, Migration4PSO,
              Migration4GA, Migration4BAT, helpCUCKOO, helpGA, helpPSO,
              helpBAT, CUCKOOerrv, errCUCKOO, AtascoCountCUCKOO]=
              CUCKOOislandBlock2(wiCUCKOO, pa, CUCKOObeta, helpPSO,
```

```
                helpBAT, helpGA, k, AtascoCountCUCKOO, errCUCKOO, CUCKOOerrv
                , BestfunCUCKOO, BestfunBAT, BestfunGA, BestfunPSO);
205
206             BAThistory = [BAThistory; BestBATWI];
207             GAhistory = [GAhistory; BestGAWI];
208             PSOhistory = [PSOhistory; BestPSOWI];
209             CUCKOOhistory = [CUCKOOhistory; BestCUCKOOWI];
210
211 %               [BestfunPSO, BestPSOWI, wiPSO, Migration4CUCKOO,
    Migration4GA, Migration4BAT, helpCUCKOO, helpGA, helpPSO, helpBAT,
    PSOerrv, errPSO, AtascoCountPSO]=PSOislandBlock2grouptest(wiPSO,
    helpCUCKOO, helpBAT, helpGA, k, AtascoCountPSO, errPSO, PSOerrv,
    BestfunCUCKOO, BestfunBAT, BestfunGA, BestfunPSO);
212 %               [BestfunBAT, BestBATWI, wiBAT, Migration4CUCKOO,
    Migration4GA, Migration4PSO, helpCUCKOO, helpGA, helpPSO, helpBAT,
    BATerrv, errBAT, AtascoCountBAT]=BATislandBlock2grouptest(wiBAT,
    aceleracion, BATalpha, BATtheta, BATgamma, helpCUCKOO, helpPSO,
    helpGA, initialpulserate, k, errBAT, AtascoCountBAT, BestfunCUCKOO,
    BestfunPSO, BestfunGA, BestfunBAT, BATerrv);
213 %               [BestfunGA, BestGAWI, wiGA, Migration4CUCKOO,
    Migration4PSO, Migration4BAT, helpCUCKOO, helpGA, helpPSO, helpBAT,
    GAerrv, errGA, AtascoCountGA]=GAislandBlock2grouptest(wiGA, pc, pm,
    mutstep, helpCUCKOO, helpBAT, helpPSO, k, AtascoCountGA, errGA, GAerrv
    , BestfunCUCKOO, BestfunBAT, BestfunGA, BestfunPSO);
214 %               [BestfunCUCKOO, BestCUCKOOWI, wiCUCKOO, Migration4PSO,
    Migration4GA, Migration4BAT, helpCUCKOO, helpGA, helpPSO, helpBAT,
    CUCKOOerrv, errCUCKOO, AtascoCountCUCKOO]=
    CUCKOOislandBlock2grouptest(wiCUCKOO, pa, CUCKOObeta, helpPSO,
    helpBAT, helpGA, k, AtascoCountCUCKOO, errCUCKOO, CUCKOOerrv,
    BestfunCUCKOO, BestfunBAT, BestfunGA, BestfunPSO);
215
216 %               [BestfunPSO, BestPSOWI, wiPSO, Migration4CUCKOO,
    Migration4GA, Migration4BAT, helpCUCKOO, helpGA, helpPSO, helpBAT,
    PSOerrv, errPSO, AtascoCountPSO]=PSOislandBlock2grouptest3(wiPSO,
    helpCUCKOO, helpBAT, helpGA, k, AtascoCountPSO, errPSO, PSOerrv,
    BestfunCUCKOO, BestfunBAT, BestfunGA, BestfunPSO);
217 %               [BestfunBAT, BestBATWI, wiBAT, Migration4CUCKOO,
    Migration4GA, Migration4PSO, helpCUCKOO, helpGA, helpPSO, helpBAT,
```

```
    BATerrv,errBAT,AtascoCountBAT]=BATislandBlock2grouptest3(wiBAT,
    aceleracion,BATalpha,BATtheta,BATgamma,helpCUCKOO,helpPSO,
    helpGA,initialpulserate,k,errBAT,AtascoCountBAT,BestfunCUCKOO,
    BestfunPSO,BestfunGA,BestfunBAT,BATerrv);
% [BestfunGA, BestGAWI, wiGA,Migration4CUCKOO,
    Migration4PSO,Migration4BAT, helpCUCKOO,helpGA,helpPSO,helpBAT,
    GAerrv,errGA,AtascoCountGA]=GAislandBlock2grouptest3(wiGA,pc,pm
    ,mutstep,helpCUCKOO,helpBAT,helpPSO,k,AtascoCountGA,errGA,
    GAerrv,BestfunCUCKOO,BestfunBAT,BestfunGA,BestfunPSO);
% [BestfunCUCKOO, BestCUCKOOWI, wiCUCKOO, Migration4PSO,
    Migration4GA,Migration4BAT, helpCUCKOO,helpGA,helpPSO,helpBAT,
    CUCKOOerrv,errCUCKOO,AtascoCountCUCKOO]=
    CUCKOOislandBlock2grouptest3(wiCUCKOO,pa,CUCKOObeta,helpPSO,
    helpBAT,helpGA,k,AtascoCountCUCKOO,errCUCKOO,CUCKOOerrv,
    BestfunCUCKOO,BestfunBAT,BestfunGA,BestfunPSO);
% %


end
numparticulas=1;
BestfunVect=[BestfunPSO,BestfunCUCKOO,BestfunBAT,BestfunGA];
BestWVect=[BestPSOWI;BestCUCKOOWI;BestBATWI;BestGAWI];
% ERRV=[PSOerrv;CUCKOOerrv;BATerrv;GAerrv;];
[BestfunVect,ind] = sort(BestfunVect,'descend');
BestWVect= BestWVect(ind,:);
% ERRV= ERRV(ind,:);
BestWSA=BestWVect(1,:);
% errvSA=[ERRV(1,:)];
BestFunSA=BestfunVect(1);

if ind(1)==1
    pHistory=PSOhistory;
    errvSA=PSOerrv;
elseif ind(1)==2
    pHistory=CUCKOOhistory;
    errvSA=CUCKOOerrv;
elseif ind(1)==3
    pHistory=BAThistory;
```

```
243         errvSA=BATerrv;
244     elseif ind(1)==4
245         pHistory=GAhistory;
246         errvSA=GAerrv;
247     end
248
249
250     SApHistory = BestWSA ;
251     SAFitHistory = BestFunSA;
252
253     go = 1;
254     k2 = 1;
255     SAAtascoCount=0;
256     itcount=1;
257     maxit=136;
258     maxatascoSA=45;
259     SAglobaldiff=PSOglobaldiff;
260     errchange=1;
261     while go
262         k2 = k2 + 1;
263         [BestWSA,BestFunSA,SAFitHistory,SApHistory, SAglobaldiff]=
              SAblock(BestWSA,SAalpha,tol,rej,itcount,BestFunSA,
              SAglobaldiff,SAFitHistory,SApHistory);
264
265         [errSA,SAAtascoCount,k2,go,errchange]=SimpleEnd(BestFunSA,
              errSA, NormalizedDesiredSolution,k2,SAAtascoCount,maxit,
              maxatascoSA,errchange);
266         cont=(k*4)+k2;
267         errvSA(40+k2-1)=errSA;
268         pHistory = [pHistory;SApHistory(end,:)];
269     end
270     k=1;
271     time=toc;
272
273     p=1;
274     [b,sizeSA]=size(errvSA);
275     errchangef=1;
276     for n=p:sizeSA
```

```
277       if k==1
278           previouserrorA=errvSA(1);
279       else
280           previouserrorA=errvSA(k-1);
281       end
282
283       err=errvSA(k);
284     if previouserrorA==err
285
286    else
287         errchangef(end+1)=k;
288         AtascoCount=0;
289     end
290  k=k+1;
291  end
292
293  BestWit=pHistory(errchangef,:);
294  [tam,b]=size(BestWit)
295   k=0;
296  for cycle=i:tam
297       k=k+1;
298       BestWitp=BestWit(k,:);
299  %      Plotting(antdim,diselementos, BestWitp ) %se grafican los
       resultados
300  %       BestWit=pHistory(end,:);
301  %       Plotting(antdim,diselementos, BestWit ) %se grafican los
       resultados
302  end
303
304
305   [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,BestWSA);
306   rad=pi/180;
307  antdim=[antdim antdim];
308  diselementos=[diselementos diselementos];
309  thetar=[0 180]*rad;
310  THETARAD=linspace(thetar(1),thetar(2),1800);      % 1 x cant va
       desde 0 a pi en 1800 pasos
311   ArrayFactor1=faf2(exp(1i*BestWSA),antdim,diselementos,THETARAD,0)
```

```
    ;
312  NormalizedAF1=abs(ArrayFactor1);
313  sumNAF1=sum(NormalizedAF1,2);
314  NAF1=NormalizedAF1./sumNAF1;
315  d=desiredsolution;
316 %  figure(4)
317 %  plot(desiredsolution)
318 %  xlabel('Theta (grados)')
319
320  final=NAF1.*d;
321 %  figure(5)
322 %  plot(x,final);
323 %  xlabel('Theta (grados)')
324 %  title('Producto entre factor de arreglo y diagrama de radiacion
       de referencia')
325
326
327  [L1,L2,L3]=Ganancia(BestWSA, U1,U2,U3,Nusuarios,wiPSO,rang)
328
329  %Plotting(antdim,diselementos, BestWSA,NormalizedDesiredSolution
       )
330 end
331
332 function [L1,L2,L3]=Ganancia(Pout, U1,U2,U3,Nusuarios,wi,rang)
333 rad=pi/180;
334 cant=1800;
335 scale=-80;
336 cant=1800;
337 THETA=linspace(0,2*pi,cant);
338 % Ref=NormalizedDesiredSolution;
339 Ref= linspace(3.17,3.17,cant);
340 THETA1=linspace(0,180,cant);
341 L1=0;
342 L2=0;
343 L3=0;
344
345
346 NumAntElem=size(wi,2);
```

```
347 NumAntElem=sqrt(NumAntElem);
348 antdim=[NumAntElem NumAntElem];
349 diselementos=[0.5 0.5];
350
351 thetar=[0 180]*rad;
352 THETARAD=linspace(thetar(1),thetar(2),cant);
353 ArrayFactor=faf2(exp(1i*Pout),antdim,diselementos,THETARAD,0);%
    Patron de radiacion de la particula i
354 NormalizedAF=abs(ArrayFactor)/sum(abs(ArrayFactor));
355 Naf=NormalizedAF*1800;
356 % AFdBph1=10.*log10((Naf).^2);
357 % plot(AFdBph1)
358
359 AFdBph1=10.*log10((NormalizedAF).^2);
360 RefdB=10.*log10((Ref(1,:)).^2);
361 [AFdB] = polardB(AFdBph1,scale);
362
363 U1=U1*10;
364 U2=U2*10;
365 U3=U3*10;
366 rang=rang*10;
367
368 % plot(prod)
369
370 % x = linspace(0,180,1800);
371 %
372 % plot(x,prod)
373 if Nusuarios==2
374     L1=NormalizedAF(U1);
375 %       [~,largo]=size(AF1);
376 %       ref1=RefdB(:,1:largo);
377     %L1=sum(AF1);
378     L1=L1/(1/cant);
379   L1=10.*log10((L1).^2);
380
381
382   L2=NormalizedAF(U2);
383 %       [~,largo]=size(AF1);
```

```
%       ref1=RefdB(:,1:largo);
      %L2=sum(AF2);
      L2=L2/(1/cant);
     L2=10.*log10((L2).^2);

      L3=0;
elseif Nusuarios==3

     L1=NormalizedAF(U1);
%       [~,largo]=size(AF1);
%       ref1=RefdB(:,1:largo);
      %L1=sum(AF1);
     L1=L1/(1/cant);
     L1=10.*log10((L1).^2);


     L2=NormalizedAF(U2);
%       [~,largo]=size(AF1);
%       ref1=RefdB(:,1:largo);
      %L2=sum(AF2);
     L2=L2/(1/cant);
     L2=10.*log10((L2).^2);



        L3=NormalizedAF(U3);
%       [~,largo]=size(AF1);
%       ref1=RefdB(:,1:largo);
%      L3=sum(AF3);
      L3=L3/(1/cant);
     L3=10.*log10((L3).^2);



end

end
```

```
422
423 function [diff]=calcproduct(Pout, desiredsolution,Nusuarios,wi)
424 rad=pi/180;
425 cant=1800;
426
427 NumAntElem=size(wi,2);
428 NumAntElem=sqrt(NumAntElem);
429 antdim=[NumAntElem NumAntElem];
430 diselementos=[0.5 0.5];
431
432 thetar=[0 180]*rad;
433 THETARAD=linspace(thetar(1),thetar(2),cant);
434 ArrayFactor=faf2(exp(1i*Pout),antdim,diselementos,THETARAD,0);%
      Patron de radiacion de la particula i
435 NormalizedAF=abs(ArrayFactor)/sum(abs(ArrayFactor));
436 prod=NormalizedAF.*desiredsolution;
437 %plot(prod)
438
439 % x = linspace(0,180,1800);
440 %
441 % plot(x,prod)
442 if Nusuarios==2
443       [maxvalue1,pos1]=max(prod);
444    if pos1>100
445           posv1=[pos1-100:pos1+100];
446           prod(1,posv1) = 0;
447       else
448           posv1=[0:pos1+100];
449           prod(1,posv1+1) = 0;
450
451     end
452
453
454
455      %plot(prod)
456
457      [maxvalue2,pos2]=max(prod);
458      if pos2>100
```

```
459             posv2=[pos2-100:pos2+100];
460             prod(1,posv2) = 0;
461         else
462             posv2=[0:pos2+100];
463             prod(1,posv2+1) = 0;
464
465         end
466
467
468         %plot(prod)
469
470         diff=abs(maxvalue1-maxvalue2);
471     elseif Nusuarios==3
472
473
474
475         [maxvalue1,pos1]=max(prod);
476           if pos1>100
477             posv1=[pos1-100:pos1+100];
478             prod(1,posv1) = 0;
479         else
480             posv1=[0:pos1+100];
481             prod(1,posv1+1) = 0;
482
483       end
484
485     %       plot(prod)
486
487       [maxvalue2,pos2]=max(prod);
488        if pos2>100
489             posv2=[pos2-100:pos2+100];
490             prod(1,posv2) = 0;
491         else
492             posv2=[0:pos2+100];
493             prod(1,posv2+1) = 0;
494
495         end
496
```

```
497
498 %       plot(prod)
499
500     [maxvalue3,pos3]=max(prod);
501      if  pos3>100
502           posv3=[pos3-100:pos3+100];
503           prod(1,posv3) = 0;
504      else
505           posv3=[0:pos3+100];
506           prod(1,posv3+1) = 0;
507
508      end
509
510    diff=abs((maxvalue1-maxvalue2)+(maxvalue1-maxvalue3));
511 end
512
513
514 end
515
516
517
518
519
520
521 function[err,AtascoCount,iter]=errorcalc(BestFun, err,
       NormalizedDesiredSolution,iter,AtascoCount)
522 previouserrorA=err;
523  err=sum(abs( BestFun-norm(NormalizedDesiredSolution)));
524  if  previouserrorA==err
525       AtascoCount=AtascoCount+1;
526  else
527       AtascoCount=0;
528  end
529  iter=iter+1;
530 end
531 function[BestW,BestFun]=Compare(NewW,NewFun,
       NormalizedDesiredSolutionVector,BestW,BestFun)
532 if  (NewFun>BestFun) && (NewFun<=norm(
```

```
    NormalizedDesiredSolutionVector))
533             BestFun=NewFun;
534             BestW=NewW;
535
536         end
537 end
538 function [wi]=NewPosition(vi,wi)
539  wi=wi+vi;
540 end
541
542 function [vi]=NewVelocity(alpha,beta,inercia,numparticulas,
    NumAntElem,vi,BestW,BestFun,wi)
543         vi=inercia*vi+alpha*rand(numparticulas,NumAntElem).*(repmat(
              BestW,numparticulas,1)-wi)+beta*rand(numparticulas,
              NumAntElem).*(repmat(BestFun,numparticulas,1)-wi);
544
545 % vi=inercia*vi+alpha*rand(n,NumAntElem).*(repmat(BestW,n,1)-wi)+
    beta*rand(n,NumAntElem).*(repmat(BestFun,n,1)-wi);
546 %    wi=wi+vi;
547 %       wi3=mod(wi,2*pi);
548 end
549 function [BestW,BestF,NormalizedDesiredSolution,I,bar]=choosebest(
    solutions,NormalizedDesiredSolutionVector,wi)
550
551 [~,I] = max(solutions);
552 NormalizedDesiredSolution=NormalizedDesiredSolutionVector(I,:);
553 BestW =wi(I,:);
554 BestF=solutions(I);
555 bar=max(NormalizedDesiredSolution);
556
557 end
558
559 function[solutions,DesiredSolution,NormalizedDesiredSolutionVector
    ]=costfunction(Nusuarios,U1,U2,U3,numparticulas,wi) %funcion
    que evalua el vector de pesos y genera un diagrama de solucion
    con los usuarios.
560 global rang
561 gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo; %
```

```
    funcion gaussiana
amp=1000; %amplitud
width=rang; %ancho (para distancias cercanas es 6, distancias
    medianas 4.5, distancias lejanas 6)
cant=1800; %cantidad de puntos
desplazamiento=0; %desplazamiento de las campanas
thetar=[0 180];  %rango de theta
THETA=linspace(thetar(1),thetar(2),cant); %theta en grados
rad=pi/180; %radianes
NumAntElem=size(wi,2);
NumAntElem=sqrt(NumAntElem); %numero de elementos de la antenna
antdim=[NumAntElem NumAntElem]; %dimensiones de la antea
diselementos=[0.5 0.5]; %distancia entre elementos en la antenna
N=[NumAntElem, NumAntElem];
thetar=[0 180]*rad; %theta en radianes
THETARAD=linspace(thetar(1),thetar(2),cant);     % 1 x cant va
    desde 0 a pi en 1800 pasos
k=2*pi;
d=diselementos;
if Nusuarios==2  %caso si se tiene 2 usuarios
    usuarios=[U1 U2]; %ubicacion usuarios
    DesiredSolution = gaus(THETA,usuarios(1),width,amp,
        desplazamiento)+gaus(THETA,usuarios(2),width,amp,
        desplazamiento);%+gaus(THETA,t0(3),sig,amp,vo);
     %se genera la solucion deseada que sera utilizada como
        referencia (funcion gaussiana)
elseif Nusuarios==3
     usuarios=[U1 U2 U3] ; %ubicacion usuarios
     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
        desplazamiento)+gaus(THETA,usuarios(2),width,amp,
        desplazamiento)+gaus(THETA,usuarios(3),width,amp,
        desplazamiento);
     %se genera la solucion deseada que sera utilizada como
        referencia (funcion gaussiana)
end

Psix = (((1:N(1))-1).*(k*d(1)*sin(THETARAD).*cos(0))')';
Psiy = (((1:N(2))-1).*(k*d(2)*sin(THETARAD).*sin(0))')';
```

```
590 for i=1:N(1)
591     sumaPsi(N(1)*(i-1)+1:i*N(1),:)=Psix(i,:)+Psiy;
592 end
593 sumaPsi=exp(1i*sumaPsi); %64x1800 complex double;
594 ArrayFactor1            = exp(1i*wi)*sumaPsi;
595
596
597
598 %ArrayFactor1=faf2(exp(1i*wi),antdim,diselementos,THETARAD,0); %se
      calcula el factor de arreglo
599 NormalizedAF1=abs(ArrayFactor1); %valor absoluto del factor de
    arreglo
600  sumNAF1=sum(NormalizedAF1,2); %el valor total sumado de todos los
       elementos
601  NAF1=NormalizedAF1./sumNAF1; %Normalizacion del factor de arreglo
602  M = max(NAF1,[],2); %se encuentra el punto maximo del factor de
      arreglo normalizado
603   DesiredSolution1=repmat(DesiredSolution,numparticulas,1); %se
      genera matriz repetida de solucion deseada
604 NormalizedDesiredSolutionVector=DesiredSolution1.*M; %multiplico
    el punto mas alto por la solucion deseada(la solcucion deseada
    (desired solution) tiene una amplitud de mil y el punto maximo
    es como de 2x10-3, este producto se hace para obtener una
    solucion deseada que este en una misma escala que las
    soluciones que se van a ir propionendo)
605 prodv=NAF1.*DesiredSolution1; %multiplico la solucion deseada por
    el factor de arreglo normalizado
606 solutions=sqrt(sum(prodv.^2,2)); %calculo la norma del producto
    anterior
607
608
609 end
610
611
612 function [wi]=enrango(wi)
613 wi=mod(wi,2*pi);
614 end
615
```

```
616 function [wi]=initalize(NumAntElem,numparticulas)
617 LB=0*ones(1,NumAntElem);              % Limite inferior
618 UB=2*pi*ones(1,NumAntElem);           % Limite Superior
619 wmin=min(LB);                  % w minimo
620 wmax=max(UB);                  % w maximo
621
622
623
624 wi=(rand(numparticulas,NumAntElem))*(wmax-wmin)+wmin;
625 end
626
627 function Plotting(antdim,diselementos, BestW,
      NormalizedDesiredSolution )
628 rad=pi/180;
629 antdim=[antdim antdim];
630 diselementos=[diselementos diselementos];
631
632 thetar=[0 180]*rad;
633 THETARAD=linspace(thetar(1),thetar(2),1800);     % 1 x cant va
      desde 0 a pi en 1800 pasos
634 x = linspace(0,180,1800);
635
636
637 AF=faf2(exp(1i*BestW),antdim,diselementos,THETARAD,0);
638 AFnorma=abs(AF/sum(abs(AF)));
639 figure(9);
640 plot(x,AFnorma);
641 title('Factor de Arreglo Normalizado')
642 xlabel('Theta (grados)')
643
644 W=exp(1i*BestW);
645 Wbestpos=abs(W);                              % Magnitudes
646 betabestpos=mod(angle(W)*180/pi,360);    % Angulos
647 nombre='PSO';
648
649 radtheta(W,antdim,diselementos,0,nombre)
650 % %----------------------Patron de Radiaci  n en Theta Polares
      ----------------------
```

```
651  radthetapol(W,antdim,diselementos,0,nombre)
652
653
654
655
656 end
```

A.1.1.4. Código de bloque BAT de estrategia híbrida MAS

```
1  function [BestfunBAT, BestBATWI, wiBAT, Migration4CUCKOO,
       Migration4GA,Migration4PSO, helpCUCKOO,helpGA,helpPSO,helpBAT,
       BATerrv,errBAT,AtascoCountBAT]=BATislandBlock2(wiBAT,
       aceleracion,BATalpha,BATtheta,BATgamma,helpCUCKOO,helpPSO,
       helpGA,initialpulserate,k,errBAT,AtascoCountBAT,BestfunCUCKOO,
       BestfunPSO,BestfunGA,BestfunBAT,BATerrv)
2  global BATglobaldiff
3  global NormalizedDesiredSolution
4  global BestFun
5  global BestW
6  global NormalizedDesiredSolution
7  global BestFun
8  global BestW
9  global numparticulas
10 global tol
11 global diselementos; %Distancia entre elementos de la antenna (
       Flotante)
12 global Nusuarios;          % Numero de usuarios
13 global U1;                   %   - U1          ubicaci n usuario 1
       en grados    (entero)
14 global U2;                   %   - U2          ubicaci n usuario 2
       en grados    (entero)
15 global U3                    %   - U3          ubicaci n usuario 3
        en grados    (entero)
16 global maxatasco
17 global maxit
18 global vi
19 global BATpHistory
20 global BATFitHistory
21 global Loudness
```

```
22 global pulserate
23 global Migration4BAT
24 global Migration4PSO
25 global Migration4CUCKOO
26 global Migration4GA
27
28 global desiredsolution
29
30
31
32 empty=isempty(Migration4BAT);
33
34 if empty==0
35     wiBAT(10,:)=Migration4BAT;
36     Migration4BAT = double.empty;
37     helpBAT=0;
38     AtascoCountBAT=0;
39 end
40
41 if helpCUCKOO==1
42     if BestfunCUCKOO<BestfunBAT
43         [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
              wiBAT);
44         Bestfunlimit=BestfunBAT*0.8 ;
45         [index]=find(cost<Bestfunlimit);
46         [~,sizeindex]=size(index);
47         if sizeindex>0
48
49             Migration4CUCKOO(1,:)=wiBAT(1,:);
50 %               Migration4CUCKOO(2,:)=wiBAT(index(1),:);
51 %               Migration4CUCKOO(3,:)=wiBAT(index(3),:);
52 %               Migration4CUCKOO(4,:)=wiBAT(index(5),:);
53 %               Migration4CUCKOO(5,:)=wiBAT(index(end),:);
54         else
55             Migration4CUCKOO(1,:)=wiBAT(1,:);
56 %               Migration4CUCKOO(2,:)=wiBAT(3,:);
57 %               Migration4CUCKOO(3,:)=wiBAT(7,:);
58 %               Migration4CUCKOO(4,:)=wiBAT(15,:);
```

```
59 %                 Migration4CUCKOO(5,:)=wiBAT(21,:);
60
61             end
62             helpCUCKOO=0;
63
64
65         else
66             Migration4CUCKOO=double.empty;
67         end
68 end
69
70 if helpPSO==1
71     if BestfunPSO<BestfunBAT
72         [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
              wiBAT);
73         Bestfunlimit=BestfunBAT*0.8  ;
74         [index]=find(cost<Bestfunlimit);
75         [~,sizeindex]=size(index);
76         if sizeindex>0
77
78             Migration4PSO(1,:)=wiBAT(1,:);
79 %             Migration4PSO(2,:)=wiBAT(index(1),:);
80 %             Migration4PSO(3,:)=wiBAT(index(3),:);
81 %             Migration4PSO(4,:)=wiBAT(index(5),:);
82 %             Migration4PSO(5,:)=wiBAT(index(end),:);
83         else
84             Migration4PSO(1,:)=wiBAT(1,:);
85 %             Migration4PSO(2,:)=wiBAT(3,:);
86 %             Migration4PSO(3,:)=wiBAT(7,:);
87 %             Migration4PSO(4,:)=wiBAT(15,:);
88 %             Migration4PSO(5,:)=wiBAT(20,:);
89
90         end
91         helpPSO=0;
92
93     else
94          Migration4PSO=double.empty;
95     end
```

```
96  end
97
98  if helpGA==1
99      if BestfunGA<BestfunBAT
100         [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
                wiBAT);
101         Bestfunlimit=BestfunBAT*0.8 ;
102         [index]=find(cost<Bestfunlimit);
103         [~,sizeindex]=size(index);
104         if sizeindex>0
105
106             Migration4GA(1,:)=wiBAT(1,:);
107 %               Migration4GA(2,:)=wiBAT(index(1),:);
108 %               Migration4GA(3,:)=wiBAT(index(3),:);
109 %               Migration4GA(4,:)=wiBAT(index(5),:);
110 %               Migration4GA(5,:)=wiBAT(index(end),:);
111         else
112             Migration4GA(1,:)=wiBAT(1,:);
113 %               Migration4GA(2,:)=wiBAT(3,:);
114 %               Migration4GA(3,:)=wiBAT(7,:);
115 %               Migration4GA(4,:)=wiBAT(15,:);
116 %               Migration4GA(5,:)=wiBAT(20,:);
117
118         end
119
120         helpGA=0;
121
122     else
123         Migration4GA=double.empty;
124     end
125 end
126
127
128 [pout] = SimpleBAT(wiBAT,aceleracion,BATalpha,BATtheta,BestW,
        BestFun,Nusuarios,U1,U2,U3,initialpulserate,k,BATgamma);
129
130 [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,pout); %
        evalua el vector de pesos wi en la funcion de costos
```

```
131 [cost2,ind] = sort(cost,'descend'); % sorting
132           pout= pout(ind,:);
133
134          pout(end,:) = BATpHistory(k-1,:); %The former best
                sustitute now the current worst
135
136          [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
                pout); % Sort again to keep the current best
137          [cost,ind] = sort(cost,'descend');
138          pout= pout(ind,:);
139
140          localsolution=cost(1);
141
142          Localbest=pout(1,:);
143 %
144 %           if Nusuarios==1
145 %               if localsolution>BestfunBAT && localsolution<=norm(
    NormalizedDesiredSolution)
146 %                   BATFitHistory = [BATFitHistory localsolution] ;
147 %                   BATpHistory = [BATpHistory;Localbest];
148 %
149 %               else
150 %                   BATFitHistory = [BATFitHistory BATFitHistory(end
    )] ;
151 %                   BATpHistory = [BATpHistory;BATpHistory(end,:)];
152 %
153 %
154 %               end
155 %
156 %
157 %           else
158              [diff]=calcproduct(Localbest, desiredsolution,Nusuarios
                    ,pout);
159
160              fitness=localsolution/diff;
161          if fitness>BestfunBAT %&& diff<=BATglobaldiff &&
                localsolution<=norm(NormalizedDesiredSolution)
162               BATFitHistory = [BATFitHistory fitness] ;
```

```
163             BATpHistory = [BATpHistory;Localbest];
164             BATglobaldiff=diff;
165         else
166             BATFitHistory = [BATFitHistory BATFitHistory(end)] ;
167             BATpHistory = [BATpHistory;BATpHistory(end,:)];
168
169
170         end
171
172
173
174          BestfunBAT=BATFitHistory(end);
175         BestBATWI=BATpHistory(end,:);
176         [errBAT,AtascoCountBAT,~,helpBAT]=SimpleEndIsland(
                BestfunBAT, errBAT, NormalizedDesiredSolution,k,
                AtascoCountBAT,maxit,maxatasco);
177         BATerrv(k)=errBAT;
178         wiBAT=pout;
179
180
181 end
182
183 function[solutions,DesiredSolution,NormalizedDesiredSolutionVector
    ]=costfunction(Nusuarios,U1,U2,U3,numparticulas,wi) %funcion
    que evalua el vector de pesos y genera un diagrama de solucion
    con los usuarios.
184 global rang
185 gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo; %
    funcion gaussiana
186 amp=1000; %amplitud
187 width=rang; %ancho (para distancias cercanas es 6, distancias
    medianas 4.5, distancias lejanas 6)
188 cant=1800; %cantidad de puntos
189 desplazamiento=0; %desplazamiento de las campanas
190 thetar=[0 180];  %rango de theta
191 THETA=linspace(thetar(1),thetar(2),cant); %theta en grados
192 rad=pi/180; %radianes
193 NumAntElem=size(wi,2);
```

```
194 NumAntElem=sqrt(NumAntElem); %numero de elementos de la antenna
195 antdim=[NumAntElem NumAntElem]; %dimensiones de la antea
196 diselementos=[0.5 0.5]; %distancia entre elementos en la antenna
197 N=[NumAntElem, NumAntElem];
198 thetar=[0 180]*rad; %theta en radianes
199 THETARAD=linspace(thetar(1),thetar(2),cant);     % 1 x cant va
    desde 0 a pi en 1800 pasos
200 k=2*pi;
201 d=diselementos;
202 if Nusuarios==2  %caso si se tiene 2 usuarios
203     usuarios=[U1 U2]; %ubicacion usuarios
204     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
        desplazamiento)+gaus(THETA,usuarios(2),width,amp,
        desplazamiento);%+gaus(THETA,t0(3),sig,amp,vo);
205     %se genera la solucion deseada que sera utilizada como
        referencia (funcion gaussiana)
206 elseif Nusuarios==3
207     usuarios=[U1 U2 U3] ; %ubicacion usuarios
208     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
        desplazamiento)+gaus(THETA,usuarios(2),width,amp,
        desplazamiento)+gaus(THETA,usuarios(3),width,amp,
        desplazamiento);
209     %se genera la solucion deseada que sera utilizada como
        referencia (funcion gaussiana)
210 end
211
212 Psix = (((1:N(1))-1).*(k*d(1)*sin(THETARAD).*cos(0))');
213 Psiy = (((1:N(2))-1).*(k*d(2)*sin(THETARAD).*sin(0))');
214 for i=1:N(1)
215     sumaPsi(N(1)*(i-1)+1:i*N(1),:)=Psix(i,:)+Psiy;
216 end
217 sumaPsi=exp(1i*sumaPsi); %64x1800 complex double;
218 ArrayFactor1            = exp(1i*wi)*sumaPsi;
219
220
221
222 %ArrayFactor1=faf2(exp(1i*wi),antdim,diselementos,THETARAD,0); %se
    calcula el factor de arreglo
```

```
223 NormalizedAF1=abs(ArrayFactor1); %valor absoluto del factor de
    arreglo
224  sumNAF1=sum(NormalizedAF1,2); %el valor total sumado de todos los
     elementos
225  NAF1=NormalizedAF1./sumNAF1; %Normalizacion del factor de arreglo
226  M = max(NAF1,[],2); %se encuentra el punto maximo del factor de
     arreglo normalizado
227  DesiredSolution1=repmat(DesiredSolution,numparticulas,1); %se
     genera matriz repetida de solucion deseada
228 NormalizedDesiredSolutionVector=DesiredSolution1.*M; %multiplico
    el punto mas alto por la solucion deseada(la solcucion deseada
    (desired solution) tiene una amplitud de mil y el punto maximo
    es como de 2x10-3, este producto se hace para obtener una
    solucion deseada que este en una misma escala que las
    soluciones que se van a ir propionendo)
229 prodv=NAF1.*DesiredSolution1; %multiplico la solucion deseada por
    el factor de arreglo normalizado
230 solutions=sqrt(sum(prodv.^2,2)); %calculo la norma del producto
    anterior
231
232
233 end
234
235
236
237 function[diff]=calcproduct(Pout, desiredsolution,Nusuarios,wi)
238 rad=pi/180;
239 cant=1800;
240
241 NumAntElem=size(wi,2);
242 NumAntElem=sqrt(NumAntElem);
243 antdim=[NumAntElem NumAntElem];
244 diselementos=[0.5 0.5];
245
246 thetar=[0 180]*rad;
247 THETARAD=linspace(thetar(1),thetar(2),cant);
248 ArrayFactor=faf2(exp(1i*Pout),antdim,diselementos,THETARAD,0);%
    Patron de radiacion de la particula i
```

```
249 NormalizedAF=abs(ArrayFactor)/sum(abs(ArrayFactor));
250 prod=NormalizedAF.*desiredsolution;
251 %plot(prod)
252
253 % x = linspace(0,180,1800);
254 %
255 % plot(x,prod)
256 if Nusuarios==2
257     [maxvalue1,pos1]=max(prod);
258   if pos1>100
259         posv1=[pos1-100:pos1+100];
260         prod(1,posv1) = 0;
261     else
262         posv1=[0:pos1+100];
263         prod(1,posv1+1) = 0;
264
265    end
266
267
268
269     %plot(prod)
270
271     [maxvalue2,pos2]=max(prod);
272     if pos2>100
273         posv2=[pos2-100:pos2+100];
274         prod(1,posv2) = 0;
275     else
276         posv2=[0:pos2+100];
277         prod(1,posv2+1) = 0;
278
279     end
280
281
282     %plot(prod)
283
284     diff=abs(maxvalue1-maxvalue2);
285 elseif Nusuarios==3
286
```

```
287
288
289       [maxvalue1,pos1]=max(prod);
290         if  pos1>100
291           posv1=[pos1-100:pos1+100];
292           prod(1,posv1) = 0;
293       else
294           posv1=[0:pos1+100];
295           prod(1,posv1+1) = 0;
296
297      end
298
299  %       plot(prod)
300
301      [maxvalue2,pos2]=max(prod);
302       if  pos2>100
303           posv2=[pos2-100:pos2+100];
304           prod(1,posv2) = 0;
305       else
306           posv2=[0:pos2+100];
307           prod(1,posv2+1) = 0;
308
309       end
310
311
312  %       plot(prod)
313
314      [maxvalue3,pos3]=max(prod);
315       if  pos3>100
316           posv3=[pos3-100:pos3+100];
317           prod(1,posv3) = 0;
318       else
319           posv3=[0:pos3+100];
320           prod(1,posv3+1) = 0;
321
322       end
323
324     diff=abs((maxvalue1-maxvalue2)+(maxvalue1-maxvalue3));
```

```
325 end
326
327
328 end
```

A.1.1.5. Código de bloque GA de estrategia híbrida MAS

```
1  function [BestfunGA, BestGAWI,wiGA, Migration4CUCKOO,Migration4PSO
       ,Migration4BAT, helpCUCKOO,helpGA,helpPSO,helpBAT,GAerrv,errGA,
       AtascoCountGA]=GAislandBlock2(wiGA,pc,pm,mutstep,helpCUCKOO,
       helpBAT,helpPSO,k,AtascoCountGA,errGA,GAerrv,BestfunCUCKOO,
       BestfunBAT,BestfunGA,BestfunPSO)
2  global GAglobaldiff
3  global NormalizedDesiredSolution
4  global BestFun
5  global NormalizedDesiredSolution
6  global BestFun
7  global BestW
8  global numparticulas
9  global Nusuarios;          % Numero de usuarios
10 global U1;                   %   - U1          ubicaci n usuario 1
     en grados   (entero)
11 global U2;                   %   - U2          ubicaci n usuario 2
     en grados   (entero)
12 global U3                    %   - U3          ubicaci n usuario 3
      en grados   (entero)
13 global maxatasco
14 global maxit
15 global vi
16 global GApHistory
17 global GAFitHistory
18 global Migration4BAT
19 global Migration4PSO
20 global Migration4CUCKOO
21 global Migration4GA
22 global desiredsolution
23
24 empty=isempty(Migration4GA);
25 if empty==0
```

```
26      wiGA(10,:)=Migration4GA;
27      Migration4GA = double.empty;
28      helpGA=0;
29      AtascoCountGA=0;
30  end
31
32  if helpCUCKOO==1
33      if BestfunCUCKOO<BestfunGA
34          [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
                wiGA);
35          Bestfunlimit=BestfunGA*0.8 ;
36          [index]=find(cost<Bestfunlimit);
37          [~,sizeindex]=size(index);
38          if sizeindex>4
39              Migration4CUCKOO(1,:)=wiGA(1,:);
40  %               Migration4CUCKOO(2,:)=wiGA(index(1),:);
41  %               Migration4CUCKOO(3,:)=wiGA(index(3),:);
42  %               Migration4CUCKOO(4,:)=wiGA(index(5),:);
43  %               Migration4CUCKOO(5,:)=wiGA(index(end),:);
44          else
45              Migration4CUCKOO(1,:)=wiGA(1,:);
46  %               Migration4CUCKOO(2,:)=wiGA(3,:);
47  %               Migration4CUCKOO(3,:)=wiGA(7,:);
48  %               Migration4CUCKOO(4,:)=wiGA(15,:);
49  %               Migration4CUCKOO(5,:)=wiGA(20,:);
50          end
51
52
53          helpCUCKOO=0;
54
55      else
56          Migration4CUCKOO=double.empty;
57      end
58  end
59
60  if helpBAT==1
61      if BestfunBAT<BestfunGA
62          [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
```

```
            wiGA);
63          Bestfunlimit=BestfunGA*0.8  ;
64          [index]=find(cost<Bestfunlimit);
65          [~,sizeindex]=size(index);
66          if  sizeindex>4
67
68
69              Migration4BAT(1,:)=wiGA(1,:);
70  %               Migration4BAT(2,:)=wiGA(index(1),:);
71  %               Migration4BAT(3,:)=wiGA(index(3),:);
72  %               Migration4BAT(4,:)=wiGA(index(5),:);
73  %               Migration4BAT(5,:)=wiGA(index(end),:);
74
75          else
76              Migration4BAT(1,:)=wiGA(1,:);
77  %               Migration4BAT(2,:)=wiGA(3,:);
78  %               Migration4BAT(3,:)=wiGA(7,:);
79  %               Migration4BAT(4,:)=wiGA(15,:);
80  %               Migration4BAT(5,:)=wiGA(20,:);
81
82          end
83
84
85          helpBAT=0;
86      else
87          Migration4BAT=double.empty;
88
89      end
90
91  end
92
93  if  helpPSO==1
94      if  BestfunPSO<BestfunGA
95          [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
              wiGA);
96          Bestfunlimit=BestfunGA*0.8  ;
97          [index]=find(cost<Bestfunlimit);
98          [~,sizeindex]=size(index);
```

```
99          if sizeindex>4
100
101             Migration4PSO(1,:)=wiGA(1,:);
102 %             Migration4PSO(2,:)=wiGA(index(1),:);
103 %             Migration4PSO(3,:)=wiGA(index(3),:);
104 %             Migration4PSO(4,:)=wiGA(index(5),:);
105 %             Migration4PSO(5,:)=wiGA(index(end),:);
106         else
107             Migration4PSO(1,:)=wiGA(1,:);
108 %             Migration4PSO(2,:)=wiGA(3,:);
109 %             Migration4PSO(3,:)=wiGA(7,:);
110 %             Migration4PSO(4,:)=wiGA(15,:);
111 %             Migration4PSO(5,:)=wiGA(20,:);
112
113         end
114
115
116             helpPSO=0;
117
118         else
119             Migration4PSO=double.empty;
120
121         end
122
123 end
124
125
126
127 pout = SimpleGA(wiGA,pc,pm,mutstep,Nusuarios,U1,U2,U3);
128 [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,pout); %
       evalua el vector de pesos wi en la funcion de costos
129 [cost2,ind] = sort(cost,'descend'); % sorting
130              pout= pout(ind,:);
131
132             pout(end,:) = GApHistory(k-1,:); %The former best
                   sustitute now the current worst
133
134             [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
```

```
            pout); % Sort again to keep the current best
135         [cost,ind] = sort(cost,'descend');
136         pout= pout(ind,:);
137
138         Localbest=pout(1,:);
139
140         localsolution=cost(1);
141             [diff]=calcproduct(Localbest, desiredsolution,
                    Nusuarios,pout);
142             fitness=localsolution/diff;
143
144
145         if fitness>BestfunGA %&& diff<=GAglobaldiff &&
                localsolution<=norm(NormalizedDesiredSolution)
146             GAFitHistory = [GAFitHistory fitness] ;
147             GApHistory = [GApHistory;Localbest];
148             GAglobaldiff=diff;
149         else
150             GAFitHistory = [GAFitHistory GAFitHistory(end)] ;
151             GApHistory = [GApHistory;GApHistory(end,:)];
152
153
154         end
155
156
157
158 %           ee=isempty(find(diff>GAbar));
159 %
160 %           i=find(cost<=norm(NormalizedDesiredSolution));
161 %           select=i(1);
162 %
163 %           if (ee==1)
164 %               GAFitHistory = [GAFitHistory cost(select)] ;
165 %               GApHistory = [GApHistory;pout(select,:)];
166 %               GAbar=0.95*GAbar;
167 %           else
168 %               GAFitHistory = [GAFitHistory GAFitHistory(end)] ;
169 %               GApHistory = [GApHistory;GApHistory(end,:)];
```

```
%           end
            BestfunGA=GAFitHistory(end);
            BestGAWI=GApHistory(end,:);
            [errGA,AtascoCountGA,~,helpGA]=SimpleEndIsland(BestfunGA,
                errGA, NormalizedDesiredSolution,k,AtascoCountGA,maxit,
                maxatasco);
            GAerrv(k)=errGA;
            wiGA=pout;

end


function[solutions,DesiredSolution,NormalizedDesiredSolutionVector
    ]=costfunction(Nusuarios,U1,U2,U3,numparticulas,wi) %funcion
    que evalua el vector de pesos y genera un diagrama de solucion
    con los usuarios.
global rang
gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo; %
    funcion gaussiana
amp=1000; %amplitud
width=rang; %ancho (para distancias cercanas es 6, distancias
    medianas 4.5, distancias lejanas 6)
cant=1800; %cantidad de puntos
desplazamiento=0; %desplazamiento de las campanas
thetar=[0 180];  %rango de theta
THETA=linspace(thetar(1),thetar(2),cant); %theta en grados
rad=pi/180; %radianes
NumAntElem=size(wi,2);
NumAntElem=sqrt(NumAntElem); %numero de elementos de la antenna
antdim=[NumAntElem NumAntElem]; %dimensiones de la antea
diselementos=[0.5 0.5]; %distancia entre elementos en la antenna
N=[NumAntElem, NumAntElem];
thetar=[0 180]*rad; %theta en radianes
THETARAD=linspace(thetar(1),thetar(2),cant);     % 1 x cant va
    desde 0 a pi en 1800 pasos
k=2*pi;
d=diselementos;
if Nusuarios==2  %caso si se tiene 2 usuarios
```

```
200     usuarios=[U1 U2]; %ubicacion usuarios
201     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
           desplazamiento)+gaus(THETA,usuarios(2),width,amp,
           desplazamiento);%+gaus(THETA,t0(3),sig,amp,vo);
202      %se genera la solucion deseada que sera utilizada como
           referencia (funcion gaussiana)
203 elseif Nusuarios==3
204      usuarios=[U1 U2 U3] ; %ubicacion usuarios
205      DesiredSolution = gaus(THETA,usuarios(1),width,amp,
           desplazamiento)+gaus(THETA,usuarios(2),width,amp,
           desplazamiento)+gaus(THETA,usuarios(3),width,amp,
           desplazamiento);
206      %se genera la solucion deseada que sera utilizada como
           referencia (funcion gaussiana)
207 end
208
209 Psix = (((1:N(1))-1).*(k*d(1)*sin(THETARAD).*cos(0))');
210 Psiy = (((1:N(2))-1).*(k*d(2)*sin(THETARAD).*sin(0))');
211 for i=1:N(1)
212      sumaPsi(N(1)*(i-1)+1:i*N(1),:)=Psix(i,:)+Psiy;
213 end
214 sumaPsi=exp(1i*sumaPsi); %64x1800 complex double;
215 ArrayFactor1            = exp(1i*wi)*sumaPsi;
216
217
218
219 %ArrayFactor1=faf2(exp(1i*wi),antdim,diselementos,THETARAD,0); %se
       calcula el factor de arreglo
220 NormalizedAF1=abs(ArrayFactor1); %valor absoluto del factor de
      arreglo
221  sumNAF1=sum(NormalizedAF1,2); %el valor total sumado de todos los
       elementos
222  NAF1=NormalizedAF1./sumNAF1; %Normalizacion del factor de arreglo
223  M = max(NAF1,[],2); %se encuentra el punto maximo del factor de
       arreglo normalizado
224   DesiredSolution1=repmat(DesiredSolution,numparticulas,1); %se
       genera matriz repetida de solucion deseada
225 NormalizedDesiredSolutionVector=DesiredSolution1.*M; %multiplico
```

```
    el punto mas alto por la solucion deseada(la solcucion deseada
    (desired solution) tiene una amplitud de mil y el punto maximo
    es como de 2x10-3, este producto se hace para obtener una
    solucion deseada que este en una misma escala que las
    soluciones que se van a ir propionendo)
prodv=NAF1.*DesiredSolution1; %multiplico la solucion deseada por
    el factor de arreglo normalizado
solutions=sqrt(sum(prodv.^2,2)); %calculo la norma del producto
    anterior


end



function[diff]=calcproduct(Pout, desiredsolution,Nusuarios,wi)
rad=pi/180;
cant=1800;

NumAntElem=size(wi,2);
NumAntElem=sqrt(NumAntElem);
antdim=[NumAntElem NumAntElem];
diselementos=[0.5 0.5];

thetar=[0 180]*rad;
THETARAD=linspace(thetar(1),thetar(2),cant);
ArrayFactor=faf(exp(1i*Pout),antdim,diselementos,THETARAD,0);%
    Patron de radiacion de la particula i
NormalizedAF=abs(ArrayFactor)/sum(abs(ArrayFactor));
prod=NormalizedAF.*desiredsolution;
%plot(prod)

% x = linspace(0,180,1800);
%
% plot(x,prod)
if Nusuarios==2
     [maxvalue1,pos1]=max(prod);
  if pos1>100
```

```
256             posv1=[pos1-100:pos1+100];
257             prod(1,posv1) = 0;
258         else
259             posv1=[0:pos1+100];
260             prod(1,posv1+1) = 0;
261
262         end
263
264
265
266         %plot(prod)
267
268         [maxvalue2,pos2]=max(prod);
269         if pos2>100
270             posv2=[pos2-100:pos2+100];
271             prod(1,posv2) = 0;
272         else
273             posv2=[0:pos2+100];
274             prod(1,posv2+1) = 0;
275
276         end
277
278
279         %plot(prod)
280
281         diff=abs(maxvalue1-maxvalue2);
282     elseif Nusuarios==3
283
284
285
286         [maxvalue1,pos1]=max(prod);
287           if pos1>100
288             posv1=[pos1-100:pos1+100];
289             prod(1,posv1) = 0;
290         else
291             posv1=[0:pos1+100];
292             prod(1,posv1+1) = 0;
293
```

```
294     end
295
296 %       plot(prod)
297
298     [maxvalue2,pos2]=max(prod);
299      if pos2>100
300           posv2=[pos2-100:pos2+100];
301           prod(1,posv2) = 0;
302      else
303           posv2=[0:pos2+100];
304           prod(1,posv2+1) = 0;
305
306      end
307
308
309 %       plot(prod)
310
311     [maxvalue3,pos3]=max(prod);
312      if pos3>100
313           posv3=[pos3-100:pos3+100];
314           prod(1,posv3) = 0;
315      else
316           posv3=[0:pos3+100];
317           prod(1,posv3+1) = 0;
318
319      end
320
321    diff=abs((maxvalue1-maxvalue2)+(maxvalue1-maxvalue3));
322 end
323
324
325 end
```

A.1.1.6. Código de bloque PSO de estrategia híbrida MAS

```
1 function [BestfunPSO, BestPSOWI,wiPSO, Migration4CUCKOO,
    Migration4GA,Migration4BAT, helpCUCKOO,helpGA,helpPSO,helpBAT,
    PSOerrv,errPSO,AtascoCountPSO]=PSOislandBlock2(wiPSO,helpCUCKOO
    ,helpBAT,helpGA,k,AtascoCountPSO,errPSO,PSOerrv,BestfunCUCKOO,
```

```
    BestfunBAT, BestfunGA, BestfunPSO)
global PSOglobaldiff
global PSOalpha
global PSObeta
global PSOinercia
global NormalizedDesiredSolution
global BestFun
global BestW
global NormalizedDesiredSolution
global BestFun
global BestW
global numparticulas
global tol
global diselementos; %Distancia entre elementos de la antenna (
    Flotante)
global Nusuarios;          % Numero de usuarios
global U1;                   %   - U1          ubicaci n usuario 1
    en grados   (entero)
global U2;                   %   - U2          ubicaci n usuario 2
    en grados   (entero)
global U3                     %   - U3          ubicaci n usuario 3
     en grados   (entero)
global maxatasco
global maxit
global vi
global PSOpHistory
global PSOFitHistory
global Migration4BAT
global Migration4PSO
global Migration4CUCKOO
global Migration4GA
global Nusuarios
global desiredsolution
global BestWhist


empty=isempty(Migration4PSO);

```

```
35 if empty==0
36     wiPSO(10,:)=Migration4PSO;
37     Migration4PSO = double.empty      ;
38     helpPSO=0;
39     AtascoCountPSO=0;
40 end
41
42 if helpCUCKOO==1
43     if BestfunCUCKOO<BestfunPSO
44         [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
               wiPSO);
45         Bestfunlimit=BestfunPSO*0.8  ;
46         [index]=find(cost<Bestfunlimit);
47         [~,sizeindex]=size(index);
48         if sizeindex>4
49             Migration4CUCKOO(1,:)=wiPSO(1,:);
50 %               Migration4CUCKOO(2,:)=wiPSO(index(1),:);
51 %               Migration4CUCKOO(3,:)=wiPSO(index(3),:);
52 %               Migration4CUCKOO(4,:)=wiPSO(index(5),:);
53 %               Migration4CUCKOO(5,:)=wiPSO(index(end),:);
54         else
55             Migration4CUCKOO(1,:)=wiPSO(1,:);
56 %               Migration4CUCKOO(2,:)=wiPSO(3,:);
57 %               Migration4CUCKOO(3,:)=wiPSO(5,:);
58 %               Migration4CUCKOO(4,:)=wiPSO(15,:);
59 %               Migration4CUCKOO(5,:)=wiPSO(20,:);
60
61         end
62
63         helpCUCKOO=0;
64
65
66     else
67         Migration4CUCKOO=double.empty;
68     end
69 end
70
71 if helpBAT==1
```

```
72      if BestfunBAT<BestfunPSO
73          [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
                wiPSO);
74          Bestfunlimit=BestfunPSO*0.8  ;
75          [index]=find(cost<Bestfunlimit);
76
77
78          [~,sizeindex]=size(index);
79          if sizeindex>4
80              Migration4BAT(1,:)=wiPSO(1,:);
81  %               Migration4BAT(2,:)=wiPSO(index(1),:);
82  %               Migration4BAT(3,:)=wiPSO(index(3),:);
83  %               Migration4BAT(4,:)=wiPSO(index(5),:);
84  %               Migration4BAT(5,:)=wiPSO(index(end),:);
85          else
86              Migration4BAT(1,:)=wiPSO(1,:);
87  %               Migration4BAT(2,:)=wiPSO(3,:);
88  %               Migration4BAT(3,:)=wiPSO(7,:);
89  %               Migration4BAT(4,:)=wiPSO(15,:);
90  %               Migration4BAT(5,:)=wiPSO(20,:);
91
92          end
93
94          helpBAT=0;
95
96
97      else
98          Migration4BAT=double.empty;
99      end
100
101 end
102
103 if helpGA==1
104     if BestfunGA<BestfunPSO
105         [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
                wiPSO);
106         Bestfunlimit=BestfunPSO*0.8  ;
107         [index]=find(cost<Bestfunlimit);
```

```
108             [~,sizeindex]=size(index);
109             if sizeindex>4
110                 Migration4GA(1,:)=wiPSO(1,:);
111 %                 Migration4GA(2,:)=wiPSO(index(1),:);
112 %                 Migration4GA(3,:)=wiPSO(index(3),:);
113 %                 Migration4GA(4,:)=wiPSO(index(5),:);
114 %                 Migration4GA(5,:)=wiPSO(index(end),:);
115             else
116                 Migration4GA(1,:)=wiPSO(1,:);
117 %                 Migration4GA(2,:)=wiPSO(3,:);
118 %                 Migration4GA(3,:)=wiPSO(7,:);
119 %                 Migration4GA(4,:)=wiPSO(15,:);
120 %                 Migration4GA(5,:)=wiPSO(20,:);
121 %
122             end
123                 helpGA=0;
124
125         else
126             Migration4GA=double.empty;
127         end
128
129 end
130
131
132
133 pout = SimplePSO(wiPSO,PSOalpha,PSObeta,PSOinercia,BestW,BestFun,
        BestWhist);
134 [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,pout); %
        evalua el vector de pesos wi en la funcion de costos
135 [cost2,ind] = sort(cost,'descend'); % sorting
136              pout= pout(ind,:);
137
138             pout(end,:) = PSOpHistory(k-1,:); %The former best
                sustitute now the current worst
139
140             [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
                pout); % Sort again to keep the current best
141             [cost,ind] = sort(cost,'descend');
```

```
142         pout= pout(ind,:);
143
144         Localbest=pout(1,:);
145         localsolution=cost(1);
146         [diff]=calcproduct(Localbest, desiredsolution,Nusuarios,
                pout);
147
148         fitness=localsolution/diff;
149
150 %           if Nusuarios==1
151 %               if localsolution>BestfunPSO && localsolution<=norm(
    NormalizedDesiredSolution)
152 %               PSOFitHistory = [PSOFitHistory localsolution] ;
153 %               PSOpHistory = [PSOpHistory;Localbest];
154 %
155 %               else
156 %                   PSOFitHistory = [PSOFitHistory PSOFitHistory(end
    )] ;
157 %                   PSOpHistory = [PSOpHistory;PSOpHistory(end,:)];
158 %
159 %               end
160 %
161 %           else
162
163
164
165
166
167
168         if fitness>BestfunPSO %&& diff<=PSOglobaldiff &&
               localsolution<=norm(NormalizedDesiredSolution)
169             PSOFitHistory = [PSOFitHistory fitness] ;
170             PSOpHistory = [PSOpHistory;Localbest];
171             PSOglobaldiff=diff;
172         else
173             PSOFitHistory = [PSOFitHistory PSOFitHistory(end)] ;
174             PSOpHistory = [PSOpHistory;PSOpHistory(end,:)];
175         end
```

```
176
177
178             BestfunPSO=PSOFitHistory(end);
179             BestPSOWI=PSOpHistory(end,:);
180             [errPSO,AtascoCountPSO,~,helpPSO]=SimpleEndIsland(
                    BestfunPSO, errPSO, NormalizedDesiredSolution,k,
                    AtascoCountPSO,maxit,maxatasco);
181             PSOerrv(k)=errPSO;
182             wiPSO=pout;
183
184
185 end
186
187 function[solutions,DesiredSolution,NormalizedDesiredSolutionVector
    ]=costfunction(Nusuarios,U1,U2,U3,numparticulas,wi) %funcion
    que evalua el vector de pesos y genera un diagrama de solucion
    con los usuarios.
188 global rang
189 gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo; %
    funcion gaussiana
190 amp=1000; %amplitud
191 width=rang; %ancho (para distancias cercanas es 6, distancias
    medianas 4.5, distancias lejanas 6)
192 cant=1800; %cantidad de puntos
193 desplazamiento=0; %desplazamiento de las campanas
194 thetar=[0 180];  %rango de theta
195 THETA=linspace(thetar(1),thetar(2),cant); %theta en grados
196 rad=pi/180; %radianes
197 NumAntElem=size(wi,2);
198 NumAntElem=sqrt(NumAntElem); %numero de elementos de la antenna
199 antdim=[NumAntElem NumAntElem]; %dimensiones de la antea
200 diselementos=[0.5 0.5]; %distancia entre elementos en la antenna
201 N=[NumAntElem, NumAntElem];
202 thetar=[0 180]*rad; %theta en radianes
203 THETARAD=linspace(thetar(1),thetar(2),cant);     % 1 x cant va
    desde 0 a pi en 1800 pasos
204 k=2*pi;
205 d=diselementos;
```

```
206 if Nusuarios==2  %caso si se tiene 2 usuarios
207     usuarios=[U1 U2]; %ubicacion usuarios
208     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
            desplazamiento)+gaus(THETA,usuarios(2),width,amp,
            desplazamiento);%+gaus(THETA,t0(3),sig,amp,vo);
209      %se genera la solucion deseada que sera utilizada como
            referencia (funcion gaussiana)
210 elseif Nusuarios==3
211      usuarios=[U1 U2 U3] ; %ubicacion usuarios
212      DesiredSolution = gaus(THETA,usuarios(1),width,amp,
            desplazamiento)+gaus(THETA,usuarios(2),width,amp,
            desplazamiento)+gaus(THETA,usuarios(3),width,amp,
            desplazamiento);
213      %se genera la solucion deseada que sera utilizada como
            referencia (funcion gaussiana)
214 end
215
216 Psix = (((1:N(1))-1).*(k*d(1)*sin(THETARAD).*cos(0))');';
217 Psiy = (((1:N(2))-1).*(k*d(2)*sin(THETARAD).*sin(0))');';
218 for i=1:N(1)
219     sumaPsi(N(1)*(i-1)+1:i*N(1),:)=Psix(i,:)+Psiy;
220 end
221 sumaPsi=exp(1i*sumaPsi); %64x1800 complex double;
222 ArrayFactor1            = exp(1i*wi)*sumaPsi;
223
224
225
226 %ArrayFactor1=faf2(exp(1i*wi),antdim,diselementos,THETARAD,0); %se
      calcula el factor de arreglo
227 NormalizedAF1=abs(ArrayFactor1); %valor absoluto del factor de
    arreglo
228  sumNAF1=sum(NormalizedAF1,2); %el valor total sumado de todos los
       elementos
229  NAF1=NormalizedAF1./sumNAF1; %Normalizacion del factor de arreglo
230  M = max(NAF1,[],2); %se encuentra el punto maximo del factor de
      arreglo normalizado
231   DesiredSolution1=repmat(DesiredSolution,numparticulas,1); %se
      genera matriz repetida de solucion deseada
```

```
232 NormalizedDesiredSolutionVector=DesiredSolution1.*M; %multiplico
    el punto mas alto por la solucion deseada(la solcucion deseada
    (desired solution) tiene una amplitud de mil y el punto maximo
    es como de 2x10-3, este producto se hace para obtener una
    solucion deseada que este en una misma escala que las
    soluciones que se van a ir propionendo)
233 prodv=NAF1.*DesiredSolution1; %multiplico la solucion deseada por
    el factor de arreglo normalizado
234 solutions=sqrt(sum(prodv.^2,2)); %calculo la norma del producto
    anterior
235
236
237 end
238
239
240
241 function[diff]=calcproduct(Pout, desiredsolution,Nusuarios,wi)
242 rad=pi/180;
243 cant=1800;
244
245 NumAntElem=size(wi,2);
246 NumAntElem=sqrt(NumAntElem);
247 antdim=[NumAntElem NumAntElem];
248 diselementos=[0.5 0.5];
249
250 thetar=[0 180]*rad;
251 THETARAD=linspace(thetar(1),thetar(2),cant);
252 ArrayFactor=faf(exp(1i*Pout),antdim,diselementos,THETARAD,0);%
    Patron de radiacion de la particula i
253 NormalizedAF=abs(ArrayFactor)/sum(abs(ArrayFactor));
254 prod=NormalizedAF.*desiredsolution;
255 %plot(prod)
256
257 % x = linspace(0,180,1800);
258 %
259 % plot(x,prod)
260 if Nusuarios==2
261     [maxvalue1,pos1]=max(prod);
```

```
262   if  pos1>100
263         posv1=[pos1-100:pos1+100];
264         prod(1,posv1) = 0;
265     else
266         posv1=[0:pos1+100];
267         prod(1,posv1+1) = 0;
268
269    end
270
271
272
273     %plot(prod)
274
275     [maxvalue2,pos2]=max(prod);
276     if  pos2>100
277         posv2=[pos2-100:pos2+100];
278         prod(1,posv2) = 0;
279     else
280         posv2=[0:pos2+100];
281         prod(1,posv2+1) = 0;
282
283     end
284
285
286     %plot(prod)
287
288     diff=abs(maxvalue1-maxvalue2);
289 elseif  Nusuarios==3
290
291
292
293     [maxvalue1,pos1]=max(prod);
294       if  pos1>100
295         posv1=[pos1-100:pos1+100];
296         prod(1,posv1) = 0;
297     else
298         posv1=[0:pos1+100];
299         prod(1,posv1+1) = 0;
```

```
300
301     end
302
303 %       plot(prod)
304
305     [maxvalue2,pos2]=max(prod);
306      if  pos2>100
307           posv2=[pos2-100:pos2+100];
308           prod(1,posv2) = 0;
309      else
310           posv2=[0:pos2+100];
311           prod(1,posv2+1) = 0;
312
313      end
314
315
316 %       plot(prod)
317
318     [maxvalue3,pos3]=max(prod);
319      if  pos3>100
320           posv3=[pos3-100:pos3+100];
321           prod(1,posv3) = 0;
322      else
323           posv3=[0:pos3+100];
324           prod(1,posv3+1) = 0;
325
326      end
327
328    diff=abs((maxvalue1-maxvalue2)+(maxvalue1-maxvalue3));
329 end
330
331
332 end
```

A.1.1.7. Código de bloque CUCKOO de estrategia híbrida MAS

```
1 function [BestfunCUCKOO, BestCUCKOOWI,wiCUCKOO, Migration4PSO,
    Migration4GA,Migration4BAT, helpCUCKOO,helpGA,helpPSO,helpBAT,
    CUCKOOerrv,errCUCKOO,AtascoCountCUCKOO]=CUCKOOislandBlock2(
```

```
    wiCUCKOO, pa , CUCKOObeta, helpPSO , helpBAT , helpGA , k ,
    AtascoCountCUCKOO , errCUCKOO , CUCKOOerrv , BestfunCUCKOO , BestfunBAT
    , BestfunGA , BestfunPSO )
global CUCKOOglobaldiff
global NormalizedDesiredSolution
global BestFun
global BestW
global NormalizedDesiredSolution
global BestFun
global BestW
global numparticulas
global tol
global diselementos; %Distancia entre elementos de la antenna (
    Flotante)
global Nusuarios;          % Numero de usuarios
global U1;                   %   - U1          ubicaci n usuario 1
    en grados    (entero)
global U2;                   %   - U2          ubicaci n usuario 2
    en grados    (entero)
global U3                    %   - U3          ubicaci n usuario 3
     en grados    (entero)
global maxatasco
global maxit
global vi
global CUCKOOpHistory
global CUCKOOFitHistory
global Migration4BAT
global Migration4PSO
global Migration4CUCKOO
global Migration4GA

global desiredsolution
global T
global acc
```

```
33 empty=isempty(Migration4CUCKOO);
34 if empty==0
35     wiCUCKOO(10,:)=Migration4CUCKOO;
36     Migration4CUCKOO = double.empty      ;
37     helpCUCKOO=0;
38     AtascoCountCUCKOO=0;
39 end
40
41 if helpPSO==1
42     if BestfunPSO<BestfunCUCKOO
43         [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
              wiCUCKOO);
44         Bestfunlimit=BestfunCUCKOO*0.8  ;
45         [index]=find(cost<Bestfunlimit);
46         [~,sizeindex]=size(index);
47       if sizeindex>0
48
49         Migration4PSO(1,:)=wiCUCKOO(1,:);
50 %         Migration4PSO(2,:)=wiCUCKOO(index(1),:);
51 %         Migration4PSO(3,:)=wiCUCKOO(index(3),:);
52 %         Migration4PSO(4,:)=wiCUCKOO(index(5),:);
53 %         Migration4PSO(5,:)=wiCUCKOO(index(end),:);
54       else
55          Migration4PSO(1,:)=wiCUCKOO(1,:);
56 %         Migration4PSO(2,:)=wiCUCKOO(3,:);
57 %         Migration4PSO(3,:)=wiCUCKOO(7,:);
58 %         Migration4PSO(4,:)=wiCUCKOO(15,:);
59 %         Migration4PSO(5,:)=wiCUCKOO(20,:);
60
61       end
62
63
64         helpPSO=0;
65
66     else
67         Migration4PSO=double.empty;
68     end
69
```

```
70  end
71
72  if helpBAT==1
73      if BestfunBAT<BestfunCUCKOO
74          [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
                wiCUCKOO);
75          Bestfunlimit=BestfunCUCKOO*0.8  ;
76          [index]=find(cost<Bestfunlimit);
77          [~,sizeindex]=size(index);
78       if sizeindex>0
79          Migration4BAT(1,:)=wiCUCKOO(1,:);
80  %          Migration4BAT(2,:)=wiCUCKOO(index(1),:);
81  %          Migration4BAT(3,:)=wiCUCKOO(index(3),:);
82  %          Migration4BAT(4,:)=wiCUCKOO(index(5),:);
83  %          Migration4BAT(5,:)=wiCUCKOO(index(end),:);
84       else
85          Migration4BAT(1,:)=wiCUCKOO(1,:);
86  %          Migration4BAT(2,:)=wiCUCKOO(3,:);
87  %          Migration4BAT(3,:)=wiCUCKOO(7,:);
88  %          Migration4BAT(4,:)=wiCUCKOO(15,:);
89  %          Migration4BAT(5,:)=wiCUCKOO(20,:);
90
91       end
92
93
94          helpBAT=0;
95
96      else
97          Migration4BAT=double.empty;
98      end
99
100 end
101
102 if helpGA==1
103     if BestfunGA<BestfunCUCKOO
104         [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
                wiCUCKOO);
105         Bestfunlimit=BestfunCUCKOO*0.8  ;
```

```
106             [index]=find(cost<Bestfunlimit);
107             [~,sizeindex]=size(index);
108         if sizeindex>4
109
110             Migration4GA(1,:)=wiCUCKOO(1,:);
111 %             Migration4GA(2,:)=wiCUCKOO(index(1),:);
112 %             Migration4GA(3,:)=wiCUCKOO(index(3),:);
113 %             Migration4GA(4,:)=wiCUCKOO(index(5),:);
114 %             Migration4GA(5,:)=wiCUCKOO(index(end),:);
115         else
116             Migration4GA(1,:)=wiCUCKOO(1,:);
117 %             Migration4GA(2,:)=wiCUCKOO(3,:);
118 %             Migration4GA(3,:)=wiCUCKOO(index(3),:);
119 %             Migration4GA(4,:)=wiCUCKOO(index(5),:);
120 %             Migration4GA(5,:)=wiCUCKOO(index(end),:);
121
122         end
123
124
125             helpGA=0;
126
127       else
128             Migration4GA=double.empty;
129       end
130 end
131
132
133 pout = SimpleCuckoo(wiCUCKOO,pa,CUCKOObeta,Nusuarios,U1,U2,U3);
134 [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,pout); %
      evalua el vector de pesos wi en la funcion de costos
135 [cost2,ind] = sort(cost,'descend'); % sorting
136             pout= pout(ind,:);
137
138             pout(end,:) = CUCKOOpHistory(k-1,:); %The former best
                sustitute now the current worst
139
140             [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
                pout); % Sort again to keep the current best
```

```
141         [cost,ind] = sort(cost,'descend');
142         pout= pout(ind,:);
143
144         Localbest=pout(1,:);
145         localsolution=cost(1);
146
147
148             [diff]=calcproduct(Localbest, desiredsolution,
                    Nusuarios,pout);
149
150
151
152                 fitness=localsolution/diff;
153             if fitness>BestfunCUCKOO %&& diff<=CUCKOOglobaldiff &&
                    localsolution<=norm(NormalizedDesiredSolution)
154                 CUCKOOFitHistory = [CUCKOOFitHistory fitness] ;
155                 CUCKOOpHistory = [CUCKOOpHistory;Localbest];
156                 CUCKOOglobaldiff=diff;
157             else
158                 CUCKOOFitHistory = [CUCKOOFitHistory
                        CUCKOOFitHistory(end)] ;
159                 CUCKOOpHistory = [CUCKOOpHistory;CUCKOOpHistory(
                        end,:)];
160
161
162
163          end
164
165
166
167
168
169         BestfunCUCKOO=CUCKOOFitHistory(end);
170         BestCUCKOOWI=CUCKOOpHistory(end,:);
171         [errCUCKOO,AtascoCountCUCKOO,k,helpCUCKOO]=SimpleEndIsland
                (BestfunCUCKOO, errCUCKOO, NormalizedDesiredSolution,k,
                AtascoCountCUCKOO,maxit,maxatasco);
172         CUCKOOerrv(k)=errCUCKOO;
```

```
173
174             wiCUCKOO=pout;
175
176 end
177
178 function[solutions,DesiredSolution,NormalizedDesiredSolutionVector
    ]=costfunction(Nusuarios,U1,U2,U3,numparticulas,wi) %funcion
    que evalua el vector de pesos y genera un diagrama de solucion
    con los usuarios.
179 global rang
180 gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo; %
    funcion gaussiana
181 amp=1000; %amplitud
182 width=rang; %ancho (para distancias cercanas es 6, distancias
    medianas 4.5, distancias lejanas 6)
183 cant=1800; %cantidad de puntos
184 desplazamiento=0; %desplazamiento de las campanas
185 thetar=[0 180]; %rango de theta
186 THETA=linspace(thetar(1),thetar(2),cant); %theta en grados
187 rad=pi/180; %radianes
188 NumAntElem=size(wi,2);
189 NumAntElem=sqrt(NumAntElem); %numero de elementos de la antenna
190 antdim=[NumAntElem NumAntElem]; %dimensiones de la antea
191 diselementos=[0.5 0.5]; %distancia entre elementos en la antenna
192 N=[NumAntElem, NumAntElem];
193 thetar=[0 180]*rad; %theta en radianes
194 THETARAD=linspace(thetar(1),thetar(2),cant);     % 1 x cant va
    desde 0 a pi en 1800 pasos
195 k=2*pi;
196 d=diselementos;
197 if Nusuarios==2 %caso si se tiene 2 usuarios
198     usuarios=[U1 U2]; %ubicacion usuarios
199     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
        desplazamiento)+gaus(THETA,usuarios(2),width,amp,
        desplazamiento);%+gaus(THETA,t0(3),sig,amp,vo);
200      %se genera la solucion deseada que sera utilizada como
         referencia (funcion gaussiana)
201 elseif Nusuarios==3
```

```
202     usuarios=[U1 U2 U3] ; %ubicacion usuarios
203     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
            desplazamiento)+gaus(THETA,usuarios(2),width,amp,
            desplazamiento)+gaus(THETA,usuarios(3),width,amp,
            desplazamiento);
204     %se genera la solucion deseada que sera utilizada como
            referencia (funcion gaussiana)
205 end
206
207 Psix = (((1:N(1))-1).*(k*d(1)*sin(THETARAD).*cos(0))');
208 Psiy = (((1:N(2))-1).*(k*d(2)*sin(THETARAD).*sin(0))');
209 for i=1:N(1)
210     sumaPsi(N(1)*(i-1)+1:i*N(1),:)=Psix(i,:)+Psiy;
211 end
212 sumaPsi=exp(1i*sumaPsi); %64x1800 complex double;
213 ArrayFactor1            = exp(1i*wi)*sumaPsi;
214
215
216
217 %ArrayFactor1=faf2(exp(1i*wi),antdim,diselementos,THETARAD,0); %se
        calcula el factor de arreglo
218 NormalizedAF1=abs(ArrayFactor1); %valor absoluto del factor de
        arreglo
219  sumNAF1=sum(NormalizedAF1,2); %el valor total sumado de todos los
         elementos
220  NAF1=NormalizedAF1./sumNAF1; %Normalizacion del factor de arreglo
221  M = max(NAF1,[],2); %se encuentra el punto maximo del factor de
        arreglo normalizado
222  DesiredSolution1=repmat(DesiredSolution,numparticulas,1); %se
        genera matriz repetida de solucion deseada
223 NormalizedDesiredSolutionVector=DesiredSolution1.*M; %multiplico
        el punto mas alto por la solucion deseada(la solcucion deseada
        (desired solution) tiene una amplitud de mil y el punto maximo
        es como de 2x10-3, este producto se hace para obtener una
        solucion deseada que este en una misma escala que las
        soluciones que se van a ir propionendo)
224 prodv=NAF1.*DesiredSolution1; %multiplico la solucion deseada por
        el factor de arreglo normalizado
```

```
225 solutions=sqrt(sum(prodv.^2,2)); %calculo la norma del producto
    anterior
226
227
228 end
229
230
231 function[diff]=calcproduct(Pout, desiredsolution,Nusuarios,wi)
232 rad=pi/180;
233 cant=1800;
234
235 NumAntElem=size(wi,2);
236 NumAntElem=sqrt(NumAntElem);
237 antdim=[NumAntElem NumAntElem];
238 diselementos=[0.5 0.5];
239
240 thetar=[0 180]*rad;
241 THETARAD=linspace(thetar(1),thetar(2),cant);
242 ArrayFactor=faf(exp(1i*Pout),antdim,diselementos,THETARAD,0);%
    Patron de radiacion de la particula i
243 NormalizedAF=abs(ArrayFactor)/sum(abs(ArrayFactor));
244 prod=NormalizedAF.*desiredsolution;
245 %plot(prod)
246
247 % x = linspace(0,180,1800);
248 %
249 % plot(x,prod)
250 if Nusuarios==2
251     [maxvalue1,pos1]=max(prod);
252   if pos1>100
253         posv1=[pos1-100:pos1+100];
254         prod(1,posv1) = 0;
255     else
256         posv1=[0:pos1+100];
257         prod(1,posv1+1) = 0;
258
259     end
260
```

```
261
262
263       %plot(prod)
264
265       [maxvalue2,pos2]=max(prod);
266       if pos2>100
267            posv2=[pos2-100:pos2+100];
268            prod(1,posv2) = 0;
269       else
270            posv2=[0:pos2+100];
271            prod(1,posv2+1) = 0;
272
273       end
274
275
276       %plot(prod)
277
278       diff=abs(maxvalue1-maxvalue2);
279  elseif Nusuarios==3
280
281
282
283       [maxvalue1,pos1]=max(prod);
284         if pos1>100
285            posv1=[pos1-100:pos1+100];
286            prod(1,posv1) = 0;
287       else
288            posv1=[0:pos1+100];
289            prod(1,posv1+1) = 0;
290
291     end
292
293  %       plot(prod)
294
295     [maxvalue2,pos2]=max(prod);
296       if pos2>100
297            posv2=[pos2-100:pos2+100];
298            prod(1,posv2) = 0;
```

```
299       else
300           posv2=[0:pos2+100];
301           prod(1,posv2+1) = 0;
302
303       end
304
305
306  %       plot(prod)
307
308      [maxvalue3,pos3]=max(prod);
309       if pos3>100
310           posv3=[pos3-100:pos3+100];
311           prod(1,posv3) = 0;
312       else
313           posv3=[0:pos3+100];
314           prod(1,posv3+1) = 0;
315
316       end
317
318     diff=abs((maxvalue1-maxvalue2)+(maxvalue1-maxvalue3));
319  end
320
321
322  end
```

A.1.1.8. Código de bloque SA de estrategia híbrida MAS

```
1  function [BestWSA,BestFunSA,FitHistory,pHistory,globaldiff]=
       SAblock(BestWSA,alpha,tol,rej,itcount,BestFunSA,globaldiff,
       FitHistory,pHistory)
2  global desiredsolution
3  global NormalizedDesiredSolution
4  global BestFun
5  global BestW
6  global numparticulas
7  global Nusuarios;          % Numero de usuarios
8  global U1;                   %   - U1          ubicacion usuario 1
       en grados    (entero)
9  global U2;                   %   - U2          ubicacion usuario 2
```

```
   en grados   (entero)
global U3                    %   - U3          ubicaci n usuario 3
    en grados   (entero)
global k2
        k2 = k2 + 1;
        numparticulas=1;
        [pout] = simpleSA(BestWSA,alpha,BestWSA,BestFunSA,tol,rej,
           itcount,Nusuarios,U1,U2,U3);


        [cost,~]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
           pout);


        localsolution=cost(1);

        Localbest=pout;




        [diff]=calcproduct(Localbest, desiredsolution,Nusuarios,
           pout);


        fitness=localsolution/diff;

        if fitness>BestFunSA %&& diff<=globaldiff && localsolution
           <=norm(NormalizedDesiredSolution)
            FitHistory = [FitHistory fitness] ;
            pHistory = [pHistory;Localbest];
            globaldiff=diff;
        else
            FitHistory = [FitHistory FitHistory(end)] ;
            pHistory = [pHistory;pHistory(end,:)];
         end


```

```
%          if (cost>BestFunSA) && (cost<=norm(
    NormalizedDesiredSolution))
%              if (ee==1)
%                   FitHistory = [FitHistory cost] ;
%                   pHistory = [pHistory;pout];
%                   bar=0.9*bar;
%              else
%                  FitHistory = [FitHistory FitHistory(end)] ;
%                  pHistory = [pHistory;pHistory(end,:)];
%
%              end

%
%
%          else
%               FitHistory = [FitHistory FitHistory(end)] ;
%               pHistory = [pHistory;pHistory(end,:)];
%          end
%
          BestFunSA=FitHistory(end);
          BestWSA=pHistory(end,:);
end

function[solutions,DesiredSolution,NormalizedDesiredSolutionVector
    ]=costfunction(Nusuarios,U1,U2,U3,numparticulas,wi) %funcion
    que evalua el vector de pesos y genera un diagrama de solucion
    con los usuarios.
global rang
gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo; %
    funcion gaussiana
amp=1000; %amplitud
width=rang; %ancho (para distancias cercanas es 6, distancias
    medianas 4.5, distancias lejanas 6)
cant=1800; %cantidad de puntos
desplazamiento=0; %desplazamiento de las campanas
thetar=[0 180];  %rango de theta
THETA=linspace(thetar(1),thetar(2),cant); %theta en grados
rad=pi/180; %radianes
```

```
74 NumAntElem=size(wi,2);
75 NumAntElem=sqrt(NumAntElem); %numero de elementos de la antenna
76 antdim=[NumAntElem NumAntElem]; %dimensiones de la antea
77 diselementos=[0.5 0.5]; %distancia entre elementos en la antenna
78 N=[NumAntElem, NumAntElem];
79 thetar=[0 180]*rad; %theta en radianes
80 THETARAD=linspace(thetar(1),thetar(2),cant);     % 1 x cant va
      desde 0 a pi en 1800 pasos
81 k=2*pi;
82 d=diselementos;
83 if Nusuarios==2  %caso si se tiene 2 usuarios
84     usuarios=[U1 U2]; %ubicacion usuarios
85     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
         desplazamiento)+gaus(THETA,usuarios(2),width,amp,
         desplazamiento);%+gaus(THETA,t0(3),sig,amp,vo);
86     %se genera la solucion deseada que sera utilizada como
         referencia (funcion gaussiana)
87 elseif Nusuarios==3
88     usuarios=[U1 U2 U3] ; %ubicacion usuarios
89     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
         desplazamiento)+gaus(THETA,usuarios(2),width,amp,
         desplazamiento)+gaus(THETA,usuarios(3),width,amp,
         desplazamiento);
90     %se genera la solucion deseada que sera utilizada como
         referencia (funcion gaussiana)
91 end
92
93 Psix = (((1:N(1))-1).*(k*d(1)*sin(THETARAD).*cos(0))');
94 Psiy = (((1:N(2))-1).*(k*d(2)*sin(THETARAD).*sin(0))');
95 for i=1:N(1)
96     sumaPsi(N(1)*(i-1)+1:i*N(1),:)=Psix(i,:)+Psiy;
97 end
98 sumaPsi=exp(1i*sumaPsi); %64x1800 complex double;
99 ArrayFactor1            = exp(1i*wi)*sumaPsi;
100
101
102
103 %ArrayFactor1=faf2(exp(1i*wi),antdim,diselementos,THETARAD,0); %se
```

```
    calcula el factor de arreglo
NormalizedAF1=abs(ArrayFactor1); %valor absoluto del factor de
    arreglo
 sumNAF1=sum(NormalizedAF1,2); %el valor total sumado de todos los
     elementos
 NAF1=NormalizedAF1./sumNAF1; %Normalizacion del factor de arreglo
 M = max(NAF1,[],2); %se encuentra el punto maximo del factor de
    arreglo normalizado
  DesiredSolution1=repmat(DesiredSolution,numparticulas,1); %se
    genera matriz repetida de solucion deseada
NormalizedDesiredSolutionVector=DesiredSolution1.*M; %multiplico
    el punto mas alto por la solucion deseada(la solcucion deseada
    (desired solution) tiene una amplitud de mil y el punto maximo
    es como de 2x10-3, este producto se hace para obtener una
    solucion deseada que este en una misma escala que las
    soluciones que se van a ir propionendo)
prodv=NAF1.*DesiredSolution1; %multiplico la solucion deseada por
    el factor de arreglo normalizado
solutions=sqrt(sum(prodv.^2,2)); %calculo la norma del producto
    anterior


end




%
%
%
%
%
% function[solutions,DesiredSolution,
    NormalizedDesiredSolutionVector]=costfunction(Nusuarios,U1,U2,
    U3,numparticulas,wi)
% gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo;
```

```
128 % amp=1000;
129 % width=3;
130 % cant=1800;
131 % desplazamiento=0;
132 % thetar=[0 180];
133 % THETA=linspace(thetar(1),thetar(2),cant);
134 % rad=pi/180;
135 % NumAntElem=size(wi,2);
136 % NumAntElem=sqrt(NumAntElem);
137 % antdim=[NumAntElem NumAntElem];
138 % diselementos=[0.5 0.5];
139 %
140 % thetar=[0 180]*rad;
141 % THETARAD=linspace(thetar(1),thetar(2),cant);  % 1 x cant va
    desde 0 a pi en 1800 pasos
142 %
143 % if Nusuarios==2
144 %    usuarios=[U1 U2];
145 %    DesiredSolution = gaus(THETA,usuarios(1),width,amp,
    desplazamiento)+gaus(THETA,usuarios(2),width,amp,desplazamiento
    );%+gaus(THETA,t0(3),sig,amp,vo);
146 % elseif Nusuarios==3
147 %     usuarios=[U1 U2 U3] ;
148 %     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
    desplazamiento)+gaus(THETA,usuarios(2),width,amp,desplazamiento
    )+gaus(THETA,usuarios(3),width,amp,desplazamiento);
149 % end
150 %
151 %
152 % for i=1:numparticulas
153 %     ArrayFactor=faf(exp(1i*wi(i,:)),antdim,diselementos,THETARAD
    ,0);% Patron de radiacion de la particula i
154 %      NormalizedAF=abs(ArrayFactor)/sum(abs(ArrayFactor));
155 %      maximumAF=max(NormalizedAF);  %valor maximo en factor de
    arreglo
156 %      NormalizedDesiredSolution=DesiredSolution*maximumAF;
157 %      Product=NormalizedAF.*DesiredSolution;
158 %     NormalizedDesiredSolutionVector(i,:)=
```

```
    NormalizedDesiredSolution;
%       solutions(i)=norm(Product);
% end
%
%
% end


function[diff]=calcproduct(Pout, desiredsolution,Nusuarios,wi)
rad=pi/180;
cant=1800;

NumAntElem=size(wi,2);
NumAntElem=sqrt(NumAntElem);
antdim=[NumAntElem NumAntElem];
diselementos=[0.5 0.5];

thetar=[0 180]*rad;
THETARAD=linspace(thetar(1),thetar(2),cant);
ArrayFactor=faf(exp(1i*Pout),antdim,diselementos,THETARAD,0);%
    Patron de radiacion de la particula i
NormalizedAF=abs(ArrayFactor)/sum(abs(ArrayFactor));
prod=NormalizedAF.*desiredsolution;
%plot(prod)

% x = linspace(0,180,1800);
%
% plot(x,prod)
if Nusuarios==2
     [maxvalue1,pos1]=max(prod);
   if pos1>100
          posv1=[pos1-100:pos1+100];
          prod(1,posv1) = 0;
     else
          posv1=[0:pos1+100];
          prod(1,posv1+1) = 0;

    end
```

```
195
196
197
198       %plot(prod)
199
200       [maxvalue2,pos2]=max(prod);
201       if pos2>100
202            posv2=[pos2-100:pos2+100];
203            prod(1,posv2) = 0;
204       else
205            posv2=[0:pos2+100];
206            prod(1,posv2+1) = 0;
207
208       end
209
210
211       %plot(prod)
212
213       diff=abs(maxvalue1-maxvalue2);
214  elseif Nusuarios==3
215
216
217
218       [maxvalue1,pos1]=max(prod);
219         if pos1>100
220            posv1=[pos1-100:pos1+100];
221            prod(1,posv1) = 0;
222       else
223            posv1=[0:pos1+100];
224            prod(1,posv1+1) = 0;
225
226      end
227
228  %       plot(prod)
229
230      [maxvalue2,pos2]=max(prod);
231       if pos2>100
232            posv2=[pos2-100:pos2+100];
```

```
233             prod(1,posv2) = 0;
234         else
235             posv2=[0:pos2+100];
236             prod(1,posv2+1) = 0;
237
238         end
239
240
241 %         plot(prod)
242
243         [maxvalue3,pos3]=max(prod);
244         if pos3>100
245             posv3=[pos3-100:pos3+100];
246             prod(1,posv3) = 0;
247         else
248             posv3=[0:pos3+100];
249             prod(1,posv3+1) = 0;
250
251         end
252
253       diff=abs((maxvalue1-maxvalue2)+(maxvalue1-maxvalue3));
254 end
255
256
257 end
```

A.1.1.9. Código de algoritmo SA

```
1  function [BestW] = simpleSA(wi,alpha,BestW,BestFun,tol,rej,i,
       Nusuarios,U1,U2,U3)
2  global T
3  global acc
4  alphapos=0.3;
5
6  iter=5;
7  itera=7;
8
9  numparticulas=1;%size(wi,1);
10 NumAntElem=size(wi,2);
```

```
11
12      [w2]=NewPosition(numparticulas,alphapos,NumAntElem,BestW);
13      [w2]=enrango(w2);
14      [solutions,NormalizedDesiredSolutionVector]=costfunction(
            Nusuarios,U1,U2,U3,numparticulas,w2); %evalua el vector de
            pesos wi en la funcion de costos
15      [BestW2,BestFun2,~]=choosebest(solutions,
            NormalizedDesiredSolutionVector,w2); %elige el conjunto de
            vectores de peso que produce la mejor solucion, el valor de
             la funcion al evaluar dicho vector y computa la solucion
            hacia la que el algoritmo deberia converger
16      DeltaF=BestFun2-BestFun;
17      [~,acc,rej,BestW]=AcceptingSolutions(DeltaF,T,tol,acc,rej,
            BestFun2,BestW2);
18      [i,acc,T]=UpdateTemperature(i,iter,acc,itera,T,alpha);
19
20
21  end
22
23
24
25
26  function [w2]=NewPosition(numparticulas,alpha,NumAntElem,BestW)
27      w2=BestW+(rand(numparticulas,NumAntElem)*2-1)*alpha;
28
29  end
30  function[err,AtascoCount,iter]=errorcalc(BestFun, err,
        NormalizedDesiredSolution,iter,AtascoCount)
31  previouserrorA=err;
32   err=sum(abs( BestFun-norm(NormalizedDesiredSolution)));
33   if previouserrorA==err
34       AtascoCount=AtascoCount+1;
35   else
36       AtascoCount=0;
37   end
38   iter=iter+1;
39  end
40  function [bestlocalfun,acc,rej,BestW]=AcceptingSolutions(DeltaF,T,
```

```
    tol,acc,rej,BestFun2,BestW2)

randomnum=rand;

      if -DeltaF>tol            % Diferencia negativa se acepta
          bestlocalfun=BestFun2   ;      % Se actualiza agrega como
             la mejor posicion
          acc=acc+1         ;         % Aceptados
          BestW=BestW2;



    elseif exp(-DeltaF/T)>randomnum       %Criterio de Metropolis
        bestlocalfun   = BestFun2;
        BestW=BestW2;
        acc=acc+1;
        rej=0;
     else
        bestlocalfun   = BestFun2;
        BestW=BestW2;
        acc=acc+1;
        bestlocalfun=0;
        rej=rej+1;                  % Rechazos
    end


end
function [i,acc,T]=UpdateTemperature(i,iter,acc,itera,T,alpha)
      i=i+1;
%         iter
%         acc
%         itera
      if i>=iter || acc>=itera  % Aceptados o Ejecucion
        T = alpha*T;          % Cambio de temperatura
        i=1;
        acc=1;
    end

```

```
77 end
78
79 %
80 % end
81
82
83 function[solutions,DesiredSolution,NormalizedDesiredSolutionVector
       ]=costfunction(Nusuarios,U1,U2,U3,numparticulas,wi) %funcion
       que evalua el vector de pesos y genera un diagrama de solucion
       con los usuarios.
84 global rang
85 gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo; %
       funcion gaussiana
86 amp=1000; %amplitud
87 width=rang; %ancho (para distancias cercanas es 6, distancias
       medianas 4.5, distancias lejanas 6)
88 cant=1800; %cantidad de puntos
89 desplazamiento=0; %desplazamiento de las campanas
90 thetar=[0 180]; %rango de theta
91 THETA=linspace(thetar(1),thetar(2),cant); %theta en grados
92 rad=pi/180; %radianes
93 NumAntElem=size(wi,2);
94 NumAntElem=sqrt(NumAntElem); %numero de elementos de la antenna
95 antdim=[NumAntElem NumAntElem]; %dimensiones de la antea
96 diselementos=[0.5 0.5]; %distancia entre elementos en la antenna
97 N=[NumAntElem, NumAntElem];
98 thetar=[0 180]*rad; %theta en radianes
99 THETARAD=linspace(thetar(1),thetar(2),cant);     % 1 x cant va
       desde 0 a pi en 1800 pasos
100 k=2*pi;
101 d=diselementos;
102 if Nusuarios==2 %caso si se tiene 2 usuarios
103     usuarios=[U1 U2]; %ubicacion usuarios
104     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
           desplazamiento)+gaus(THETA,usuarios(2),width,amp,
           desplazamiento);%+gaus(THETA,t0(3),sig,amp,vo);
105      %se genera la solucion deseada que sera utilizada como
            referencia (funcion gaussiana)
```

```
106 elseif Nusuarios==3
107     usuarios=[U1 U2 U3] ; %ubicacion usuarios
108     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
            desplazamiento)+gaus(THETA,usuarios(2),width,amp,
            desplazamiento)+gaus(THETA,usuarios(3),width,amp,
            desplazamiento);
109     %se genera la solucion deseada que sera utilizada como
            referencia (funcion gaussiana)
110 end
111
112 Psix = (((1:N(1))-1).*(k*d(1)*sin(THETARAD).*cos(0))');
113 Psiy = (((1:N(2))-1).*(k*d(2)*sin(THETARAD).*sin(0))');
114 for i=1:N(1)
115     sumaPsi(N(1)*(i-1)+1:i*N(1),:)=Psix(i,:)+Psiy;
116 end
117 sumaPsi=exp(1i*sumaPsi); %64x1800 complex double;
118 ArrayFactor1            = exp(1i*wi)*sumaPsi;
119
120
121
122 %ArrayFactor1=faf2(exp(1i*wi),antdim,diselementos,THETARAD,0); %se
        calcula el factor de arreglo
123 NormalizedAF1=abs(ArrayFactor1); %valor absoluto del factor de
        arreglo
124  sumNAF1=sum(NormalizedAF1,2); %el valor total sumado de todos los
         elementos
125  NAF1=NormalizedAF1./sumNAF1; %Normalizacion del factor de arreglo
126  M = max(NAF1,[],2); %se encuentra el punto maximo del factor de
        arreglo normalizado
127   DesiredSolution1=repmat(DesiredSolution,numparticulas,1); %se
        genera matriz repetida de solucion deseada
128 NormalizedDesiredSolutionVector=DesiredSolution1.*M; %multiplico
        el punto mas alto por la solucion deseada(la solcucion deseada
        (desired solution) tiene una amplitud de mil y el punto maximo
        es como de 2x10-3, este producto se hace para obtener una
        solucion deseada que este en una misma escala que las
        soluciones que se van a ir propionendo)
129 prodv=NAF1.*DesiredSolution1; %multiplico la solucion deseada por
```

```
    el factor de arreglo normalizado
130 solutions=sqrt(sum(prodv.^2,2)); %calculo la norma del producto
    anterior
131
132
133 end
134
135
136
137
138 function [wi]=enrango(wi)
139 wi=mod(wi,2*pi);
140 end
141 function [wi]=initalize(NumAntElem,numparticulas)
142 LB=0*ones(1,NumAntElem);              % Limite inferior
143 UB=2*pi*ones(1,NumAntElem);           % Limite Superior
144 wmin=min(LB);                  % w minimo
145 wmax=max(UB);                  % w maximo
146
147
148
149 wi=(rand(numparticulas,NumAntElem))*(wmax-wmin)+wmin;
150 end
151 function [BestW,BestF,NormalizedDesiredSolution]=choosebest(
    solutions,NormalizedDesiredSolutionVector,wi)
152
153 [~,I] = max(solutions);
154 NormalizedDesiredSolution=NormalizedDesiredSolutionVector(I,:);
155 BestW =wi(I,:);
156 BestF=solutions(I);
157
158
159 end
160 function [bestfunfinal,BestFun,BestW]=SelectingSolution(
    NormalizedDesiredSolution,BestFun,BestFun2,BestW2,BestW)
161  if BestFun<=BestFun2 && BestFun2<=norm(NormalizedDesiredSolution)
     %se selecciona cual de los dos casos es mejor, si los valores
     obtenidos con xi1 o xi2
```

```
162             BestW=BestW2;
163             bestfunfinal=BestFun2;
164             BestFun=BestFun2;
165     else
166              bestfunfinal=BestFun;
167              BestFun=BestFun;
168              BestW=BestW;
169
170     end
171
172 end
173 function Plotting(antdim,diselementos, BestW )
174 rad=pi/180;
175 antdim=[antdim antdim];
176 diselementos=[diselementos diselementos];
177
178 thetar=[0 180]*rad;
179 THETARAD=linspace(thetar(1),thetar(2),1800);     % 1 x cant va
      desde 0 a pi en 1800 pasos
180
181
182 AF=faf(exp(1i*BestW),antdim,diselementos,THETARAD,0);
183 AFnorma=abs(AF/sum(abs(AF)));
184 figure(9);
185 plot(AFnorma);
186
187
188 W=exp(1i*BestW);
189 Wbestpos=abs(W);                                  % Magnitudes
190 betabestpos=mod(angle(W)*180/pi,360);     % Angulos
191 nombre='PSO';
192
193 radtheta(W,antdim,diselementos,0,nombre)
194 % %----------------------Patron de Radiaci n en Theta Polares
      ------------------------
195   radthetapol(W,antdim,diselementos,0,nombre)
196
197
```

```
198
199
200 end
```

A.1.1.10. Código de algoritmo PSO

```
1  function wi = SimplePSO(wi,alpha,beta,inercia,BestW,BestFun,
       BestWhist);
2  global viPSO
3
4  numparticulas=size(wi,1);
5  NumAntElem=64;%size(wi,2);
6
7  [vi]=NewVelocity(alpha,beta,inercia,numparticulas,NumAntElem,viPSO
       ,BestW,BestFun,wi);  %Calcula la nueva velocidad
8  [wi]=NewPosition(vi,wi);  %usuando la nueva velocidad calcula
9  [wi]=enrango(wi);%verifica que el vector de pesos wi este dentro
       del rango 0 a 2pi
10 % [solutions,~,wi]=costfunction(Nusuarios,U1,U2,U3,numparticulas,
       wi); %evalua el vector de pesos wi en la funcion de costos
11 % [NewW,NewFun,~]=choosebest(solutions,
       NormalizedDesiredSolutionVector,wi);%elige la mejor solucion y
       computa la solucion hacia la que el algoritmo deberia converger
12 % [BestW,BestFun]=Compare(NewW,NewFun,NormalizedDesiredSolution,
       BestW,BestFun); %compara la nueva solucion actual con la global
        y elige si es mejor
13 %
14
15 end
16
17
18
19
20
21
22 function[err,AtascoCount,iter]=errorcalc(BestFun, err,
       NormalizedDesiredSolution,iter,AtascoCount)
23 previouserrorA=err;
24  err=sum(abs( BestFun-norm(NormalizedDesiredSolution)));
```

```
25  if previouserrorA==err
26       AtascoCount=AtascoCount+1;
27  else
28       AtascoCount=0;
29  end
30  iter=iter+1;
31 end
32 function [BestW,BestFun]=Compare(NewW,NewFun,
       NormalizedDesiredSolutionVector,BestW,BestFun)
33 if (NewFun>BestFun) && (NewFun<=norm(
       NormalizedDesiredSolutionVector))
34           BestFun=NewFun
35           BestW=NewW
36
37      end
38 end
39 function [wi]=NewPosition(vi,wi)
40  wi=wi+vi;
41 end
42
43 function [vi]=NewVelocity(alpha,beta,inercia,numparticulas,
       NumAntElem,vi,BestW,BestWhist,wi)
44       vi=inercia*vi+alpha*rand(numparticulas,NumAntElem).*(repmat(
             BestW,numparticulas,1)-wi)+beta*rand(numparticulas,
             NumAntElem).*(repmat(BestWhist,numparticulas,1)-wi);
45
46 % vi=inercia*vi+alpha*rand(n,NumAntElem).*(repmat(BestW,n,1)-wi)+
       beta*rand(n,NumAntElem).*(repmat(BestFun,n,1)-wi);
47 %    wi=wi+vi;
48 %       wi3=mod(wi,2*pi);
49 end
50 function [BestW,BestF,NormalizedDesiredSolution]=choosebest(
       solutions,NormalizedDesiredSolutionVector,wi)
51
52 [~,I] = max(solutions);
53 NormalizedDesiredSolution=NormalizedDesiredSolutionVector(I,:);
54 BestW =wi(I,:);
55 BestF=solutions(I);
```

```
56
57
58 end
59
60
61
62 function[solutions,DesiredSolution,NormalizedDesiredSolutionVector
   ]=costfunction(Nusuarios,U1,U2,U3,numparticulas,wi) %funcion
   que evalua el vector de pesos y genera un diagrama de solucion
   con los usuarios.
63 global rang
64 gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo; %
   funcion gaussiana
65 amp=1000; %amplitud
66 width=rang; %ancho (para distancias cercanas es 6, distancias
   medianas 4.5, distancias lejanas 6)
67 cant=1800; %cantidad de puntos
68 desplazamiento=0; %desplazamiento de las campanas
69 thetar=[0 180]; %rango de theta
70 THETA=linspace(thetar(1),thetar(2),cant); %theta en grados
71 rad=pi/180; %radianes
72 NumAntElem=size(wi,2);
73 NumAntElem=sqrt(NumAntElem); %numero de elementos de la antenna
74 antdim=[NumAntElem NumAntElem]; %dimensiones de la antea
75 diselementos=[0.5 0.5]; %distancia entre elementos en la antenna
76 N=[NumAntElem, NumAntElem];
77 thetar=[0 180]*rad; %theta en radianes
78 THETARAD=linspace(thetar(1),thetar(2),cant);     % 1 x cant va
   desde 0 a pi en 1800 pasos
79 k=2*pi;
80 d=diselementos;
81 if Nusuarios==2 %caso si se tiene 2 usuarios
82     usuarios=[U1 U2]; %ubicacion usuarios
83     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
          desplazamiento)+gaus(THETA,usuarios(2),width,amp,
          desplazamiento);%+gaus(THETA,t0(3),sig,amp,vo);
84      %se genera la solucion deseada que sera utilizada como
           referencia (funcion gaussiana)
```

```
85  elseif Nusuarios==3
86      usuarios=[U1 U2 U3] ; %ubicacion usuarios
87      DesiredSolution = gaus(THETA,usuarios(1),width,amp,
            desplazamiento)+gaus(THETA,usuarios(2),width,amp,
            desplazamiento)+gaus(THETA,usuarios(3),width,amp,
            desplazamiento);
88      %se genera la solucion deseada que sera utilizada como
            referencia (funcion gaussiana)
89  end
90
91  Psix = (((1:N(1))-1).*(k*d(1)*sin(THETARAD).*cos(0))');
92  Psiy = (((1:N(2))-1).*(k*d(2)*sin(THETARAD).*sin(0))');
93  for i=1:N(1)
94      sumaPsi(N(1)*(i-1)+1:i*N(1),:)=Psix(i,:)+Psiy;
95  end
96  sumaPsi=exp(1i*sumaPsi); %64x1800 complex double;
97  ArrayFactor1            = exp(1i*wi)*sumaPsi;
98
99
100
101 %ArrayFactor1=faf2(exp(1i*wi),antdim,diselementos,THETARAD,0); %se
        calcula el factor de arreglo
102 NormalizedAF1=abs(ArrayFactor1); %valor absoluto del factor de
        arreglo
103  sumNAF1=sum(NormalizedAF1,2); %el valor total sumado de todos los
         elementos
104  NAF1=NormalizedAF1./sumNAF1; %Normalizacion del factor de arreglo
105  M = max(NAF1,[],2); %se encuentra el punto maximo del factor de
        arreglo normalizado
106   DesiredSolution1=repmat(DesiredSolution,numparticulas,1); %se
        genera matriz repetida de solucion deseada
107 NormalizedDesiredSolutionVector=DesiredSolution1.*M; %multiplico
        el punto mas alto por la solucion deseada(la solcucion deseada
        (desired solution) tiene una amplitud de mil y el punto maximo
        es como de 2x10-3, este producto se hace para obtener una
        solucion deseada que este en una misma escala que las
        soluciones que se van a ir propionendo)
108 prodv=NAF1.*DesiredSolution1; %multiplico la solucion deseada por
```

```
    el factor de arreglo normalizado
109 solutions=sqrt(sum(prodv.^2,2)); %calculo la norma del producto
    anterior
110
111
112 end
113
114
115
116 function [wi]=enrango(wi)
117 wi=mod(wi,2*pi);
118 end
119 function [wi]=initalize(NumAntElem,numparticulas)
120 LB=0*ones(1,NumAntElem);              % Limite inferior
121 UB=2*pi*ones(1,NumAntElem);           % Limite Superior
122 wmin=min(LB);                  % w minimo
123 wmax=max(UB);                  % w maximo
124
125
126
127 wi=(rand(numparticulas,NumAntElem))*(wmax-wmin)+wmin;
128 end
```

A.1.1.11. Código de algoritmo GA

```
1  function pout = SimpleGA(p,pc,pm,mutstep,Nusuarios,U1,U2,U3);
2
3  % This a minimalist Genetic Algorithm (GA) function
4
5  % Input parameters: % pin = solutions matrix;  elitism, pc, pm,
     and acep are parameters of GA.
6  % Output parameters: % p_out = solutions matrix. Saved is a
     cellarray with the new best
7  % solution {solution fitness}
8
9       popsize = size(p,1); stepsize = .1; solnew = zeros(size(p));
10      [cost,~]=costfunction(Nusuarios,U1,U2,U3,popsize,p);
11      dice = rand(1,popsize);
12      [~,ind]= find((pc-dice)>0); % Select solutions to modify
```

```
13
14 %     Crossover. Totem pole strategy
15     R = cumsum(cost);
16     R = R/R(end);
17     crossFather = rand(popsize,1);
18     crossMother = rand(popsize,1);
19     for ric = 1:length(cost)
20         indF = find(R>crossFather(ric));
21         indM = find(R>crossMother(ric));
22         solnew(ric,:) = p(indM(1),:) + (p(indM(1),:) - p(indF(1)
               ,:)).*stepsize.*rand;
23     end
24
25     solnew=enrango(solnew);
26 %     ind = find(abs(solnew)>1); solnew(ind) = -1;% condition to
       return the chromosomes if it's out of bounds
27
28 %   Mutation
29     dice = rand(1,popsize);
30     ind = find((pm-dice)>0);
31     solnew(ind,:) = solnew(ind,:) + mutstep*randn;
32
33 %     ind = find(abs(solnew)>1); solnew(ind) = -1;% condition to
       return the chromosomes if it's out of bounds
34
35     solnew=enrango(solnew);
36     pout = solnew;
37
38 end
39
40
41 function[solutions,DesiredSolution,NormalizedDesiredSolutionVector
       ]=costfunction(Nusuarios,U1,U2,U3,numparticulas,wi) %funcion
       que evalua el vector de pesos y genera un diagrama de solucion
       con los usuarios.
42 global rang
43 gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo; %
       funcion gaussiana
```

```
44 amp=1000; %amplitud
45 width=rang; %ancho (para distancias cercanas es 6, distancias
       medianas 4.5, distancias lejanas 6)
46 cant=1800; %cantidad de puntos
47 desplazamiento=0; %desplazamiento de las campanas
48 thetar=[0 180]; %rango de theta
49 THETA=linspace(thetar(1),thetar(2),cant); %theta en grados
50 rad=pi/180; %radianes
51 NumAntElem=size(wi,2);
52 NumAntElem=sqrt(NumAntElem); %numero de elementos de la antenna
53 antdim=[NumAntElem NumAntElem]; %dimensiones de la antea
54 diselementos=[0.5 0.5]; %distancia entre elementos en la antenna
55 N=[NumAntElem, NumAntElem];
56 thetar=[0 180]*rad; %theta en radianes
57 THETARAD=linspace(thetar(1),thetar(2),cant);     % 1 x cant va
       desde 0 a pi en 1800 pasos
58 k=2*pi;
59 d=diselementos;
60 if Nusuarios==2 %caso si se tiene 2 usuarios
61     usuarios=[U1 U2]; %ubicacion usuarios
62     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
           desplazamiento)+gaus(THETA,usuarios(2),width,amp,
           desplazamiento);%+gaus(THETA,t0(3),sig,amp,vo);
63      %se genera la solucion deseada que sera utilizada como
           referencia (funcion gaussiana)
64 elseif Nusuarios==3
65      usuarios=[U1 U2 U3] ; %ubicacion usuarios
66      DesiredSolution = gaus(THETA,usuarios(1),width,amp,
           desplazamiento)+gaus(THETA,usuarios(2),width,amp,
           desplazamiento)+gaus(THETA,usuarios(3),width,amp,
           desplazamiento);
67      %se genera la solucion deseada que sera utilizada como
           referencia (funcion gaussiana)
68 end
69
70 Psix = (((1:N(1))-1).*(k*d(1)*sin(THETARAD).*cos(0))');
71 Psiy = (((1:N(2))-1).*(k*d(2)*sin(THETARAD).*sin(0))');
72 for i=1:N(1)
```

```
73      sumaPsi(N(1)*(i-1)+1:i*N(1),:)=Psix(i,:)+Psiy;
74  end
75  sumaPsi=exp(1i*sumaPsi); %64x1800 complex double;
76  ArrayFactor1            = exp(1i*wi)*sumaPsi;
77
78
79
80  %ArrayFactor1=faf2(exp(1i*wi),antdim,diselementos,THETARAD,0); %se
        calcula el factor de arreglo
81  NormalizedAF1=abs(ArrayFactor1); %valor absoluto del factor de
       arreglo
82   sumNAF1=sum(NormalizedAF1,2); %el valor total sumado de todos los
         elementos
83   NAF1=NormalizedAF1./sumNAF1; %Normalizacion del factor de arreglo
84   M = max(NAF1,[],2); %se encuentra el punto maximo del factor de
        arreglo normalizado
85    DesiredSolution1=repmat(DesiredSolution,numparticulas,1); %se
        genera matriz repetida de solucion deseada
86  NormalizedDesiredSolutionVector=DesiredSolution1.*M; %multiplico
       el punto mas alto por la solucion deseada(la solcucion deseada
       (desired solution) tiene una amplitud de mil y el punto maximo
       es como de 2x10-3, este producto se hace para obtener una
       solucion deseada que este en una misma escala que las
       soluciones que se van a ir propionendo)
87  prodv=NAF1.*DesiredSolution1; %multiplico la solucion deseada por
       el factor de arreglo normalizado
88  solutions=sqrt(sum(prodv.^2,2)); %calculo la norma del producto
       anterior
89
90
91  end
92
93
94
95
96  function [wi]=enrango(wi)
97  wi=mod(wi,2*pi);
98  end
```

A.1.1.12. Código de algoritmo BAT

```
function [OutW] = SimpleBAT(wi,aceleracion,alpha,theta,BestW,
    BestFun,Nusuarios,U1,U2,U3,initialpulserate,k,gamma)
    global Loudness
    global pulserate
    global viBAT

    t=0.9;
    numparticulas=size(wi,1);
    NumAntElem=size(wi,2);
    fmin    = 0;                    % Frecuencia minima
    fmax    = 2;                    % Frecuncia maxima

    [fi]=NewFrequency(fmax,fmin,numparticulas);
    [viBAT]=NewVelocity(aceleracion,theta,fi,numparticulas,viBAT,
        BestW,wi,NumAntElem);
    [wi2]=NewPosition(viBAT,wi);  %usuando la nueva velocidad
        calcula
    [wi2]=enrango(wi2);%verifica que el vector de pesos wi este
        dentro del rango 0 a 2pi
    [solutions,~,NormalizedDesiredSolutionVector]=costfunction(
        Nusuarios,U1,U2,U3,numparticulas,wi2); %evalua el vector de
         pesos wi en la funcion de costos
    [Bestpos2,BestFun2,~]=choosebest(solutions,
        NormalizedDesiredSolutionVector,wi);

   [wi]=UpdateInitialPosition(pulserate,NumAntElem,Loudness,
       numparticulas,BestW,wi);
   [wi]=enrango(wi);%verifica que el vector de pesos wi este
       dentro del rango 0 a 2pi
   [solutions,~,NormalizedDesiredSolutionVector]=costfunction(
       Nusuarios,U1,U2,U3,numparticulas,wi); %evalua el vector de
       pesos wi en la funcion de costos
   [Bestpos1,BestFun1,~]=choosebest(solutions,
       NormalizedDesiredSolutionVector,wi);

   [pulserate, Loudness]=UpdateLoudnessPulseRate(pulserate,
```

```
        Loudness,BestFun,numparticulas,initialpulserate,gamma,alpha,
        k);
25
26      [~, ~,wi,~,OutW]=UpdateBestSolution(BestFun2,Bestpos2,BestFun1,
        Bestpos1,wi2,wi);
27
28
29
30
31
32  end
33
34
35
36
37
38
39  function[err,AtascoCount,iter]=errorcalc(BestFun, err,
      NormalizedDesiredSolution,iter,AtascoCount)
40  previouserrorA=err;
41   err=sum(abs( BestFun-norm(NormalizedDesiredSolution)));
42   if previouserrorA==err
43       AtascoCount=AtascoCount+1;
44   else
45       AtascoCount=0;
46   end
47   iter=iter+1;
48  end
49
50
51  function [viBAT]=NewVelocity(aceleracion,theta,fi,numparticulas,
      viBAT,BestW,wi,NumAntElem)
52      viBAT=theta.*viBAT+aceleracion.*rand(numparticulas,NumAntElem
          ).*(repmat(BestW,numparticulas,1)-wi).*fi;
53  end
54
55  function [fi]=NewFrequency(fmax,fmin,numparticulas)
56      fi=fmin+(fmax-fmin).*rand(numparticulas,1);
```

```
57 end
58
59 function [wi]=UpdateInitialPosition(pulserate,NumAntElem,Loudness,
       numparticulas,BestW,wi)
60
61     aleator1=rand(numparticulas,NumAntElem);
62     I1=rand(numparticulas,1) < pulserate; %busca un valor aleatorio
           que sea menor a la tasa de pulsos
63     al=aleator1(I1,NumAntElem);
64     promloud=mean(Loudness);
65     wi(I1,:)=BestW+0.1*(al*2-1)*promloud; %se actualiza xi1 de
          acuerdo al valor aleatorio menor que la tasa de pulsos y el
          volumen promedio
66
67
68 end
69
70
71 function [pulserate, Loudness]=UpdateLoudnessPulseRate(pulserate,
       Loudness,BestFun,numparticulas,initialpulserate,gamma,alpha,t)
72
73      randomnumber2=rand(numparticulas,1);
74      I2=randomnumber2 > Loudness;
75      randomnumber3=rand(numparticulas,1);
76      I3=randomnumber3 < BestFun;
77      I4=and(I2,I3);  %Verifica que exista un valor mayor al volumen
            y menor al mejor obtenido luego de evaluar la funcion
78     %En caso que se cumpla la condicion anterior el vector r y A
          son
79     %actualizados para reemplazar ese valor por una tasa de ruido
          mas alta
80     %y un volumen menor
81
82      pulserate(I4)     = initialpulserate(I4)*(1-exp(-gamma*t));  %
          Aumenta cota r_0
83
84      Loudness(I4)      = alpha*Loudness(I4)      ;               %
          Disminuye
```

```
85
86
87  end
88
89
90  function [bestfun, bestpos,wi,wi2,OutW]=UpdateBestSolution(
        bestfun2,bestpos2,bestfun1,bestpos1,wi2,wi)
91
92  if bestfun2<=bestfun1 %se selecciona cual de los dos casos es
        mejor, si los valores obtenidos con xi1 o xi2
93          wi=wi;
94          bestpos=bestpos1;
95
96          bestfun=bestfun1;
97          OutW=wi;
98      else
99          wi=wi2;
100         bestpos=bestpos2;
101
102         bestfun=bestfun2;
103         OutW=wi2;
104     end
105
106
107
108
109
110 end
111
112 function[solutions,DesiredSolution,NormalizedDesiredSolutionVector
        ]=costfunction(Nusuarios,U1,U2,U3,numparticulas,wi) %funcion
        que evalua el vector de pesos y genera un diagrama de solucion
        con los usuarios.
113 global rang
114 gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo; %
        funcion gaussiana
115 amp=1000; %amplitud
116 width=rang; %ancho (para distancias cercanas es 6, distancias
```

```
    medianas 4.5, distancias lejanas 6)
cant=1800; %cantidad de puntos
desplazamiento=0; %desplazamiento de las campanas
thetar=[0 180];  %rango de theta
THETA=linspace(thetar(1),thetar(2),cant); %theta en grados
rad=pi/180; %radianes
NumAntElem=size(wi,2);
NumAntElem=sqrt(NumAntElem); %numero de elementos de la antenna
antdim=[NumAntElem NumAntElem]; %dimensiones de la antea
diselementos=[0.5 0.5]; %distancia entre elementos en la antenna
N=[NumAntElem, NumAntElem];
thetar=[0 180]*rad; %theta en radianes
THETARAD=linspace(thetar(1),thetar(2),cant);     % 1 x cant va
    desde 0 a pi en 1800 pasos
k=2*pi;
d=diselementos;
if Nusuarios==2  %caso si se tiene 2 usuarios
    usuarios=[U1 U2]; %ubicacion usuarios
    DesiredSolution = gaus(THETA,usuarios(1),width,amp,
        desplazamiento)+gaus(THETA,usuarios(2),width,amp,
        desplazamiento);%+gaus(THETA,t0(3),sig,amp,vo);
     %se genera la solucion deseada que sera utilizada como
        referencia (funcion gaussiana)
elseif Nusuarios==3
     usuarios=[U1 U2 U3] ; %ubicacion usuarios
     DesiredSolution = gaus(THETA,usuarios(1),width,amp,
        desplazamiento)+gaus(THETA,usuarios(2),width,amp,
        desplazamiento)+gaus(THETA,usuarios(3),width,amp,
        desplazamiento);
     %se genera la solucion deseada que sera utilizada como
        referencia (funcion gaussiana)
end

Psix = (((1:N(1))-1).*(k*d(1)*sin(THETARAD).*cos(0))');
Psiy = (((1:N(2))-1).*(k*d(2)*sin(THETARAD).*sin(0))');
for i=1:N(1)
    sumaPsi(N(1)*(i-1)+1:i*N(1),:)=Psix(i,:)+Psiy;
end
```

```
146 sumaPsi=exp(1i*sumaPsi); %64x1800 complex double;
147 ArrayFactor1            = exp(1i*wi)*sumaPsi;
148
149
150
151 %ArrayFactor1=faf2(exp(1i*wi),antdim,diselementos,THETARAD,0); %se
        calcula el factor de arreglo
152 NormalizedAF1=abs(ArrayFactor1); %valor absoluto del factor de
      arreglo
153  sumNAF1=sum(NormalizedAF1,2); %el valor total sumado de todos los
        elementos
154  NAF1=NormalizedAF1./sumNAF1; %Normalizacion del factor de arreglo
155  M = max(NAF1,[],2); %se encuentra el punto maximo del factor de
        arreglo normalizado
156   DesiredSolution1=repmat(DesiredSolution,numparticulas,1); %se
        genera matriz repetida de solucion deseada
157 NormalizedDesiredSolutionVector=DesiredSolution1.*M; %multiplico
      el punto mas alto por la solucion deseada(la solcucion deseada
      (desired solution) tiene una amplitud de mil y el punto maximo
      es como de 2x10-3, este producto se hace para obtener una
      solucion deseada que este en una misma escala que las
      soluciones que se van a ir propionendo)
158 prodv=NAF1.*DesiredSolution1; %multiplico la solucion deseada por
      el factor de arreglo normalizado
159 solutions=sqrt(sum(prodv.^2,2)); %calculo la norma del producto
      anterior
160
161
162 end
163
164
165
166 function [wi]=enrango(wi)
167 wi=mod(wi,2*pi);
168 end
169 function [wi]=initalize(NumAntElem,numparticulas)
170 LB=0*ones(1,NumAntElem);              % Limite inferior
171 UB=2*pi*ones(1,NumAntElem);           % Limite Superior
```

```
172 wmin=min(LB);                % w minimo
173 wmax=max(UB);                % w maximo
174
175
176
177 wi=(rand(numparticulas,NumAntElem))*(wmax-wmin)+wmin;
178 end
179
180 function [BestW,BestF,NormalizedDesiredSolution]=choosebest(
        solutions,NormalizedDesiredSolutionVector,wi)
181
182 [~,I] = max(solutions);
183 NormalizedDesiredSolution=NormalizedDesiredSolutionVector(I,:);
184 BestW =wi(I,:);
185 BestF=solutions(I);
186
187
188 end
189 function [wi]=NewPosition(viBAT,wi)
190  wi=wi+viBAT;
191 end
192
193
194 function Plotting(antdim,diselementos, BestW )
195 rad=pi/180;
196 antdim=[antdim antdim];
197 diselementos=[diselementos diselementos];
198
199 thetar=[0 180]*rad;
200 THETARAD=linspace(thetar(1),thetar(2),1800);     % 1 x cant va
        desde 0 a pi en 1800 pasos
201
202
203 AF=faf(exp(1i*BestW),antdim,diselementos,THETARAD,0);
204 AFnorma=abs(AF/sum(abs(AF)));
205 figure(9);
206 plot(AFnorma);
207
```

```
208
209 W=exp(1i*BestW);
210 Wbestpos=abs(W);                                  % Magnitudes
211 betabestpos=mod(angle(W)*180/pi,360);             % Angulos
212 nombre='PSO';
213
214 radtheta(W,antdim,diselementos,0,nombre)
215 % %----------------------Patron de Radiaci n en Theta Polares
    ----------------------
216  radthetapol(W,antdim,diselementos,0,nombre)
217
218
219
220
221 end
```

A.1.1.13. Código de algoritmo CUCKOO

```
1  function pout = SimpleCuckoo(p,pa,beta,Nusuarios,U1,U2,U3)
2
3
4  % This a minimalist Cuckoo Search Algorithm (CS) function
5  % Input parameters: % pin = solutions matrix;  gamm, lambda, pa
       and beta
6  % are parameters of CS.
7  % Output parameters: % p_out = solutions matrix. Saved is a
       cellarray with the new best
8  % solution {solution fitness}
9       popsize = size(p,1);
10      [fitness,~]=costfunction(Nusuarios,U1,U2,U3,popsize,p);
11      step = levy(p,beta); % Levy flight
12      Newcuckoos = p + step;
13
14      %[~,m] = find(abs(Newcuckoos(1,:))>1);  % Checking if any
           solution is out of bound
15      %if ~isempty(m)                          % If so, I put them at
            the border
16      %  Newcuckoos(1,[m]) = .99*sign(Newcuckoos(1,[m]));
17      %end
```

```
18      %[~,z] = find(abs(Newcuckoos(2,:))>1);
19      %if ~isempty(z)
20      %    Newcuckoos(2,[z]) = .99*sign(Newcuckoos(2,[z]));
21      %end
22
23      %%%%%%%%%% Evaluating new cuckoos %%%%%%%%%%
24      fitnew = costfunction(Nusuarios,U1,U2,U3,popsize,Newcuckoos);;
25      pinit=p;
26      %%%%%%%%%% Get a cuckoo randomly and insert the new into
           population if it's better
27      for mir = 1:size(p,1)
28          dice = randi(size(p,1));
29          if fitnew(mir)>fitness(dice)
30              p(dice,:) = Newcuckoos(mir,:);
31          end
32      end
33
34      %%%%%%%%%% sorting %%%%%%%%%%%%%
35      [fitness,~]=costfunction(Nusuarios,U1,U2,U3,popsize,p);
36      [~,ind] = sort(fitness,'descend');
37      p = p(ind,:);
38
39      %%%%%%%%%%%% Abandoning cuckoos %%%%%%%%%%
40      % Generate random cuckoos
41      Inserthem = 2*rand(size(p,1)-pa+1,size(p,2))-1;
42      % Insert them in the place of the worst solutions
43      p(pa:end,:) = Inserthem;
44
45      %%%%%%%%%% Sorting and insert best %%%%%%%%%%
46      [fitness,~]=costfunction(Nusuarios,U1,U2,U3,popsize,p);
47      [~,ind] = sort(fitness,'descend'); % sorting
48      p = p(ind,:); %p(:,end) = Saved{1}; % copying former
           generation best into current generation
49      pout = p;
50       [pout]=enrango(pout);
51  %   %%%%%%% Sorting again to save new best %%%%%%%%%
52   %  fitness = costfunction(p);
53   %  [~,ind] = sort(fitness,'descend'); % sorting
```

```
54    %a   p = p(:,ind); Saved{1} = p(:,1); Saved{2} = fitness(1); %
          Saving new best
55
56        function [wi]=enrango(wi)
57        wi=mod(wi,2*pi);
58        end
59
60  function[solutions,DesiredSolution,NormalizedDesiredSolutionVector
      ]=costfunction(Nusuarios,U1,U2,U3,numparticulas,wi) %funcion
      que evalua el vector de pesos y genera un diagrama de solucion
      con los usuarios.
61  global rang
62  gaus = @(x,mu,sig,amp,vo)amp*exp(-(((x-mu).^2)/(2*sig.^2)))+vo; %
      funcion gaussiana
63  amp=1000; %amplitud
64  width=rang; %ancho (para distancias cercanas es 6, distancias
      medianas 4.5, distancias lejanas 6)
65  cant=1800; %cantidad de puntos
66  desplazamiento=0; %desplazamiento de las campanas
67  thetar=[0 180];  %rango de theta
68  THETA=linspace(thetar(1),thetar(2),cant); %theta en grados
69  rad=pi/180; %radianes
70  NumAntElem=size(wi,2);
71  NumAntElem=sqrt(NumAntElem); %numero de elementos de la antenna
72  antdim=[NumAntElem NumAntElem]; %dimensiones de la antea
73  diselementos=[0.5 0.5]; %distancia entre elementos en la antenna
74  N=[NumAntElem, NumAntElem];
75  thetar=[0 180]*rad; %theta en radianes
76  THETARAD=linspace(thetar(1),thetar(2),cant);     % 1 x cant va
      desde 0 a pi en 1800 pasos
77  k=2*pi;
78  d=diselementos;
79  if Nusuarios==2  %caso si se tiene 2 usuarios
80      usuarios=[U1 U2]; %ubicacion usuarios
81      DesiredSolution = gaus(THETA,usuarios(1),width,amp,
          desplazamiento)+gaus(THETA,usuarios(2),width,amp,
          desplazamiento);%+gaus(THETA,t0(3),sig,amp,vo);
82       %se genera la solucion deseada que sera utilizada como
```

```
        referencia (funcion gaussiana)
elseif Nusuarios==3
    usuarios=[U1 U2 U3] ; %ubicacion usuarios
    DesiredSolution = gaus(THETA,usuarios(1),width,amp,
        desplazamiento)+gaus(THETA,usuarios(2),width,amp,
        desplazamiento)+gaus(THETA,usuarios(3),width,amp,
        desplazamiento);
    %se genera la solucion deseada que sera utilizada como
        referencia (funcion gaussiana)
end

Psix = (((1:N(1))-1).*(k*d(1)*sin(THETARAD).*cos(0))');
Psiy = (((1:N(2))-1).*(k*d(2)*sin(THETARAD).*sin(0))');
for i=1:N(1)
    sumaPsi(N(1)*(i-1)+1:i*N(1),:)=Psix(i,:)+Psiy;
end
sumaPsi=exp(1i*sumaPsi); %64x1800 complex double;
ArrayFactor1            = exp(1i*wi)*sumaPsi;



%ArrayFactor1=faf2(exp(1i*wi),antdim,diselementos,THETARAD,0); %se
    calcula el factor de arreglo
NormalizedAF1=abs(ArrayFactor1); %valor absoluto del factor de
   arreglo
 sumNAF1=sum(NormalizedAF1,2); %el valor total sumado de todos los
     elementos
 NAF1=NormalizedAF1./sumNAF1; %Normalizacion del factor de arreglo
 M = max(NAF1,[],2); %se encuentra el punto maximo del factor de
    arreglo normalizado
  DesiredSolution1=repmat(DesiredSolution,numparticulas,1); %se
    genera matriz repetida de solucion deseada
NormalizedDesiredSolutionVector=DesiredSolution1.*M; %multiplico
   el punto mas alto por la solucion deseada(la solcucion deseada
   (desired solution) tiene una amplitud de mil y el punto maximo
   es como de 2x10-3, este producto se hace para obtener una
   solucion deseada que este en una misma escala que las
   soluciones que se van a ir propionendo)
```

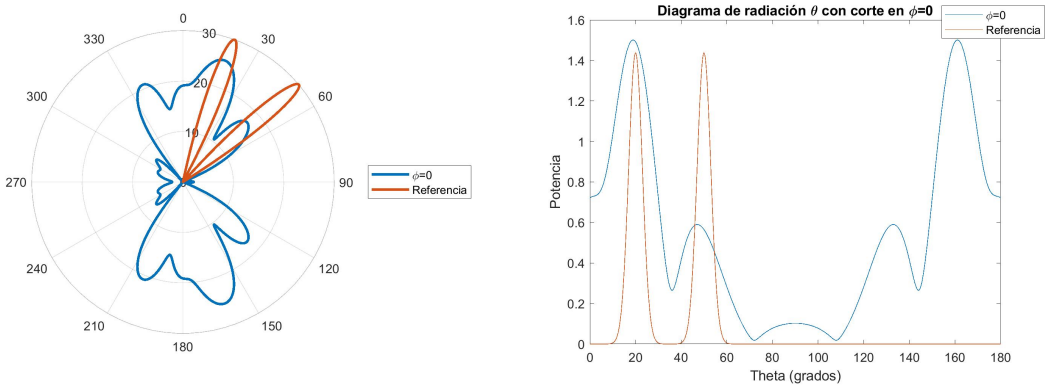
```
106 prodv=NAF1.*DesiredSolution1; %multiplico la solucion deseada por
    el factor de arreglo normalizado
107 solutions=sqrt(sum(prodv.^2,2)); %calculo la norma del producto
    anterior
108
109
110 end
111 end
```

A.1.1.14. Código Factor de arreglo

```
1  function AFV=faf2(w,N,d,t,p)
2  %Function Array Factor
3  %-------------------------------------DESCRIPCION
   ------------------------------------
4  % Factor de Arreglo Plano
5  %-------------------------------------INPUT
   ------------------------------------
6  %   - w          Coeficientes de exitacion      (vector)
7  %   - N          Cantidad de elementos      (vector)
8  %   - d          Distancia entre elementos   (vector)
9  %   - t          Theta Vector
10 %   - p          Phi Vector
11 %-------------------------------------OUTPUT
   ------------------------------------
12 %       - AF        Factor de arreglo          (Matriz)
13 %
   ------------------------------------------------------------------------------

14 k=2*pi;
15 AF=0;
16 [sizew,~]=size(w);
17 ii=0;
18 Psix = k*d(1)*sin(t).*cos(p);
19 Psix=repmat(Psix,N(1),1);
20 Psiy = k*d(2)*sin(t).*sin(p);
21 Psiy=repmat(Psiy,N(2),1);
22 m=[1:N(1)];
23 n=[1:N(2)];
```

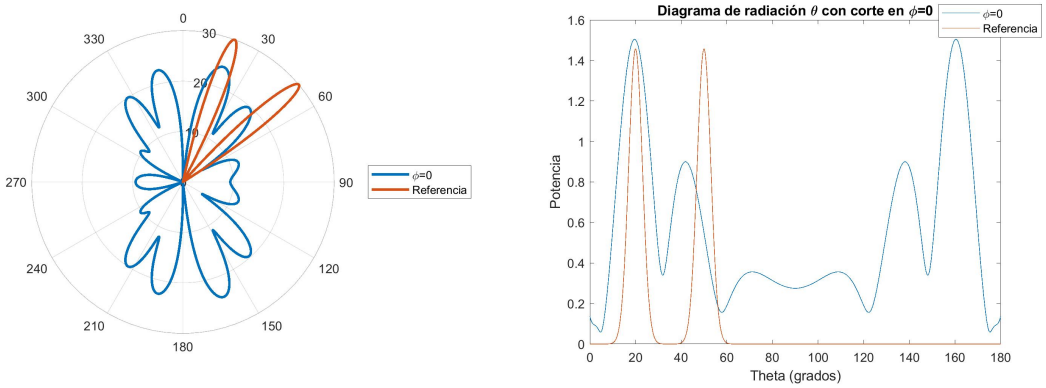
```
 a=transpose(m-1).*Psix;
 aa=exp(1i*a);
 b=transpose(n-1).*Psiy;
 bb=exp(1i*b);


for p=1:N(1)
 if p==1
     j=aa(p,:).*bb;

 else
 d1=aa(p,:).*bb;
 j=cat(1,j,d1);
end
end
i=1;
while i<sizew+1
pp=transpose(w(i,:));
AF=pp.*j;

sumi=sum(AF(1:(end-1),:));
AF=AF(end,:)+sumi;
AFV(i,:)=AF;
i=i+1;
end


% for ii=1:sizew
%          ii=ii+1
%          d=repmat(c(1),1,2)
%
%          AF=AF+w(ii,:).*exp(1i*((m-1).*Psix)).*exp(1i*((n-1).*
    Psiy));
% end
%
%
%% cambiar a un solo for 1 hasta 64
% end
```

A.1.2. Diagramas de Radiación



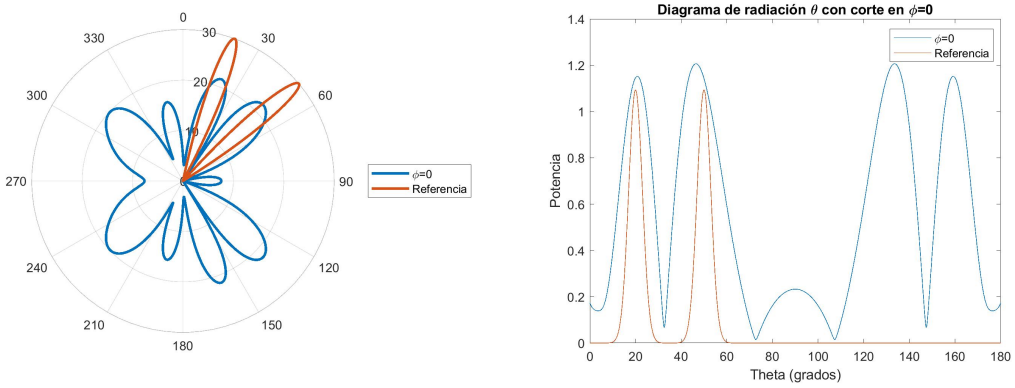
(a) Polares (b) Lineal

Figura A.1: Diagrama de Radiación 2 usuarios fijos θ de Cuckoo.



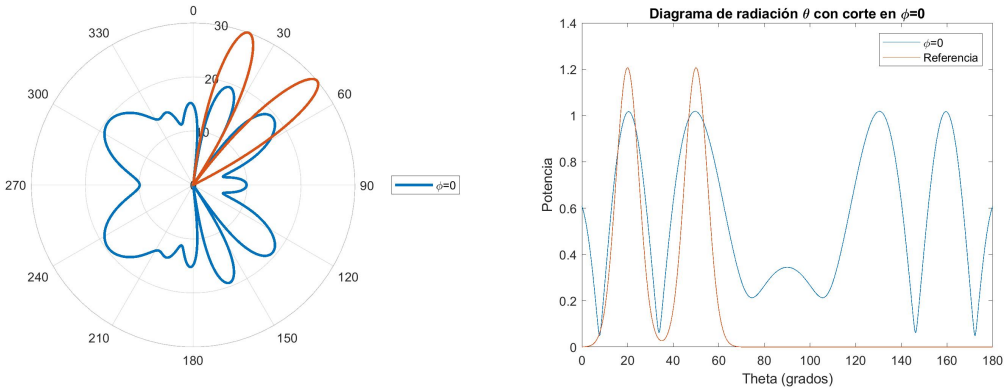
(a) Polares (b) Lineal

Figura A.2: Diagrama de Radiación Nómada θ de PSO.



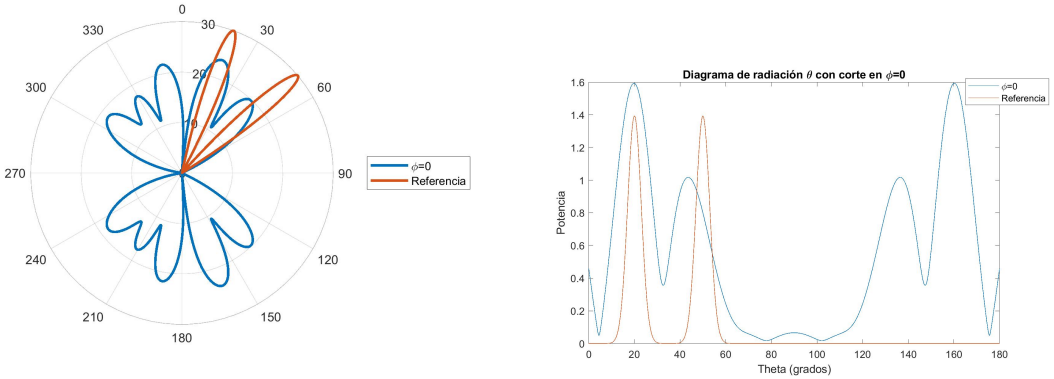
(a) Polares (b) Lineal

Figura A.3: Diagrama de Radiación 2 usuarios fijos θ de MAS.



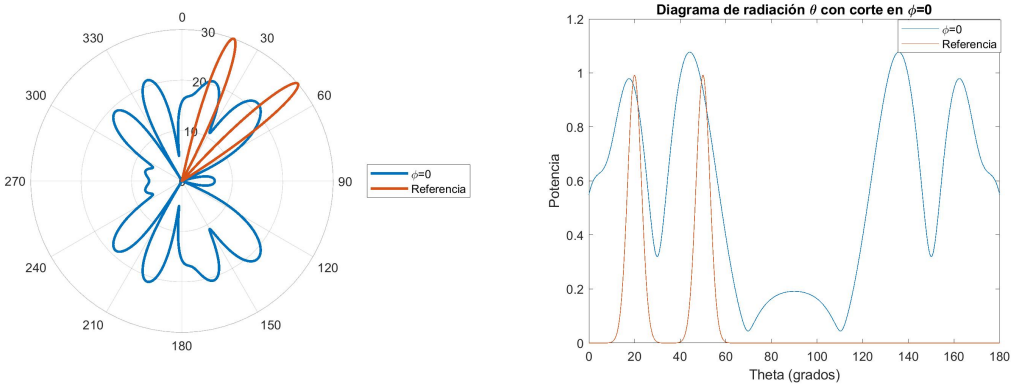
(a) Polares (b) Lineal

Figura A.4: Diagrama de Radiación 2 usuarios fijos θ de RBE



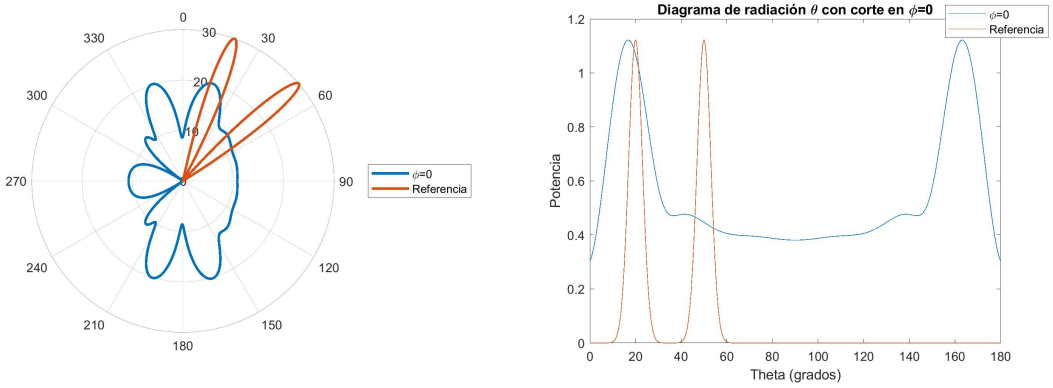
(a) Polares (b) Lineal

Figura A.5: Diagrama de Radiación 2 usuarios fijos θ de BAT



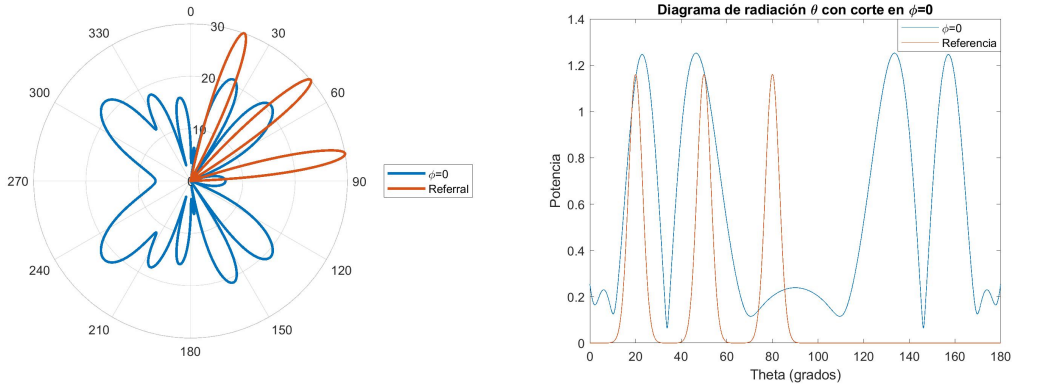
(a) Polares (b) Lineal

Figura A.6: Diagrama de Radiación 2 usuarios fijos θ de GA



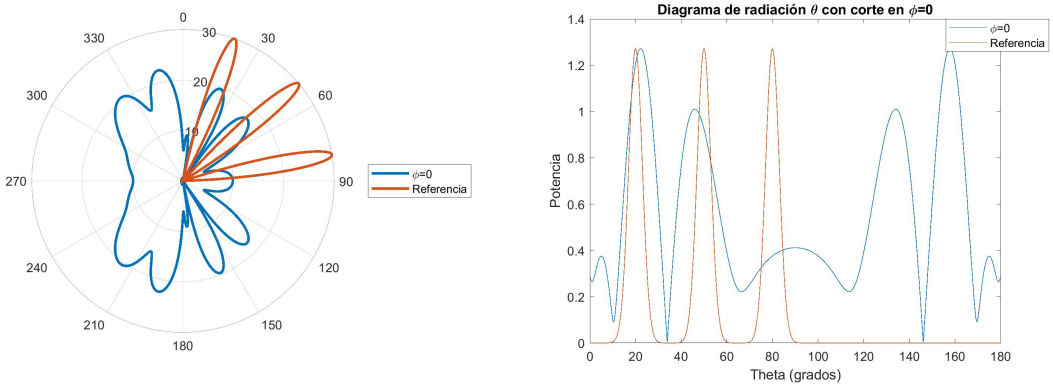
(a) Polares (b) Lineal

Figura A.7: Diagrama de Radiación 2 usuarios fijos θ de SA



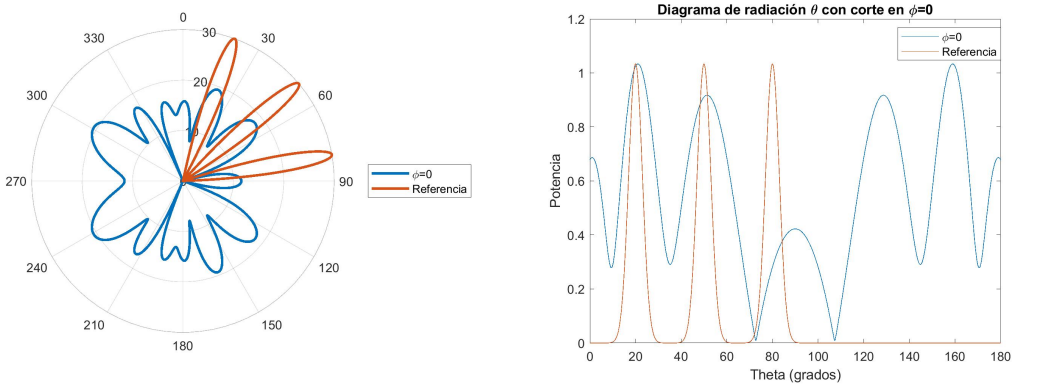
(a) Polares (b) Lineal

Figura A.8: Diagrama de Radiación 3 usuarios fijos θ de PSO



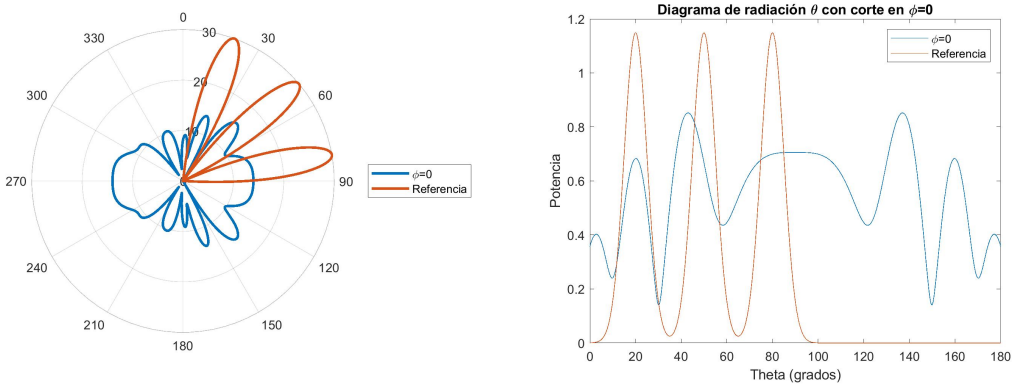
(a) Polares (b) Lineal

Figura A.9: Diagrama de Radiación 3 usuarios fijos θ de CUCKOO



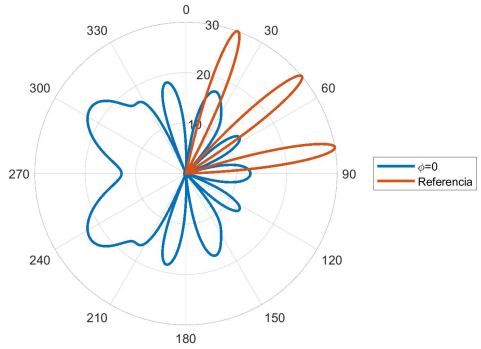
(a) Polares (b) Lineal

Figura A.10: Diagrama de Radiación Nómada θ de BAT.

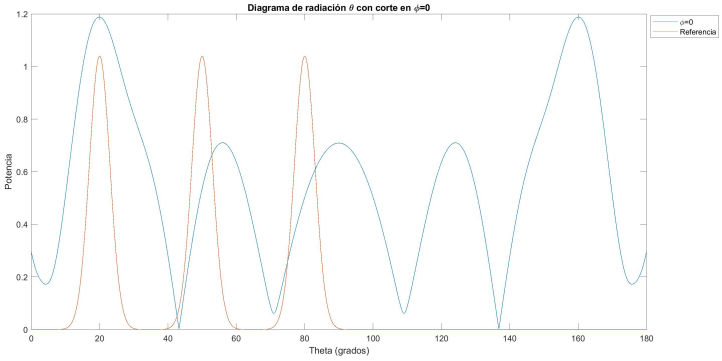


(a) Polares (b) Lineal

Figura A.11: Diagrama de Radiación Nómada θ de MAS.

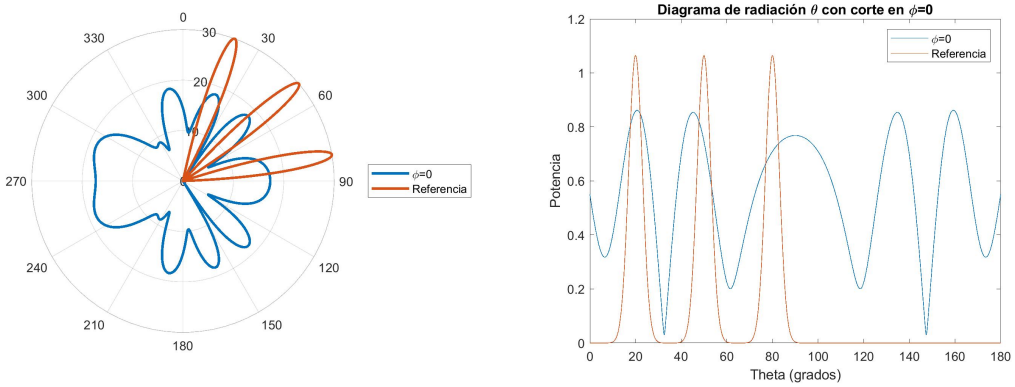


(a) Polares



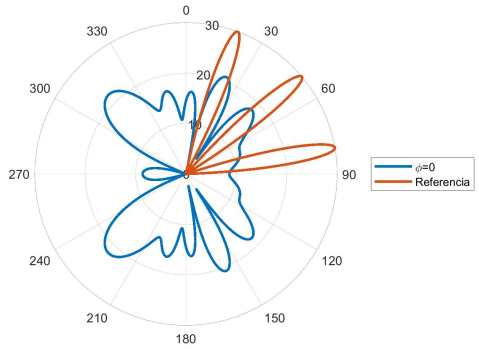
(b) Lineal

Figura A.12: Diagrama de Radiación 3 usuarios fijos θ de GA

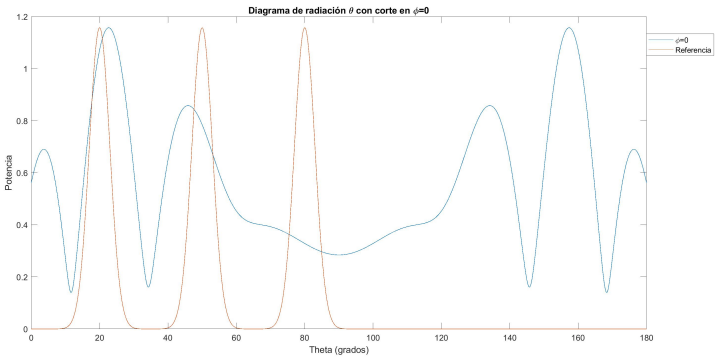


(a) Polares (b) Lineal

Figura A.13: Diagrama de Radiación Nómada θ de RBE.

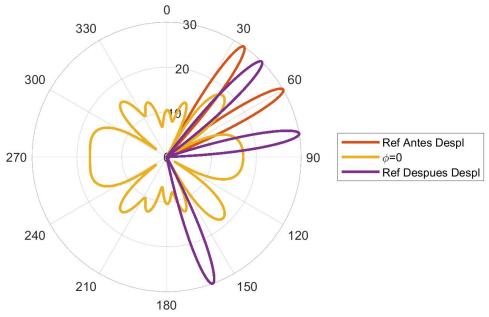


(a) Polares

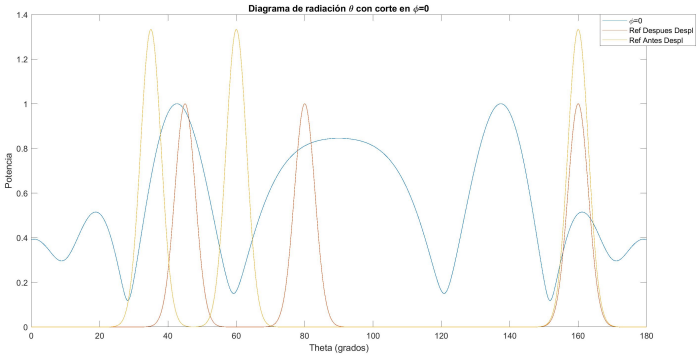


(b) Lineal

Figura A.14: Diagrama de Radiación 3 usuarios fijos θ de SA

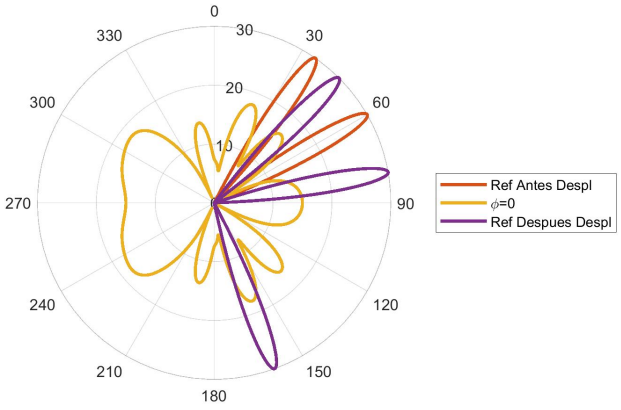


(a) Polares

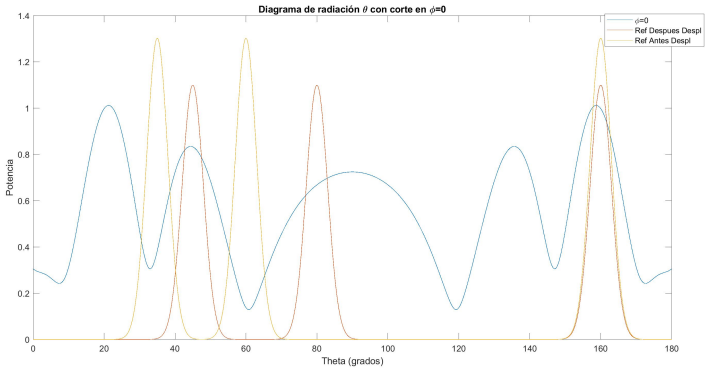


(b) Lineal

Figura A.15: Diagrama de Radiación 3 usuarios Nómadas θ de BAT.

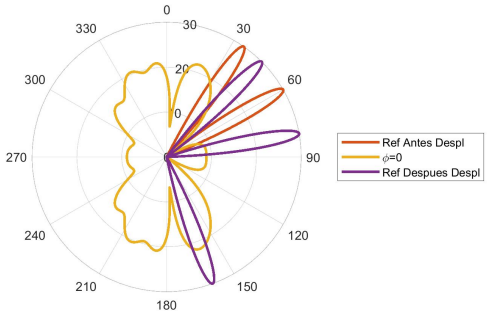


(a) Polares

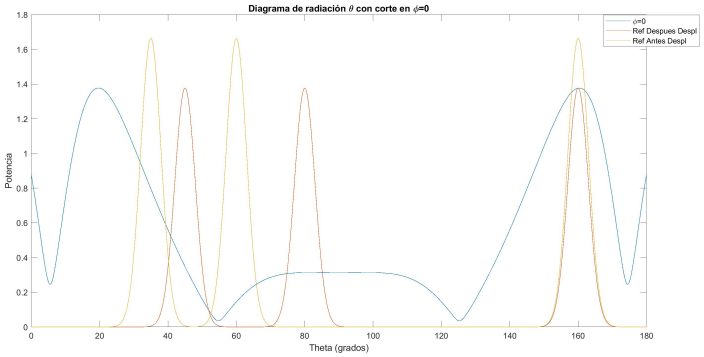


(b) Lineal

Figura A.16: Diagrama de Radiación 3 usuarios Mixtos θ de MAS

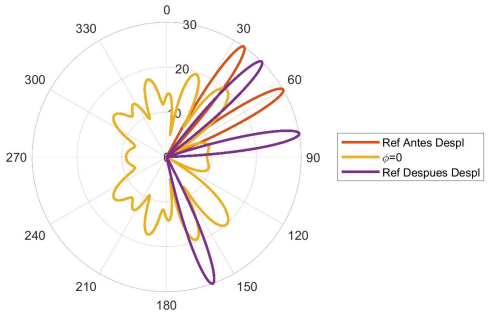


(a) Polares

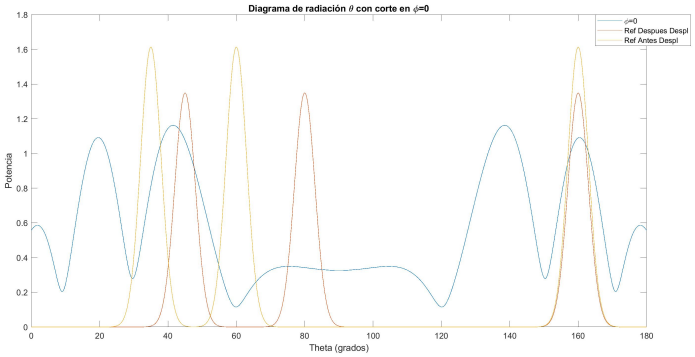


(b) Lineal

Figura A.17: Diagrama de Radiación 3 usuarios fijos θ de PSO

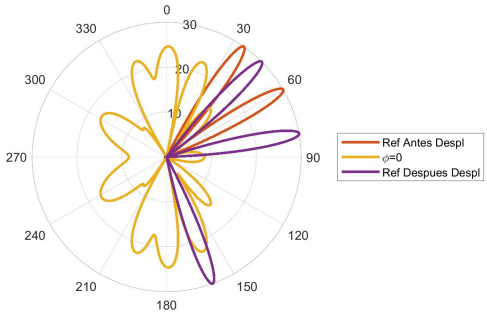


(a) Polares

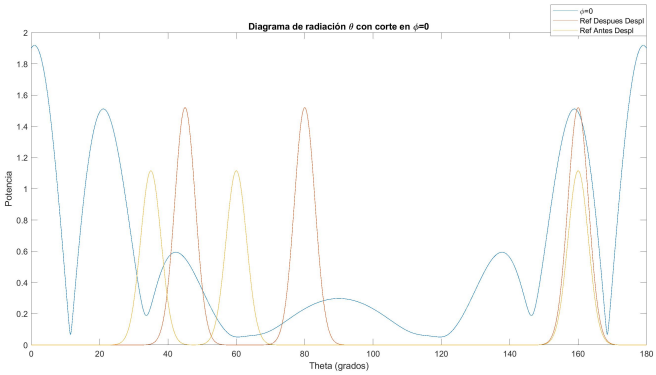


(b) Lineal

Figura A.18: Diagrama de Radiación 3 usuarios fijos θ de GA

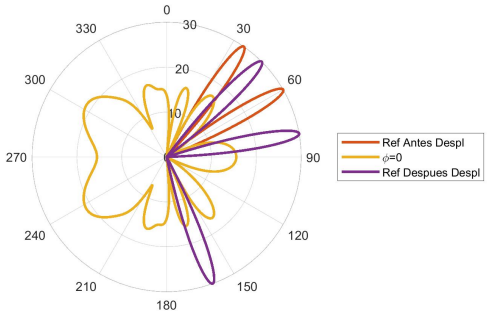


(a) Polares

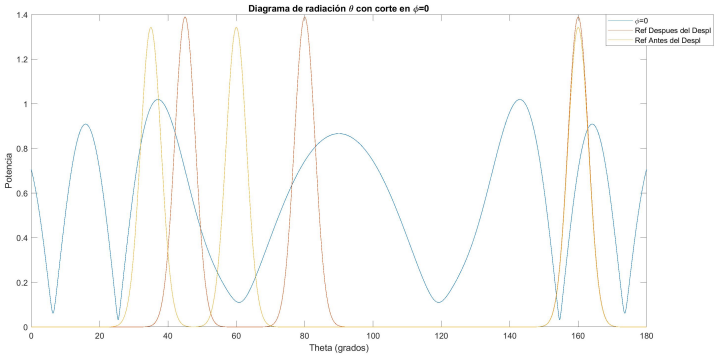


(b) Lineal

Figura A.19: Diagrama de Radiación 3 usuarios fijos θ de CUCKOO

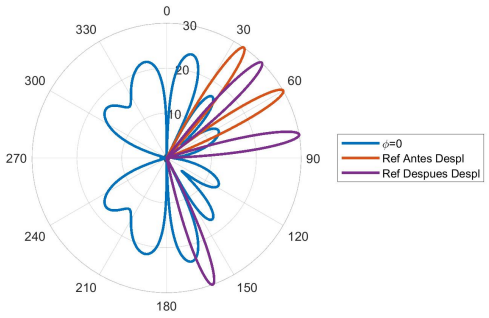


(a) Polares

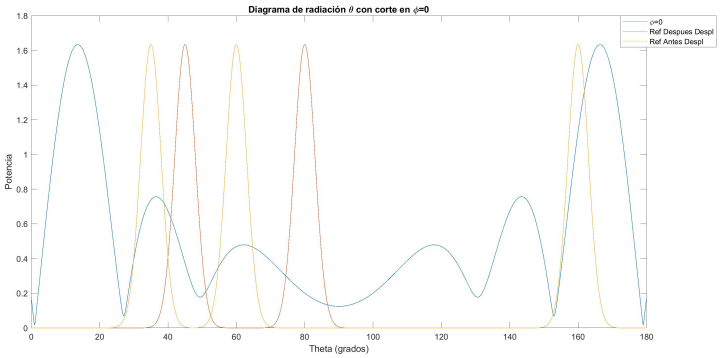


(b) Lineal

Figura A.20: Diagrama de Radiación 3 usuarios fijos θ de RBE

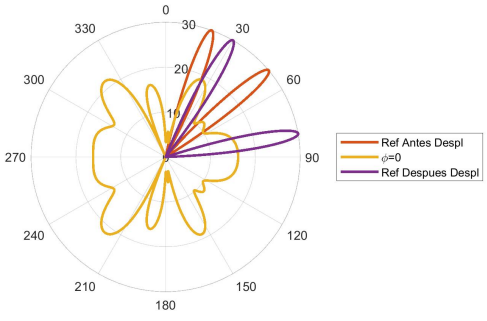


(a) Polares

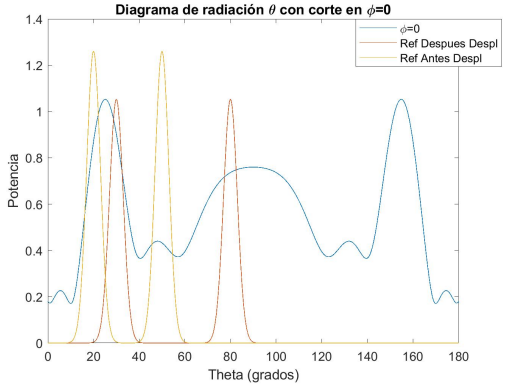


(b) Lineal

Figura A.21: Diagrama de Radiación 2 usuarios mixtos θ de SA

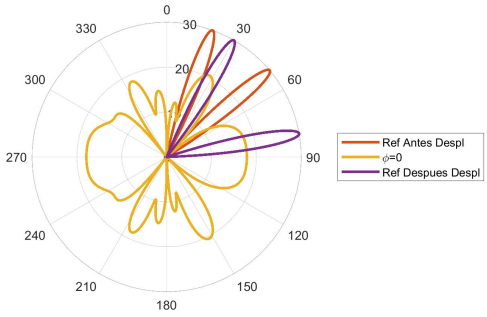


(a) Polares

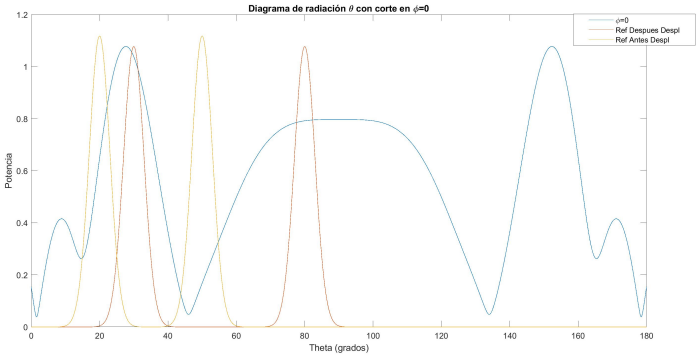


(b) Lineal

Figura A.22: Diagrama de Radiación 2 usuarios Nómadas θ de CUCKOO

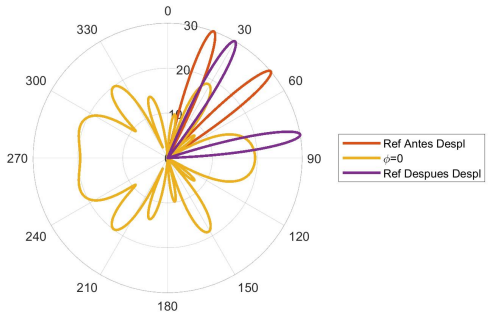


(a) Polares

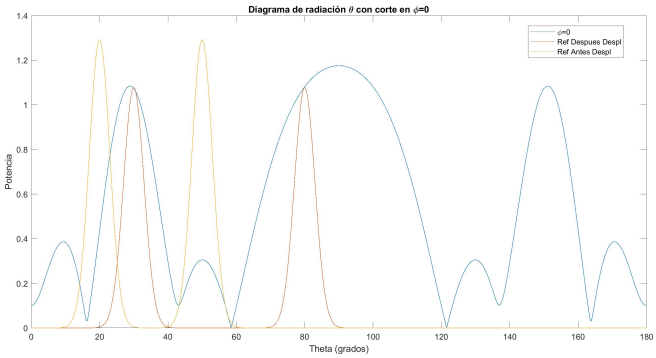


(b) Lineal

Figura A.23: Diagrama de Radiación 2 usuarios Nómadas θ de BAT.

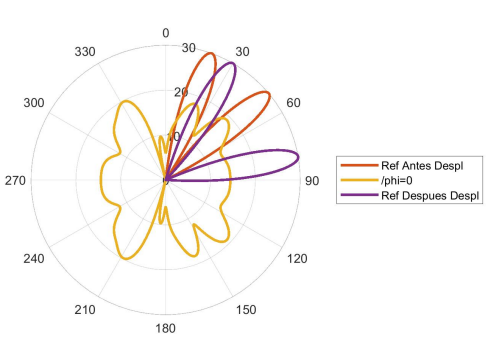


(a) Polares

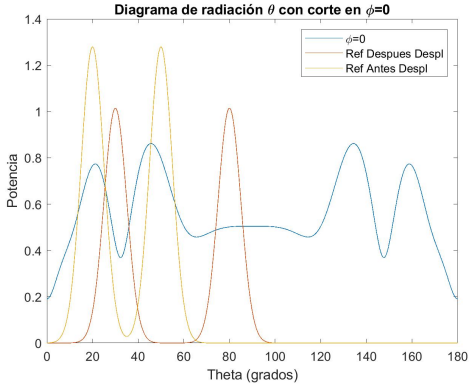


(b) Lineal

Figura A.24: Diagrama de Radiación 2 usuarios Nómadas θ de MAS.

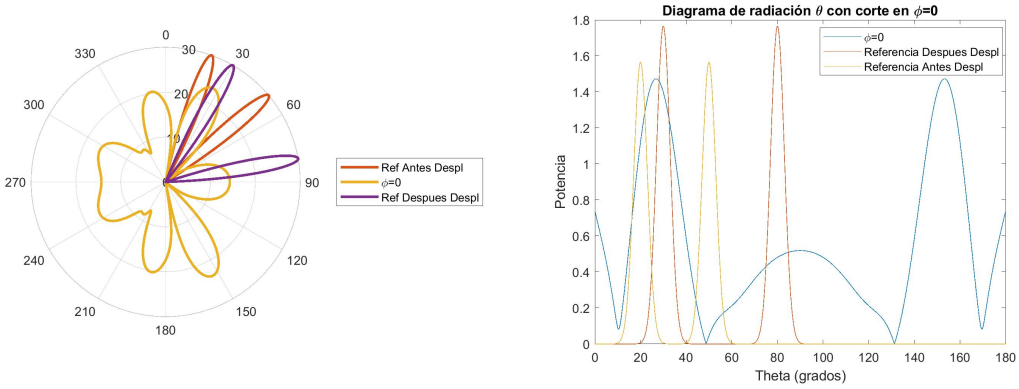


(a) Polares



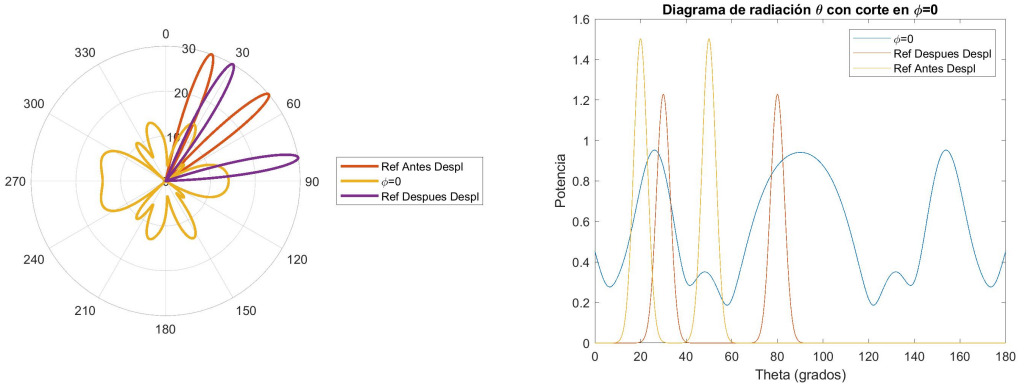
(b) Lineal

Figura A.25: Diagrama de Radiación 2 usuarios Nómadas θ de GA.



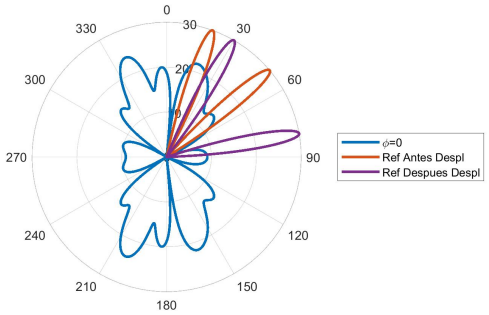
(a) Polares (b) Lineal

Figura A.26: Diagrama de Radiación 2 usuarios Nómada θ de PSO.

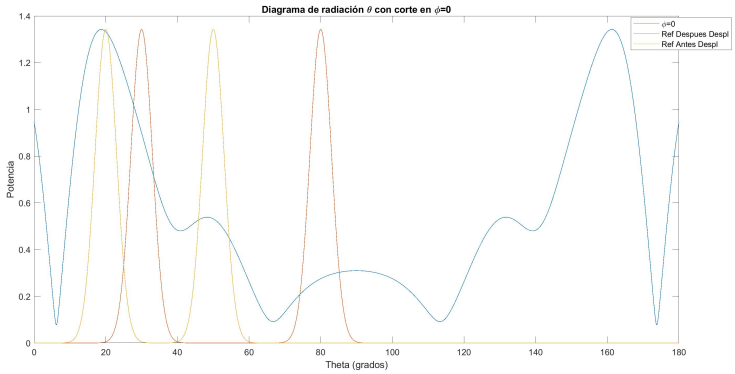


(a) Polares (b) Lineal

Figura A.27: Diagrama de Radiación 2 usuarios Nómadas θ de RBE.



(a) Polares



(b) Lineal

Figura A.28: Diagrama de Radiación 2 usuarios Nómadas θ de SA.

Bibliografía

[1] A. Balanis, *Antenna Theory: Analysis and Design*, 4th ed. Hoboken, New Jersey: Jon Wiley Sons, 2016.

[2] E. Björnson, J. Hoydis, and L. Sanguinetti, "Massive MIMO Networks: Spectral, Energy, and Hardware Efficiency," *Signal Procesing*, vol. 11, no. 3-4, pp. 154–655, 2017.

[3] P. Newson, H. Parekh, and H. Matharu. (2018) Realizing 5g new radio massive mimo systems. [Online]. Available: https://www.ednasia.com/realizing-5g-new-radio-massive-mimo-systems/

[4] I. M. N. R. e. a. Ali, E., "Beamforming techniques for massive mimo systems in 5g: overview, classification, and trends for future research," *Frontiers Inf Technol Electronic Eng*, no. 18, pp. 753–772, 2017.

[5] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

[6] T. Ting, X.-S. Yang, S. Cheng, and K. Huang, *Recent Advances in Swarm Intelligence and Evolutionary Computation, Studies in Computational Intelligence*. Switzerland: Springer International Publishing, 2015.

[7] E. Talbi, "A Taxonomy of Hybrid Metaheuristics," *Journal of Heuristics*, vol. 1, no. 8, pp. 541–564, 2002.

[8] M. de Tecnologías de la Información y las Comunicaciones. (2020) BoletÍn trimestral de las tic septiembre de 2020. [Online]. Available: https://colombiatic.mintic.gov.co/679/articles-151338_archivo_pdf

[9] L. Lu, G. Y. Li, A. L. Swindlehurst, A. Ashikhmin, and R. Zhang, "An overview of massive mimo: Benefits and challenges," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 742–758, 2014.

[10] S. Chen, S. Sun, Q. Gao, and X. Su, "Adaptive Beamforming in TDD-Based Mobile Communication Systems: State of the Art and 5G Research Directions," *IEEE Wireless Communications*, vol. 23, no. 6, pp. 81–87, 2016.

[11] K. Lee, In Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015.

[12] M. de Tecnologías de la Información y las Comunicaciones. (2013) BoletÍn trimestral de las tic septiembre de 2013. [Online]. Available: https://colombiatic.mintic.gov.co/679/articles-3853_archivo.pdf

[13] H.-H. Sung, M. Jacinto, K. Jat, and G. Dousson. (2016) Determining network congestion based on target user throughput. [Online]. Available: https://patents.google.com/patent/US9867080B2/en

[14] S. Bellofiore, J. Foutz, C. A. Balanis, and A. S. Spanias, "Smart-antenna system for mobile communication networks .Part 2. Beamforming and network throughput," *IEEE Antennas and Propagation Magazine*, vol. 44, no. 4, pp. 106–114, 2015.

[15] Z. Ning, Q. Song, L. Guo, and A. Jamalipour, "Throughput improvement by network coding and spatial reuse in wireless mesh networks," *IEEE Global Communications Conference (GLOBECOM)*, pp. 4572–4577, 2013.

[16] E. Fishler, A. Haimovich, R. S. Blum, L. J. Cimini, D. Chizhik, and R. A. Valenzuela, "Spatial diversity in radars—models and detection performance," *IEEE Transactions on signal processing*, vol. 54, no. 3, pp. 823–838, 2006.

[17] I. Lakkis. (2007) Beamforming in mimo systems. [Online]. Available: https://patents.google.com/patent/US7916081B2/en

[18] M. Passoja. (2018) 5g nr: Massive mimo and beamforming – what does it mean and how can i measure it in the field? [Online]. Available: https://www.rcrwireless.com/20180912/5g/5g-nr-massive-mimo-and-beamforming-what-does-it-mean-and-how-can-i-measure-it-in-the-field

[19] E. J. Black, "Holographic beam forming and mimo," *PIVOTAL COMMWARE*, 2017.

[20] A. Gandomi, X. Yang, and A. Talatahari, "Bat algorithm for constrained optimization tasks," *Neural Comput. Appl*, vol. 22, no. 6, pp. 1239–1255, 2013.

[21] X.-S. Yang, *Nature-Inspired Optimization Algorithms*, 1st ed. 225 Wyman Street, Waltham, MA 02451, USA: Elsevier Inc, 2014.

[22] V. Osuna-Enciso, E. Cuevas, and H. Sossa, "A comparison of nature inspired algorithms for multi-threshold image segmentation," *Expert Syst. Appl.*, vol. 40, no. 4, pp. 1213–1219, 2013.

[23] A. Kellner, "Multi-objective Ant Colony Optimisation in Wireless Sensor Networks," *Nature-Inspired Computing and Optimization, Modeling and Optimization in Science and Technologies 10*, vol. 10, no. 4, pp. 51–78, 2017.

[24] J. Polanco, *Beamforming de arreglos planos MIMO Masivo usando optimización meta-heurística.* Colombia: Pontificia Universidad Javeriana Cali, 2020.

[25] Huawei. (2019) World's first 8x8 mimo outdoor cpe commercialized for maximum lte network value, assuring gb/s experience. [Online]. Available: https://www.huawei.com/en/news/2019/10/world-first-8x8-mimo-outdoor-cpe-b3368

[26] Maxwave. (2018) Lte/wifi mimo antenna for transportation. [Online]. Available: https://www.maxtena.com/products/5g-4g-lte-wifi-mimo-antennas/4x4-lte-wifi-mimo-antenna/

[27] organización de las naciones unidas. (2015) Objetivos y metas de desarrollo sostenible. [Online]. Available: https://www.un.org/sustainabledevelopment/es/infrastructure/

[28] J. Zhang, Y. Wei, E. Bjornson, Y. Han, and X. Li, "Spectral and energy efficiency of cell-free massive mimo systems with hardware impairments," *9th International Conference on Wireless Communications and Signal Processing (WCSP)*, vol. 9, pp. 1–6, 2017.

[29] M. de Tecnologías de la Información y las Comunicaciones. (2019) La mitad de colombia no tiene internet. [Online]. Available: https://www.mintic.gov.co/portal/inicio/Sala-de-Prensa/MinTIC-en-los-Medios/100837:La-mitad-de-Colombia-no-tiene-internet

[30] gobierno de Colombia. (2018) Plan nacional de desarrollo 2018-2022. [Online]. Available: https://colaboracion.dnp.gov.co/CDT/Prensa/Resumen-PND2018-2022-final.pdf

[31] C. E. para las telecomunicaciones. (2020) The commission adopts implementing regulation to pave the way for high capacity 5g network infrastructure. [Online]. Available: https://ec.europa.eu/digital-single-market/en/news/commission-adopts-implementing-regulation-pave-way-high-capacity-5g-network-infrastructure

[32] C. F. para las comunicaciones. (2021) America's 5g future. [Online]. Available: https://www.fcc.gov/5G

[33] M. de tecnologías de la información y las comunicaciones. (2019) Plan 5g. [Online]. Available: https://www.mintic.gov.co/portal/604/articles-101369_plan_5g_v20190626.pdf

[34] R. N. Kostoffa, P. Herouxb, M. Aschnerc, and A. Tsatsakis, "Adverse health effects of 5G mobile networking technology under real-life conditions," *Toxicology Letters*, no. 323, pp. 35–40, 2020.

[35] X.-S. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*, 1st ed. Jon Wiley Sons, 2011.

[36] S. Jayaprakasam, S. Rahim, and C. Leow, "PSOGSA-Explore: A new hybrid metaheuristic approach for beampattern optimization in collaborative beamforming," *Applied Soft Computing*, vol. 30, pp. 229–237, 2015.

[37] A. Magdy, M. Osama, H. EL-Ghandour, and F. Hamed, "Performance Enhancement for Adaptive Beam-Forming Application Based Hybrid PSOGSA Algorithm," *Journal of Electromagnetic Analysis and Applications*, vol. 7, no. 4, pp. 1–8, 2015.

[38] C. Blum, A. R. M. Blesa, and M. Sampels, *Hybrid Metaheuristics:An Emerging Approach to Optimization.* Berlin, Alemania: Springer, 2008.

[39] G. Raidl, "A Unified View on Hybrid Metaheuristics," *Hybrid Metaheuristics*, no. 4030, pp. 1–11, 2006.

[40] G. Ram, P. S. Pal, D. Mandal, R. Kar, and S. P. Ghosal, "Social emotional optimization algorithm for beamforming of linear antenna arrays," *TENCON 2014 - 2014 IEEE Region 10 Conference*, pp. 1–5, 2014.

[41] W. Long, X. Liang, Y. Huang, and Y. Chen, "An effective hybrid cuckoo search algorithm for constrained global optimization," *Neural Computation Applications*, vol. 25, no. 12, pp. 911–926, 2014.

[42] P. Rocca and R. L. Haupt, "Biologically inspired optimization of antenna arrays," *ACES JOURNAL*, vol. 29, no. 12, pp. 1047–1060, 2014.

[43] D. Whitley, S. Rana, and R. Heckendorn, *Island model genetic algorithms and linearly separable problems.* Colorado, USA: Lecture Notes in Computer Science, 2006.

[44] TP-Link. (2021) Tl-wa7510n information site. [Online]. Available: https://www.tp-link.com/ar/business-networking/outdoor-radio/tl-wa7510n/

[45] U. Networks. (2018) Nanostation ac datasheet. [Online]. Available: https://dl.ubnt.com/datasheets/NanoStation_AC/NanoStation_AC_DS.pdf

[46] R. Padovani. (2014) Antenna radiation diagram plotter. [Online]. Available: https://sourceforge.net/p/ardp/wiki/Home/