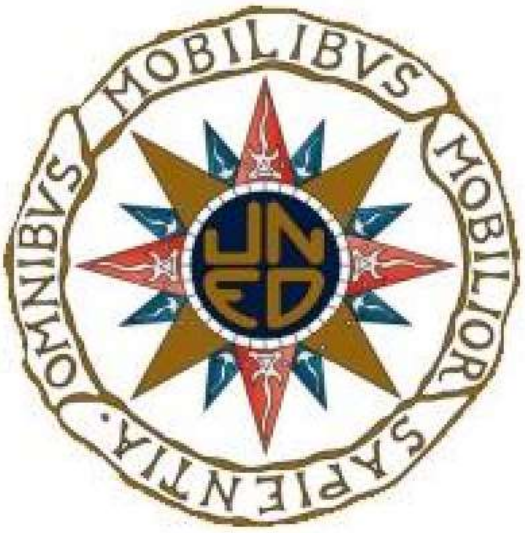


Máster en Ingeniería de Sistemas y Control

Trabajo de Fin de Máster

Optimización de trayectorias de búsqueda de UAVs mediante algoritmos evolutivos diferenciales

Alumna: Elena Núñez Torreiro

Directores: Eva Besada Portas y José Antonio López Orozco

Curso 2017-2018

Convocatoria: Septiembre 2018

Autorización

Autorizo a la Universidad Complutense y a la Universidad Nacional de Educación a Distancia a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado: Elena Núñez Torreiro

Resumen

Son numerosos los estudios relacionados con la aplicación de diversas técnicas heurísticas para la generación y planificación de trayectorias de UAVs (Unmanned Aerial Vehicle, vehículo aéreo no tripulado)

El presente Trabajo Fin de Master (TFM) tiene como objetivo principal el estudio sobre el comportamiento de los algoritmos evolutivos diferenciales aplicados a este problema.

La mayor parte de los estudios se centran en la dinámica del UAV, es decir, en obtener la secuencia de acciones, las órdenes de control, que determinarán cómo ha de moverse el UAV por el área de búsqueda. El aspecto diferencial de este trabajo es plantear un enfoque diferente, las soluciones serán los puntos de paso (waypoints) que conforman la trayectoria en nuestro mapa de búsqueda.

Será objeto del estudio analizar distintas parametrizaciones del algoritmo, considerando diferentes escenarios, tipos de perturbación, técnicas de selección de la base del algoritmo y finalmente evaluar su comportamiento o dependencia operacional con los escenarios de búsqueda.

Los resultados obtenidos nos permiten identificar las parametrizaciones que mejor funcionan, en general, en los diferentes escenarios propuestos; si bien, debemos concluir que sí existe cierta dependencia, lo que implica la necesidad de tener un conocimiento previo del área de búsqueda para obtener los mejores resultados.

Palabras clave

Planificación de trayectorias - UAV - Differential Evolution – Waypoints – Búsqueda de objetivos

Índice

Índice de figuras

Índice de tablas

1. Introducción

El término UAV es el acrónimo de Unmanned Aerial Vehicle (vehículo aéreo no tripulado). Como su nombre indica, se denominan así a las aeronaves que no precisan piloto a bordo.

Dichas aeronaves pueden ser dirigidas por control remoto, volar autónomamente según un plan de vuelo pre-programado o emplear sistemas dinámicos más complejos, que incluyen otros elementos como estaciones terrestres y enlaces de comunicaciones, lo que ha llevado a acuñar el término como "sistema de aeronave no tripulada", UAS (Unmanned Aerial System) [1].

El origen de este tipo de vehículos tuvo inicialmente fines militares y en este contexto se define una primera clasificación OTAN de los mismos [2], teniendo en cuenta el peso y las condiciones de operación de la aeronave, como se muestra a continuación:

Class	Category	Normal employment	Normal Operating Altitude	Normal Mission Radius	Primary Supported Commander	Example platform
CLASS I (less than 150 kg)	SMALL >20 KG	Tactical Unit (employs launch system)	Up to 5K ft AGL	50 km (LOS)	BN/Regt, BG	Hermes 90 Luna
	MINI 2-20 kg	Tactical Sub-unit (manual launch)	Up to 3K ft AGL	25 km (LOS)	Coy/Sqn	Aladin DH3 DRAC Eagle Raven Scan Skylark Strix T-Hawk
	MICRO <2 kg	Tactical PI, Sect, Individual (single operator)	Up to 200 ft AGL	5 km (LOS)	PI, Sect	Black Widow
CLASS II (150 kg to 600 kg)	TACTICAL	Tactical Formation	Up to 10,000 ft AGL	200 km (LOS)	Bde Comd	Aerostar Hermes 450 iView 250 Ranger Sperwer
CLASS III (more than 600 kg)	Strike/ Combat	Strategic/National	Up to 65,000 ft	Unlimited (BLOS)	Theater COM	
	HALE	Strategic/National	Up to 65,000 ft	Unlimited (BLOS)	Theater COM	Global Hawk
	MALE	Operational/theater	Up to 45,000 ft MSL	Unlimited (BLOS)	JTF COM	Predator B Predator A Harfang Heron Heron TP Hermes 900

Figura 1.Clasificación NATO UAS-Septiembre 2009 JCGUAV

Fuente: Revista Bit Agosto 2017 [2]

Pero en la actualidad son muchas sus aplicaciones civiles y comerciales[3] [4]. Debido a su elevada capacidad para acceder a lugares complejos o peligrosos para el ser humano, son de especial utilidad en numerosas aplicaciones entre las que destacan las tareas de búsqueda y rescate, vigilancia, fotografía y filmografía, investigación, cartografía, entrega de paquetes.

En la siguiente figura se muestran dos ejemplos de UAVs para aplicaciones diferentes, el de la izquierda es un vehículo aéreo no tripulado militar para labores de vigilancia y el de la derecha un pequeño drone comercial para uso recreativo.

Global Hawk

Uso vigilancia aérea militar

DJI Phantom

Uso comercial y fotografía aérea recreativa

Figura 2.Tipos de UAVs

Fuentes: www.theuav.com [3], www.dji.com [4]

Los primeros UAV eran versiones avanzadas de aviones controlados por radio, dotados de sistemas de control que llevaban a cabo tareas de bajo nivel como funciones de navegación simples, control de velocidad, estabilización de trayectorias y seguimiento de puntos de referencia preestablecidos. Pero en la actualidad la tendencia es dotar de una mayor autonomía a los vehículos, entendiendo autonomía como la capacidad de tomar decisiones sin intervención humana. En este sentido son muchos los ámbitos que deben desarrollarse para que los UAVs alcancen un nivel de autonomía elevado. Entre los mismos destacan:

- Análisis y combinación de información de diferentes sensores para su uso a bordo del vehículo

- Coordinación entre múltiples agentes y tácticas cooperativas, para maximizar las probabilidades de éxito en cualquier escenario de misión dado.
- Generación de trayectorias y planificación de rutas, determinando las maniobras de control óptimas o cumpliendo con ciertos objetivos y limitaciones, como obstáculos.

De todas las líneas de trabajo mencionadas en este Trabajo Fin de Master (TFM) nos centraremos en la última: la generación y planificación de trayectorias. Sin embargo, no es objeto de este trabajo ahondar en los numerosos métodos existentes para la resolución de problemas relacionados con la planificación de trayectorias y las numerosas experiencias reales en las que son de aplicación [5] [6] [7] [8] [9] [11] [12], si no centrarnos en un tipo de problema específico: la planificación de trayectorias de búsqueda de objetivos mediante algoritmos evolutivos diferenciales.

Este problema constituye un reto tecnológico en sí mismo, ya que supone resolver un problema probabilístico en el que se debe optimizar el tiempo esperado y la probabilidad de detección de los objetivos que se buscan [10] [13] [16] [17]. De hecho, se trata de un campo de investigación en auge, porque no hay un único algoritmo que proporcione fiabilidad, robustez, bajo coste computacional, etc. para diferentes escenarios y casuísticas.

Como este tipo de problemas no pueden ser resueltos de forma óptima en un tiempo polinomial, son NP duros [20], emplearemos técnicas heurísticas que nos proporcionarán soluciones aceptablemente buenas en un tiempo razonable. Pero todavía queda mucho camino por recorrer, ya que el objetivo final es conseguir que el UAV sea autónomo en escenarios altamente complejos, salvando obstáculos, evitando amenazas, modificando su trayectoria online según el entorno cambiante en el que se encuentre, coordinándose con otros agentes, etc.

Existen numerosos trabajos siguiendo esta línea de investigación: en su tesis, Pablo Lanillos [19] propone tres soluciones diferentes basadas en programación con restricciones, en métodos de optimización de entropía cruzada [17] y en algoritmos basados en el método del gradiente [18], y en sus TFMs Judith Manso [21] emplea algoritmos basados en redes neuronales, Francisco Javier Yáñez [23] utiliza algoritmos genéticos y optimización bayesiana, Fernando Pilar Arce [13] resuelve el problema mediante programación genética con el objeto de obtener un algoritmo que facilite soluciones genéricas para un espectro más amplio de escenarios, y Francisco Javier

Huertos [22] emplea algoritmos de colonias de hormigas. Además, los recientes trabajos publicados por Sara Perez-Carabaza *et al.* [Perez-Carabaza 2016, 2017] resuelven versiones realistas del problema mediante algoritmos genéticos o algoritmos de colonias de hormigas.

En todos los trabajos previamente mencionados, las soluciones obtenidas facilitan la secuencia de acciones que permite a los agentes (normalmente UAVs) desplazarse por el área de búsqueda mientras hacen las observaciones. Es decir, la solución facilita las órdenes de control que determinarán cómo ha de moverse el agente de una celda (o de una posición del espacio) a la siguiente.

Sin embargo, hay otros trabajos que no se centran en la dinámica del UAV, como es el trabajo publicado recientemente por Victor San Juan *et al* [6]. En este caso, la planificación de la ruta se calcula como una trayectoria a lo largo de todos los puntos discretos, correspondientes a diferentes celdas en las que se divide el mapa de búsqueda. El UAV se mueve de una celda a otra (adyacente) con una velocidad constante. El número de celdas visitadas indica la longitud de la trayectoria y, por lo tanto, el tiempo empleado.

El presente trabajo, se puede considerar una combinación de ambos enfoques:

- En primer lugar, continuamos la línea de estudio sobre la aplicación de algoritmos heurísticos al problema de la planificación de trayectorias, y en este caso analizaremos el funcionamiento de un algoritmo evolutivo diferente llamado evolución diferencial (DE, Differential Evolution) [14][15].
- En segundo lugar, las soluciones no estarán centradas en la dinámica del UAV. Las soluciones serán los puntos de paso (denominados habitualmente waypoints) que conforman la trayectoria, definidos por el desplazamiento diferencial en los ejes de coordenadas en nuestro mapa de búsqueda. Además, se permitirá que dichos desplazamientos se realicen a diferentes velocidades, cumpliendo con ciertas restricciones como el consumo y una buena relación entre el tiempo empleado y la probabilidad de detección acumulada.

1.1 Objetivos

El objetivo general de este TFM es el desarrollo de un nuevo algoritmo de búsqueda de objetivos en entornos con incertidumbre, modelada probabilísticamente, que proporcione a un UAV una trayectoria que cumpla diferentes restricciones de la misión y optimice el tiempo empleado y la probabilidad de detección acumulada.
De acuerdo con lo expuesto hasta el momento, los objetivos específicos de este trabajo son:

- Entender cómo se modelan los problemas de búsqueda de objetivos de forma probabilística mediante algoritmos heurísticos evolutivos.
- Proponer un algoritmo basado en el evolución diferencial (DE), que proporcione los puntos de paso (waypoints) que definirán la trayectoria en el área de búsqueda, y las velocidades del UAV entre cada pareja de puntos.
- Probar el comportamiento de nuestro algoritmo en diferentes escenarios de búsqueda, donde el objetivo permanezca estático (no se mueva y por lo tanto no evolucione el mapa de probabilidad por sí mismo) durante la misión.
- Extraer conclusiones y líneas futuras de trabajo relacionadas con los resultados obtenidos.

1.2 Organización de la memoria

Además de este primer capítulo introductorio, donde recogemos los objetivos de este trabajo y una breve reseña a trabajos existentes previos relacionados, esta memoria se estructura en 4 capítulos más:

En el capítulo 2 se presenta una descripción funcional completa tanto del problema que se desea resolver como de la solución propuesta. Además se lleva a cabo un repaso detallado del algoritmo Differential Evolution básico y sus operadores.

En el capítulo 3 se desarrolla la formulación matemática del algoritmo implementado, se presentan las variables de decisión, la función objetivo y restricciones, y se describen cómo se han implementado las distintas parametrizaciones del algoritmo diferencial que serán aplicadas a nuestro problema.

En el capítulo 4 se presentan y analizan los resultados obtenidos. En él se puede observar el comportamiento del algoritmo bajo diferentes escenarios, y se analiza la influencia que tienen las diferentes parametrizaciones del algoritmo de evolución diferencial sobre los mismos.

En el capítulo 5 se presentan las conclusiones y las líneas de trabajo futuras.

Finalmente se adjunta un anexo donde se muestra la evolución comparativa de varias parametrizaciones en distintas iteraciones y diferentes escenarios.

2. Problema de optimización

En este capítulo describiremos los elementos principales del problema sobre el que vamos a trabajar. En él se trata de obtener trayectorias para búsqueda de objetivos en un área establecida, optimizando la probabilidad de detección acumulada respecto al tiempo empleado.

Como hemos comentado anteriormente, este tipo de problemas surge con frecuencia cuando se trata de misiones de reconocimiento o rescate, donde es de vital importancia que se localice el objetivo en el menor tiempo posible y con una alta probabilidad, pues puede haber vidas en juego. Además a nuestro problema añadiremos otra componente, el consumo o autonomía de nuestro agente, ya que también es un factor crítico en este tipo de misiones. De hecho, la trayectoria propuesta debe garantizar que el UAV puede regresar a la base para repostar y continuar posteriormente con la misión, si es preciso.

2.1 Planteamiento del problema

En este apartado haremos una descripción completa del problema que vamos a resolver y de los elementos que lo componen (datos de entrada del problema, variables de decisión, función objetivo y restricciones)

2.1.1 Datos de entrada del problema

Como datos del problema tendremos los **escenarios o mapas de creencia (belief)**. Estos mapas son matrices donde se mezclan distintas funciones probabilísticas (gaussianas, uniformes, etc) que describen las áreas o zonas calientes donde es más probable localizar el objetivo. Los escenarios se analizarán tras un escalado de los mismos en ambos ejes, obteniendo una red de celdas que serán las que evaluaremos durante la trayectoria del UAV. En el proceso de evaluación se utilizará un sensor ideal que evaluará la celda al completo, consumiendo toda la probabilidad que tenga asignada cuando el UAV sobrevuela la celda.

Además, trabajaremos con distintos mapas de creencia con varias áreas calientes o con una única que concentre la probabilidad de detección. Para representar esos mapas

utilizaremos un código de colores donde las celdas con probabilidades más bajas o nulas tienen colores fríos y aquéllas con probabilidades más elevadas colores cálidos.

A continuación mostramos un mapa de creencia a modo de ejemplo, con su correspondiente escalado en celdas y un área de búsqueda que sigue una distribución gaussiana centrada hacia la esquina inferior derecha.

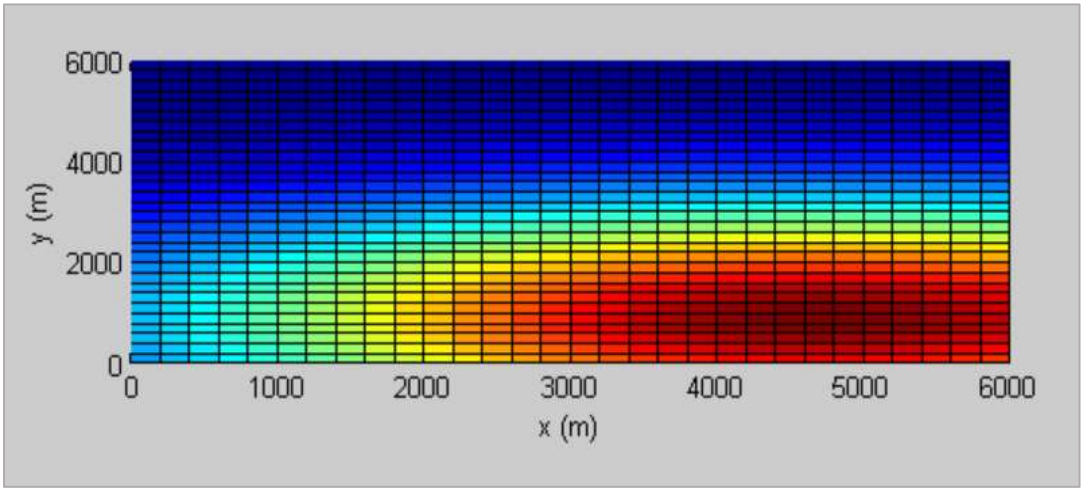


Figura 3. Ejemplo escenario de búsqueda

Los **mapas serán estáticos** durante todo el tiempo de búsqueda, es decir, nuestro objetivo no se mueve. Por otro lado, no consideraremos tampoco en dichos mapas obstáculos que dificulten el desplazamiento o identificación de nuestro objetivo.

También serán datos de nuestro problema la **velocidad mínima y máxima de crucero** de nuestro UAV, así como la función que define **la curva de consumo del UAV**. Esta función relaciona el gasto de combustible con la velocidad, siendo este aspecto muy importante para poder analizar el consumo de nuestro agente. La función que emplearemos es ficticia, aunque sí permite modelar un comportamiento análogo al de curvas reales de consumo de dispositivos similares.

En este caso se han tenido en cuenta las velocidades mínimas y máximas de crucero y no las de funcionamiento, porque supondría permitir tramos con consumos más elevados y precisamente la autonomía es un aspecto crítico en este tipo de dispositivos.

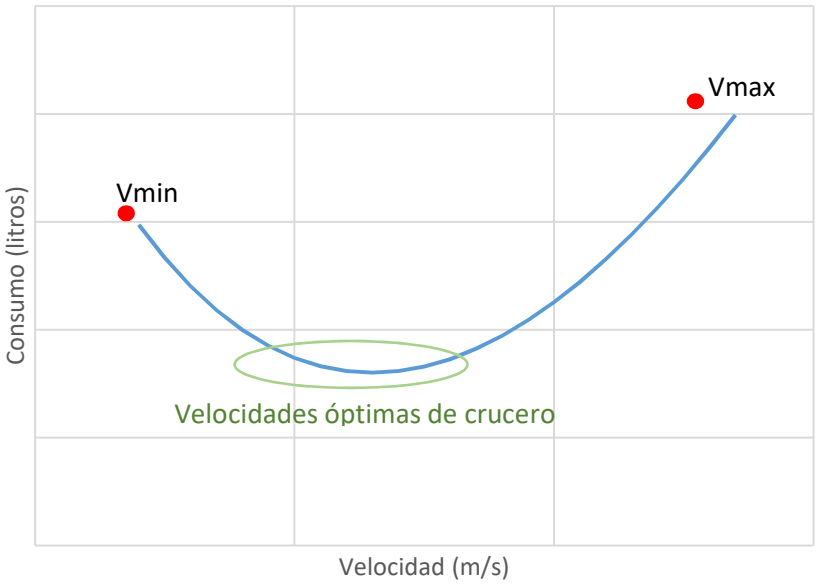


Figura 4. Ejemplo curva de consumo de un UAV

La curva de consumo, como se aprecia en la gráfica anterior, estará definida por una función en la que es fácil identificar unas velocidades óptimas de crucero donde el consumo es menor. El consumo modelado se comporta de forma similar al de un vehículo terrestre, donde se incrementa cuanto mayor es la velocidad pero también es más alto cuando circulamos con velocidades bajas y "marchas cortas".

Otro dato conocido es el **número N de puntos de paso (waypoints)** que formarán nuestra trayectoria, teniendo en cuenta que siempre el punto inicial y el final son el mismo. Suponemos que nuestro UAV parte del origen de coordenadas (0,0) y regresa al mismo, donde está su base de operaciones. El número de puntos de paso N se determinará gracias a un estudio previo del belief, analizando cuántas celdas son necesarias para que la probabilidad de detección sea una certeza (1), es decir

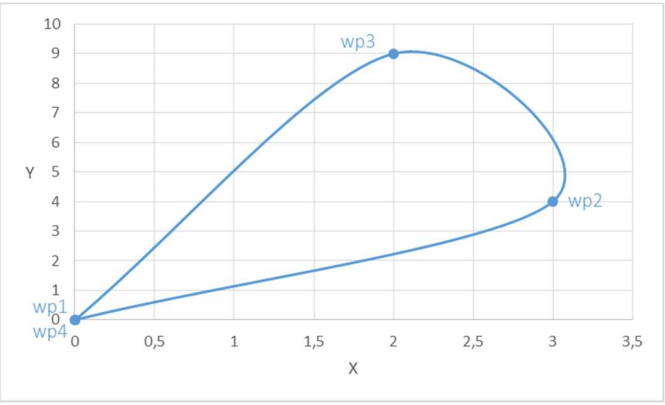
$$\sum_{i=1}^{N} P_i = 1 \qquad (1)$$

2.1.2 Variables de decisión

Las variables de decisión de nuestro algoritmo serán los wp que conforman la trayectoria de nuestro UAV. Cada wp es un vector de tres elementos, el primero define la velocidad con la que se ha desplazado el UAV para llegar a dicho wp y los otros dos elementos se definen como (Δx, Δy), es decir, el incremento en las coordenadas cartesianas desde el wp inmediatamente anterior.

Las variables de decisión integran nuestra solución o candidato: una trayectoria. Para obtenerla tendremos que componerla. Para una mayor comprensión, podemos verlo con un ejemplo.

Para representar gráficamente la trayectoria de puntos que se ha implementado se ha empleado Excel, usando líneas suavizadas. Éstas permiten representar trayectorias más parecidas a las de un agente real, donde los cambios de dirección no son tan bruscos como lo que implica la unión de puntos mediante líneas rectas:



wp1= (0, 0, 0)

wp2= (120, 3, 4)

wp3= (90, -1, 5)

wp4= (135, -2, -9)

Trayectoria (wp1, wp2, wp3, wp4)

Figura 5. Ejemplo de trayectoria compuesta por 4 waypoints

Es importante señalar que las componentes (Δx, Δy) del último wp estarán determinadas por la trayectoria realizada por el UAV, permitiendo el regreso al origen de coordenadas.

También se podrían establecer como variables de decisión para definir la posición, en lugar de las componentes incrementales (Δx, Δy) los valores absolutos de las coordenadas (x, y). De hecho, inicialmente el problema fue abordado de esta forma, pero durante el proceso de análisis descartamos esta codificación alternativa, ya que

considerar posiciones absolutas permite que las variables tomen valores en un intervalo muy amplio (toda la rejilla de búsqueda) hecho que ralentiza la convergencia de nuestro algoritmo, requiriendo tiempos de computación muy elevados. Además, con la codificación absoluta de las posiciones se obtienen candidatos demasiado alejados de lo que sería una solución realista al permitir que un punto de la trayectoria se encuentre, por ejemplo, en el origen de coordenadas y el siguiente en el extremo opuesto de la misma.

Además, al adoptar valores incrementales desde la posición anterior, no sólo reducimos el intervalo de valores que pueden adoptar las variables de decisión, sino que también estamos añadiendo coherencia a la implementación de la trayectoria, al ser el siguiente punto de paso un punto más o menos próximo al entorno de su inmediato antecesor. Esto hace que el algoritmo defina trayectorias que se aproximan más a un comportamiento real y se reduce considerablemente el tiempo de computación, aunque éste no es un objetivo de nuestro algoritmo en sí mismo.

Finalmente, para evaluar la calidad de una solución, consideraremos que, independientemente de la posición del UAV dentro de la celda, el sensor evaluará la celda correspondiente a dicha posición de forma completa, acumulándose la totalidad de la probabilidad que tiene asignada.

2.1.3 Función objetivo

Nuestro objetivo consistirá en encontrar la trayectoria que nos permita maximizar la probabilidad de detección respecto al tiempo. A diferencia de los problemas de detección en tiempo mínimo [7] [12] [21], [23], donde el objetivo es acumular la máxima probabilidad en el menor tiempo posible, en nuestro algoritmo no es relevante el orden en el que se visitan las celdas. Esto es debido a que como nuestras trayectorias tendrán un número predefinido de puntos de paso, nos importa acumular la mayor probabilidad, independientemente del orden en que sean visitadas las celdas de la rejilla.

Además la función objetivo también tendrá en cuenta el tiempo empleado en cada trayectoria, ya que éste se ve influenciado por las velocidades establecidas entre cada pareja de puntos de paso. Es decir, aunque para un escenario dado todas las trayectorias tienen el mismo número de puntos paso, unas serán más rápidas que otras, debido a la velocidad utilizada para llevarlas a cabo.

Para simultáneamente minimizar el tiempo de la trayectoria y maximizar la probabilidad acumulada a lo largo de la trayectoria minimizaremos la relación tiempo empleado/Probabilidad acumulada. Además, como el tiempo se reduce cuando se vuela entre todos los puntos de paso a la máxima velocidad permitida, es necesario tener en cuenta alguna restricción que contrarreste este comportamiento obvio y tenga en cuenta que en una aplicación real el consumo de combustible del UAV se encuentra limitado.

2.1.4 Restricciones

En nuestro planteamiento aplicaremos una única restricción que está relacionada con el consumo (autonomía) de nuestro UAV: el agente deberá ser capaz de regresar a la base desde el penúltimo punto de la trayectoria.

Para ello tendremos en cuenta un valor de autonomía máximo de nuestro agente, que no deberá ser superado por el consumo acumulado en el desplazamiento, definido por cada uno de los puntos de la trayectoria del agente hasta la base, tal y como se muestra en la siguiente expresión

$$\sum_{i=1}^{N} Ci \leq Automía\ máx\ UAV \tag{2}$$

siendo el consumo Ci, la función definida por la relación entre la velocidad de desplazamiento y el tiempo empleado entre wp_i y wp_{i+1}.

$$C_i = f(v_i, t_i) \tag{3}$$

Durante las iteraciones de nuestro algoritmo prevalecerán aquellas soluciones que cumplan con dicha restricción, o en su defecto, se tomarán aquellas que disten lo menos posible del cumplimiento de la misma.

2.2. Evolución diferencial (DE)

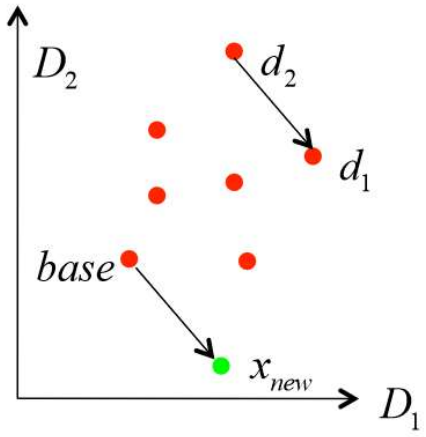
En este apartado haremos un breve repaso del algoritmo evolutivo DE y sus operadores básicos. A continuación describiremos de forma cualitativa los operadores específicos implementados en el planificador de trayectorias implementado en este TFM.

2.2.1 Descripción del funcionamiento del algoritmo básico

DE es un tipo de metaheurística basada en técnicas evolutivas, como también lo son los algoritmos genéticos (AG) y los basados en nubes de partículas (PSO). Este tipo de metaheurísticas se diferencian de otras, principalmente, en que su estrategia se basa en dirigir un conjunto de posibles soluciones del espacio de búsqueda hacia las soluciones óptimas, en lugar de transformar de forma iterativa y mediante algún tipo de regla, una única solución de partida en otra, como es el caso de las metaheurísticas basadas en métodos de búsqueda.

Otra característica especialmente atractiva de este tipo de algoritmos es que tienen en cuenta, para la iteración de cada solución, los valores del resto de soluciones, de forma que las soluciones interactúan entre sí.

En el caso particular del algoritmo DE, éste tiene en cuenta, tal y como se esquematiza en la figura 6, las diferencias entre parejas de soluciones de la población para proponer nuevas soluciones del algoritmo. Además, ha sido aplicado con éxito en diferentes tipos de problemas con variables de decisión reales.



Se le suma al individuo *base* la diferencia entre los individuos *d1* y *d2* para obtener un nuevo individuo *xnew*

Figura 6. Perturbación de individuo base

Fuente: Apuntes del Tema 3 de la asignatura Optimización Heurística y Aplicaciones

El pseudocódigo del algoritmo DE básico es el que se muestra a continuación:

```
x=DE()
1  x=Initialize()
2   repeat:
3       x_new=Pertubate(x)
4       x_new=Cross(x,x_new)
5       x=Recombine(x,x_new)
6   until Stop(x)
7   return x
```

Figura 7. Pseudocódigo algoritmo DE básico

Fuente: Apuntes del Tema 3 de la asignatura Optimización Heurística y Aplicaciones

Como podemos observar los operadores básicos del algoritmo son los siguientes:

Initialize() es la función que genera la población inicial. Normalmente se generan de forma aleatoria, siguiendo una distribución uniforme sobre los valores permitidos de las variables de decisión

Perturbate() es la función que la perturba de forma diferencial, teniendo en cuenta pares de soluciones de la misma población, las soluciones existentes en la población. Las soluciones perturbadas pueden ser elegidas aleatoriamente o teniendo en cuenta la aptitud del individuo.

Cross() se encarga de combinar las soluciones perturbadas con las existentes. Suele ser un proceso aleatorio de modo que de un mismo par de individuos se pueden obtener distintas soluciones tras el cruce.

Recombine() es la función que selecciona qué soluciones pasan a la siguiente iteración, entre las soluciones nuevas y las antiguas.

Stop() determina cuándo se finaliza la ejecución del algoritmo.

2.2.2 Operadores seleccionados

Cada uno de los operadores descritos en el apartado anterior se pueden implementar de diferentes formas. A continuación describiremos funcionalmente los operadores específicos implementados en el planificador desarrollado en este TFM.

Initialize() Generador de población inicial aleatorio

Generaremos nuestra población inicial de forma aleatoria. La población estará integrada por un conjunto de trayectorias, cada una de ellas con un número predefinido de puntos de paso.

Cada punto de paso se codifica mediante un vector de tres elementos:

$$wp_i = (v_i, \Delta x_i, \Delta y_i) \tag{4}$$

La generación de la velocidad de navegación es aleatoria, siguiendo una distribución uniforme, dentro del intervalo de velocidades de crucero del UAV

$$v_i \in (v_{minUAV}, v_{maxUAV}) \tag{5}$$

La generación de los valores diferenciales de las posiciones cartesianas, son aleatorios, siguiendo una distribución uniforme y garantizando que los puntos resultantes estén dentro del área de búsqueda. Es decir, se implementarán los mecanismos necesarios para asegurar que ningún punto de paso se salga del mapa definido por razones que explicaremos con más detalle en capítulos posteriores.

El rango de valores que pueden tomar los elementos diferenciales que definirán las posiciones cartesianas del wp, será un intervalo definido por una constante de proporcionalidad aplicada al factor de escala empleado en la segmentación de nuestro mapa de búsqueda. De hecho, será objeto de análisis del capítulo de resultados, el comportamiento del algoritmo empleando valores diferentes en dicha constante.

$$(\Delta x_i, \Delta y_i) \in (k * scale, k * scale) \tag{6}$$

Siendo *k*, constante de proporcionalidad

scale, número en el que se dividen los ejes cartesianos de cada celda del área de búsqueda para conformar el mapa de rejilla.

Perturbate() Operador de perturbación

El operador de perturbación se implementará en tres pasos:

- En primer lugar, se seleccionará un individuo como base (*base*). Será objeto de estudio de este trabajo el comportamiento del algoritmo empleando diferentes métodos para la selección de dicha base.
- En segundo lugar, se definirá la función diferencial de otros dos individuos diferentes a la base (*Perturbación*). También será objeto de estudio el análisis del comportamiento del algoritmo, en función de los criterios empleados para la definición de dicha función.
- Por último, el operador perturbación será el resultado de sumar a la base seleccionada la función diferencial de los otros dos individuos, obteniendo así un individuo perturbado (*Perturbado=base + Perturbación*).

En el capítulo siguiente se explicará con más detalle la métrica de nuestro algoritmo, y las parametrizaciones implementadas en nuestro operador de perturbación.

Cross() Operador de combinación

Son múltiples las opciones que se pueden implementar para combinar soluciones perturbadas con soluciones existentes. En nuestro caso emplearemos un cruce simple por un punto.

En el cruce por un punto se determina aleatoriamente, empleando una distribución uniforme, el índice que identificará el punto de cruce de los dos individuos. Así la solución resultante será la formada por las secuencias de ambos progenitores, tal y como se muestra en la figura siguiente.

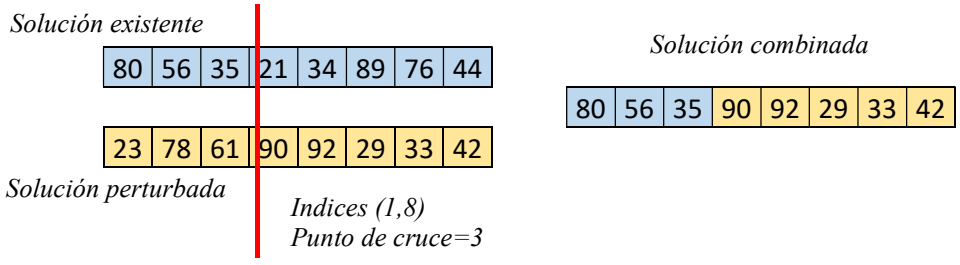


Figura 8. Representación de cruce por un punto

Aunque también se pueden emplear otras formas de combinación, como el cruce por N puntos, uniforme, aritmético, en este TFG solo se comprobará el funcionamiento del algoritmo mediante el cruce simple por un punto.

Recombine() Operador de selección

Para la selección de soluciones que pasan a la siguiente iteración, nuestro algoritmo comparará la población de soluciones existentes con la población de soluciones nuevas dos a dos.

$$\begin{bmatrix} Solución\ existente\ 1 \\ Solución\ existente\ 2 \\ \dots\dots \\ Solución\ existente\ n \end{bmatrix} \qquad \begin{bmatrix} Solución\ nueva\ 1 \\ Solución\ nueva\ 2 \\ \dots\dots \\ Solución\ nueva\ n \end{bmatrix}$$

Figura 9. Selección entre poblaciones

La selección de la solución superviviente de cada pareja se basará principalmente en dos condiciones: por una parte que cumplan la restricción de autonomía y por otra que mejore la aptitud del individuo

La aplicación de las condiciones seguirá los siguientes criterios:

- En caso de cumplir ambos con la restricción de autonomía, se quedará con aquél con mejor aptitud
- En caso de que uno cumpla la restricción y el otro no, se quedará con aquél que cumpla la restricción
- En el caso de que ninguno cumpla con la restricción, se quedará con aquél que diste menos de alcanzar el cumplimiento de la misma, independientemente de la aptitud del individuo.

Stop() Operador que detiene el algoritmo

Nuestro algoritmo se ejecutará indefinidamente mientras no se cumpla alguna de las siguientes condiciones:

- Que el número de iteraciones haya alcanzado un número máximo previamente definido
- Que la probabilidad acumulada de la solución haya alcanzado una probabilidad objetivo previamente definida

3. Construcción de la solución

A continuación procedemos a describir el algoritmo implementado teniendo en cuenta el problema descrito y las consideraciones que hemos aplicado a la solución que se han comentado en el capítulo 2.

Comenzaremos indicando qué parámetros son datos conocidos en nuestro problema.

3.1 Datos del problema

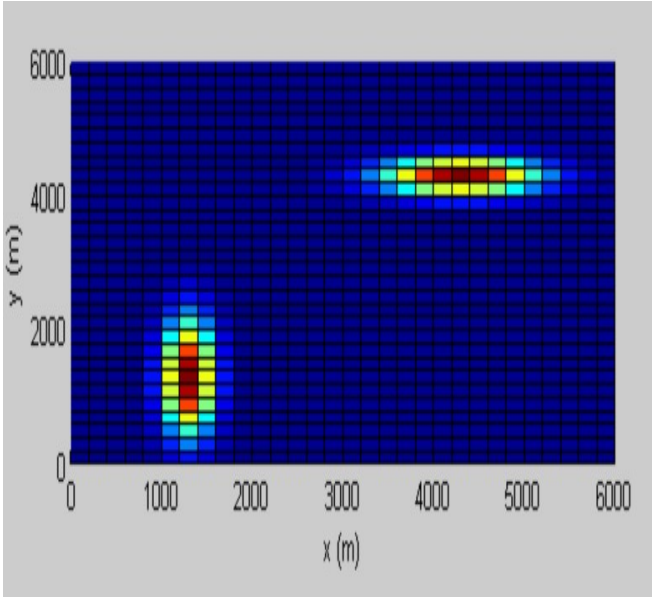
Para poder construir una solución a un problema, debemos conocer los datos disponibles. En los siguientes apartados se describen con detalle los datos de los que partimos para implementar la solución.

3.1.1 Escenarios

Para analizar el comportamiento de nuestro algoritmo debemos probarlo sobre escenarios de los que tendremos la siguiente información:

- *n_cols*, número de columnas de nuestro mapa de probabilidad.
- *n_rows*, número de filas de nuestro mapa de probabilidad.
- *scale*, factor de escala aplicado al mapa de probabilidades, tanto al número de filas como al de columnas, y que define la rejilla sobre la que haremos la búsqueda.
- *belief*, función que define el mapa de probabilidad. Esta función tiene como variables de entrada el número de filas, el número de columnas y el factor de escala de nuestro mapa. La función devuelve la probabilidad (creencia) de que el objetivo se encuentre en cada celda del mapa, siguiendo una distribución predefinida (gaussiana, uniforme, etc.).

En este trabajo tomaremos como escenario de referencia el que mostramos a continuación, con dos zonas “calientes”, donde la probabilidad de localizar al objetivo es más elevada, definidas por sendas distribuciones gaussianas:



n_rows=31

n_cols=31

scale=200

Figura 10. Escenario básico de referencia

En el mapa, tal y como se observa, los valores que pueden tomar x e y se encuentran en los intervalos:

$$x_i \in [0, (n_{rows} - 1) * scale] \quad (7)$$

$$y_i \in [0, (n_{cols} - 1) * scale] \quad (8)$$

Por tanto, a la función que define nuestro mapa de creencia le daremos como parámetros de entrada $(n_{rows}, n_{cols}, y\ scale)$ para su posicionamiento en los ejes cartesianos.

Será el valor de la coordenada Z, definida por la función de densidad de probabilidad, la que nos dé el valor de la probabilidad de detección en un punto cualesquiera del mapa.

En el capítulo siguiente analizaremos cómo se comportan distintas parametrizaciones de nuestro algoritmo sobre dicho escenario. Aquélla con la que obtengamos los mejores resultados será la que aplicaremos sobre otros escenarios, para determinar la bondad del algoritmo sobre:

- Escenarios con una o varias áreas de probabilidad.
- Escenarios con áreas de probabilidad con distintas distribuciones (gaussiana, uniforme, etc.).

De este modo podremos comprobar si existe una dependencia directa entre la parametrización óptima y el escenario bajo estudio.

3.1.2. Parámetros del UAV

Con el fin de que el planteamiento se aproxime a un caso real, hemos tenido en cuenta características técnicas de drones existentes en el mercado para establecer ciertos parámetros de aplicación de nuestro problema:

- **La velocidad del UAV** en cada uno de los tramos de las trayectorias, se generará de forma aleatoria, pero siempre dentro del intervalo de velocidades de crucero del UAV:

  $$v_i \in (v_{minUAV} = 90km/h, v_{maxUAV} = 150km/h) \quad (9)$$

  Por tanto, el intervalo que delimita las velocidades de tramo es conocido y constante durante todas las iteraciones.
  La velocidad óptima de crucero vendrá definida por la curva de consumo del UAV y que se explica a continuación

- **La curva de consumo del UAV** es la función que relaciona la velocidad de nuestro dispositivo, el tiempo de vuelo y el consumo de combustible. Se trata de una función diseñada específicamente para este problema, pero que sigue un comportamiento análogo al de dispositivos comerciales.
  Los dispositivos comerciales tienen curvas de consumo como la que se muestra a continuación

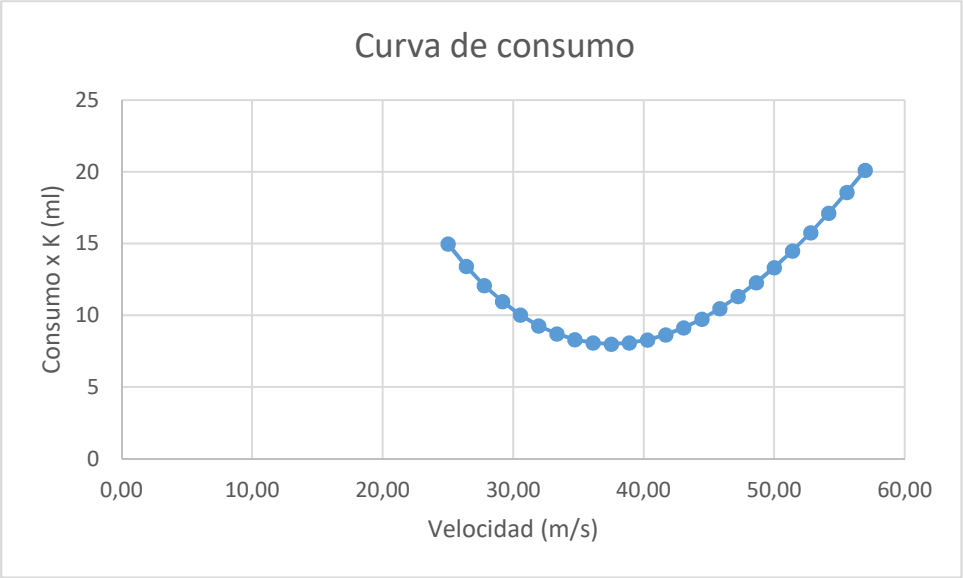


Figura 11. Curva de consumo del UAV utilizada

Dicho comportamiento implica un crecimiento exponencial del consumo con el incremento de la velocidad, pero también un incremento (aunque menos abrupto) cuando la velocidad de navegación es especialmente baja.

Hemos modelado esta función del siguiente modo:

$$C_i = (8 + \frac{((v_i - 37{,}50) \cdot 3{,}6)^2}{(v_i + 55{,}55) \cdot 3{,}6}) \cdot tiempo_i \qquad (10)$$

Donde

v_i	Velocidad del UAV en el tramo i (Unidades *metros/segundo*)
$tiempo_i$	Tiempo que tarda en recorrer el UAV el tramo i (Unidades *segundos*)
37,50	Velocidad de crucero óptima en m/s (135Km/h)
55,55	Velocidad máxima de operación en m/s (200Km/h)

El consumo así obtenido se obtiene en un orden de magnitud de décimas de mililitro, porque en este tipo de dispositivos el peso es un factor crítico.

Para este trabajo consideraremos que el depósito puede albergar como máximo 25 litros, y por tanto aplicaremos el escalado correspondiente para considerar este límite en las restricciones

3.2 Variables de decisión

Las variables de decisión de nuestro algoritmo, tal y como comentamos en capítulos anteriores, serán los valores diferenciales que podrán adoptar las coordenadas de los puntos que formarán las trayectorias o soluciones a evaluar y las velocidades de vuelo entre cada tramos.

Por tanto nuestras trayectorias se implementarán, como matrices donde cada fila es un waypoint:

$$trayectoria = \begin{bmatrix} wp1 \\ \dots \\ wpn \end{bmatrix} \tag{11}$$

O dicho de otro modo, como una matriz de 3 columnas, donde la primera es un vector de velocidades, la segunda un vector de diferenciales de x y la tercera un vector de diferenciales de y.

Además, los valores de las variables incrementales se encuentran comprimidos en los siguientes rangos:

$$\Delta x_i \in [0, scale * K] \tag{12}$$

$$\Delta y_i \in [0, scale * K] \tag{13}$$

siendo K un factor aplicado al escalado de nuestro mapa y que determinará el rango de valores que pueden tomar nuestros diferenciales de x e y. Asi por ejemplo, en el caso en el que *scale*=200 y *K*=3, se tendrá que $\Delta x_i \in [0{,}600]$ y que $\Delta y_i \in [0{,}600]$.

3.3 Operadores del algoritmo

A continuación, considerando la descripción de lo que son los operadores del DE descritos en el capítulo 2, se describen los implementados para nuestro algoritmo.

3.3.1 Operador generación.

Esta función nos permite generar una población inicial (*Ini*) integrada por un número predefinido de individuos (*n_poblacionInicial*).

Los parámetros que se pasan a la función son los valores que determinan los intervalos de los diferenciales, las velocidades mínima, máxima y el número de waypoints.

Cada individuo `Ini{i}`es una trayectoria definida por los vectores (4): velocidad, diferencial de x, diferencial de y. Los diferenciales de x e y tomarán valores dentro de los intervalos definidos en (12) y (13).

Consideramos que la velocidad de un tramo, definida por los puntos (wp1,wp2) viene definida por el valor que tiene el vector velocidad en wp2. Por tanto, la velocidad del primer waypoint de cada individuo por defecto será 0, y la velocidad del último tramo, será la velocidad óptima de crucero (135 Km/h), pues es el recorrido de vuelta a la base de operaciones del UAV.

Por otro lado, tal y como comentaremos en el apartado de resultados, nuestra generación tendrá la peculiaridad de no ser totalmente aleatoria, se comprobará que el valor que toma el diferencial está dentro del mapa de búsqueda.

A continuación se muestra el control sobre el valor que puede tomar el valor diferencial durante la generación de la población inicial

Si $\sum_{i=1}^{n} \Delta x_i < \Delta x_{n+1}$ se calcula

$$\Delta x_{n+1} \in [-\sum_{i=1}^{n} \Delta x_i \, , \Delta x_{n+1}] \qquad (14)$$

En caso contrario,

si $\sum_{i=1}^{n} \Delta x_i > (n_{rows} - 1) * scale - \Delta x_{n+1}$, se calcula

$$\Delta x_{n+1} \in [-\Delta x_{n+1}, (n_{rows} - 1) * scale - \sum_{i=1}^{n} \Delta x_i] \quad (15)$$

En caso contrario se calcula

$$\Delta x_{n+1} \in [-\Delta x_{n+1}, \Delta x_{n+1}] \quad (16)$$

Observamos que si no se hace este control, se ralentiza considerablemente la convergencia del algoritmo e incluso es más probable que se quede atrapado en óptimos locales.

3.3.2 Operador perturbación. Funciones *eligeBase.m* y *perturbar.m*

Como se ha explicado anteriormente, el operador perturbación se ha implementado en tres etapas:

- Se seleccionará un individuo como base.
- Se seleccionan otros dos individuos diferentes a la base para definir la función perturbación.
- Se obtendrá el individuo perturbado como suma de la base y la perturbación.

Selección de base. Función *eligeBase.m*

Esta función nos permitirá obtener el individuo base. Se han implementado tres formas diferentes de obtenerlo:

- *Opción 1: Selección aleatoria.* Se elige un individuo al azar.
- *Opción 2: Selección del mejor candidato según sus aptitudes.* Se evalúan todos los individuos de la población y se elige aquél que tenga mejores aptitudes.

Será objeto de este estudio analizar el comportamiento del algoritmo según se lleve a cabo un tipo de selección u otro. Por ello, *opcionbase* será un parámetro de entrada de la función *eligeBase.m*,. Según el valor que tome *opcionbase* estaremos implementando uno u otro método de elección del individuo.

La función evaluar se explicará más adelante en este mismo apartado.

Selección de dos individuos para función diferencial

De acuerdo con el algoritmo básico DE, una vez elegida la base, seleccionamos otros dos individuos diferentes a dicha base, pero de la misma población.

En nuestro caso, emplearemos la función *randperm*, para obtener esos dos individuos, *IndividuoA* e *IndividuoB.*

Para asegurarnos que no seleccionamos la base, desplazaremos la base a la posición final de la población y haremos el randperm sobre los n-1 individuos iniciales.

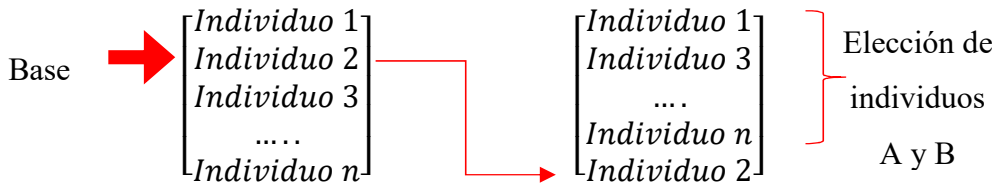


Figura 12. Elección de individuos A y B diferentes a la Base

Obtención de individuo perturbado. Función *perturbar.m*

Del mismo modo que para la selección de la base, *perturbar.m* tendrá un parámetro (*opcionpert)*, que podrá tomar tres valores diferentes. Dicho valor determinará cuál ha sido la implementación llevada a cabo de la función diferencial que se sumará a la base para obtener el individuo perturbado:

- *Opción 1: Factor F igual para todas las soluciones*
- *Opción 2: Se emplea el mismo factor F para cada nuevo elemento de la población en todas sus variables de decisión*
- *Opción 3: Se emplea el distinto factor F para cada nuevo elemento de la población y cada variable de decisión que lo compone*

Siendo F el factor que escala la discrepancia de *IndividuoA* e *IndividuoB.*

La función nos devolverá el candidato perturbado, fruto de sumar la base y la perturbación:

$$\text{NuevoCandidato}=\text{Base}+\text{Perturbacion} \quad (17)$$

3.3.3 Operador de combinación

Con el cruce simple por un punto obtendremos nuevos candidatos que combinan soluciones existentes de la población inicial, *Ini*, con soluciones perturbadas. Estos nuevos candidatos serán los individuos que formarán la población *NuevoIni.*

El punto de cruce se determina aleatoriamente, empleando una distribución uniforme. No debemos olvidar en este punto que debemos reconstruir el nuevo candidato en el último waypoint, para asegurarnos que el valor diferencial de x e y lo devuelve a la posición de la base del UAV.

3.3.4 Operador de selección. Funciones *comparar.m*, *calculaDTPC.m*, *idealSensor.m y evaluar.m*

Tal y como se ha explicado anteriormente, en el apartado de los operadores del algoritmo, se llevará a cabo la selección, que consistirá en comparar uno a uno los individuos de la población inicial (*Ini*) con los de la nueva población (*nuevoIni*)

$$\begin{bmatrix} Ini(1) \\ Ini(2) \\ \dots\dots \\ Ini(n) \end{bmatrix} \longleftrightarrow \begin{bmatrix} nuevoIni(1) \\ nuevoIni(2) \\ \dots\dots \\ nuevoIni(n) \end{bmatrix}$$

Figura 13. Selección entre poblaciones Ini y nuevoIni

Función *comparar.m*

Los parámetros que recibe la función son los dos candidatos que se encuentran en la misma posición (uno de cada población), el número de waypoints, el mapa de creencia, el factor de escala del mapa y la autonomía del UAV.

Compara los individuos de forma secuencial, devolviendo el individuo elegido e indicando si pertenece a la población inicial o a la nueva población.

En primer lugar se calculan ciertos parámetros de las trayectorias (distancia recorrida, tiempo, probabilidad y consumo) de ambos candidatos al llamar a la función *calculaDTPC.m.* Esta función la describiremos más adelante.

Posteriormente, se comparan los consumos de ambos individuos para verificar si cumplen con la restricción de no agotar el combustible del UAV. Si es así, entonces se comparan según la función objetivo y el algoritmo elegirá al mejor candidato para que pase a la nueva población.

Si, consumoPadre & consumoHijo < autonomiaUAV

Elegido=minimo (T/P(Padre), T/P(Hijo))

donde,

T/P es la relación entre el tiempo de la trayectoria y la probabilidad acumulada

En el caso de que sólo uno cumpla dicha restricción, será ése el que pase a formar parte de la nueva población.

Finalmente si ninguno cumple con la restricción, entonces el algoritmo elegirá a aquel candidato que se aleje menos de cumplirla.

Función *calculaDTPC.m*

Esta función nos permite obtener varios parámetros de una trayectoria:

D- Distancia total recorrida por el UAV en esa trayectoria

T-Tiempo total empleado por el UAV en realizar la trayectoria

P-Probabilidad de detección del objetivo

C- Consumo total del UAV al realizar la trayectoria

La función calcula la distancia, tiempo, probabilidad y consumo de cada uno de los tramos de la trayectoria, como se muestra a continuación:

distancia(i)=sqrt((Δx)^2+(Δy)^2); (18)

tiempo(i)=distancia(i)/velocidad(i); (19)

probabilidad(i)= idealSensor(belief); (20)

consumo(i)=(8+(((veloc(i)-37.5)*3.6)^2/((veloc(i)+55.55)*3.6)))*tiempo(i) (21)

Para el cálculo de la probabilidad emplearemos la función *idealSensor.m*, que es la que permitirá leer el valor de probabilidad de la celda del mapa de rejilla definida por el punto de la trayectoria en nuestro mapa de creencia.

Una vez leída la probabilidad, dicha función pone a cero la probabilidad de esa celda del mapa para evitar que puntos coincidentes en la misma celda no acumulen probabilidad al ser leídos nuevamente.

Los valores de distancia, tiempo, probabilidad y consumo totales, serán el sumatorio de los obtenidos en cada uno de los tramos de la trayectoria:

$Distancia\ total$	$\sum_{i=1}^{N} dist(i)$
$Tiempo\ total$	$\sum_{i=1}^{N} tiemp(i)$
$Probabilidad\ total$	$\sum_{i=1}^{N} prob(i)$
$Consumo\ total$	$\sum_{i=1}^{N} cons(i)$

Figura 14. Cálculo de parámetros totales de la trayectoria

Función *idealSensor.m*

Esta función recibe los valores de las coordenadas cartesianas (x,y) de nuestro mapa y devuelve el valor de la probabilidad de la celda asociada a dicha posición. Se considera que nuestro sensor es ideal y es capaz de captar toda la probabilidad de la celda sobre la que hace la lectura.

Además modifica el valor de la probabilidad en esa posición para que no se acumule de nuevo si hay otra posición que defina la misma celda del mapa.

Función *evaluar.m*

Esta función devuelve el mejor candidato de una población, para ello realiza los siguientes pasos:

- Calcula los parámetros de cada uno de los individuos (distancia, tiempo, probabilidad y consumo), empleando para ello la función *calculaDTPC.m,*
- Identifica todos los individuos que cumplen con la restricción de consumo.
- De entre todos los candidatos identificados en el paso anterior, escoge aquél con la menor relación Tiempo/Probabilidad.
- En caso contrario, escoge al candidato que diste menos de cumplir la restricción de autonomía del UAV.

4. Evaluación de resultados

Tal y como se ha explicado en capítulos anteriores, a continuación se comienza probando el algoritmo con diferentes parametrizaciones sobre un mismo escenario para intentar determinar cuál es la mejor versión del algoritmo desarrollado para dicho escenario, y así estudiar el comportamiento del mismo. A continuación se probará la mejor parametrización sobre una batería de escenarios con el objeto de poder determinar si la parametrización elegida es apropiada para resolver el problema de forma genérica.

El escenario básico de referencia, sobre el que vamos a hacer nuestro primer análisis de resultados, es el representado anteriormente en la figura 10, integrado por dos áreas de probabilidad definidas por sendas gaussianas.

Los parámetros variables que hemos considerado en nuestro algoritmo para el análisis preliminar son los siguientes:

wp	Número de waypoints
Iteraciones	Número máximo de iteraciones
Probab	Probabilidad máxima objetivo
Pob_Ini	Número de individuos de la población inicial
Autonomia	Autonomía del UAV
OPert	Tipo de perturbación
OBase	Tipo de selección de base
Diferencial	Factor de escala aplicado al diferencial
F const	Valor de F aplicado a todas las soluciones
F	Valor de F aplicado a cada solución, aplicado a cada parámetro de cada solución

Tabla 1. Parámetros del algoritmo

4.1 Variaciones con waypoints, número de individuos, tipo de perturbación y tipo de selección

El análisis preliminar consta de una batería de 25 pruebas donde se implementan diferentes combinaciones modificando sólo cuatro parámetros: número de waypoints, número de individuos de la población inicial, tipo de perturbación y tipo de selección de base. Las combinaciones son las que se muestran en la siguiente tabla (donde se resaltan en gris las columnas asociadas a los parámetros analizados en esta sección):

Prueba	N	Iterac	Probab	POB_Ini	Autonomia	OPert	OBase	Diferencial	Fconst	F
1	80	600	0,9	10	50.000,00	1	1	5	rand	rand
2	120	600	0,9	20	50.000,00	1	1	5	rand	rand
3	120	600	0,9	40	50.000,00	1	1	5	rand	rand
4	200	600	0,9	40	50.000,00	1	1	5	rand	rand
5	80	600	0,9	10	50.000,00	1	2	5	rand	rand
6	120	600	0,9	20	50.000,00	1	2	5	rand	rand
7	120	600	0,9	40	50.000,00	1	2	5	rand	rand
8	200	600	0,9	40	50.000,00	1	2	5	rand	rand
9	80	600	0,9	10	50.000,00	2	1	5	rand	rand
10	120	600	0,9	20	50.000,00	2	1	5	rand	rand
11	120	600	0,9	40	50.000,00	2	1	5	rand	rand
12	200	600	0,9	40	50.000,00	2	1	5	rand	rand
13	80	600	0,9	10	50.000,00	2	2	5	rand	rand
14	120	600	0,9	20	50.000,00	2	2	5	rand	rand
15	120	600	0,9	40	50.000,00	2	2	5	rand	rand
16	200	600	0,9	40	50.000,00	2	2	5	rand	rand
17	80	600	0,9	10	50.000,00	3	1	5	rand	rand
18	120	600	0,9	20	50.000,00	3	1	5	rand	rand
19	120	600	0,9	40	50.000,00	3	1	5	rand	rand
20	200	600	0,9	40	50.000,00	3	1	5	rand	rand
21	80	600	0,9	10	50.000,00	3	2	5	rand	rand
22	120	600	0,9	20	50.000,00	3	2	5	rand	rand
23	120	600	0,9	40	50.000,00	3	2	5	rand	rand
24	200	600	0,9	40	50.000,00	3	2	5	rand	rand
25	200	600	0,9	40	50.000,00	2	1	5	rand	rand

Tabla 2. Parametrización preliminar. Batería de pruebas inicial

Para poder analizar los resultados obtenidos en cada una de las pruebas y cada una de las ejecuciones, se tienen en cuenta los siguientes parámetros:

Ejecución	La ejecución
Pini	Probabilidad acumulada de la mejor solución inicial (primera iteración)
Pfin	Probabilidad acumulada en la mejor solución final (última iteración)
%mejora	La mejora porcentual que supone la probabilidad final frente a la inicial
Iteración	La iteración en la que se produce el último cambio de individuos en la población
Consumo	Consumo de la solución final obtenida
Distancia	Distancia recorrida con la solución obtenida
Nº cambios	Número de cambios (de individuos) realizados durante la ejecución

Tabla 3. Descripción de resultados

Es importante aclarar que el parámetro Nº cambios nos facilita una información muy interesante. Cuando se comparan los individuos uno a uno de las poblaciones Ini y nuevoIni, se elige el individuo con mayor aptitud. Si este individuo pertenece a nuevoIni se registra el cambio, incrementando en 1 el parámetro y almacenando también en qué iteración se ha producido dicho cambio. Por tanto, si este parámetro es muy alto y la iteración en que se registra el último cambio es muy alto, podemos deducir que el algoritmo no ha terminado de converger.

Como el algoritmo de evolución diferencial tiene un comportamiento estocástico y puede devolver en cada ejecución una solución diferente, es necesario ejecutarlo múltiples veces sobre cada uno de las 25 parametrizaciones recogidas en la tabla 2. Los resultados obtenidos para cada una de las variables analizadas se promedian, de forma que se puedan comparar simultáneamente los valores de las múltiples ejecuciones de cada una de las parametrizaciones.

A continuación se muestra, como ejemplo, los resultados obtenidos en 5 de las 10 ejecuciones de la prueba 12. En la última fila de la tabla de resultados se calcula el promedio de cada uno de los valores obtenidos en las distintas ejecuciones de la misma prueba.

Ejecución	Pini	Pfin	%mejora	Iteración	Consumo	Distancia	Nº cambios
1	0,1781	0,7130	300%	599	38.722,01	127.321,84	289
2	0,3555	0,5682	60%	598	31.886,05	103.521,05	301
3	0,1744	0,7796	347%	599	32.667,99	111.310,26	266
4	0,1261	0,7560	500%	600	32.926,22	112.168,68	315
5	0,2261	0,8168	261%	600	31.869,00	102.259,79	298
Promedio	**0,2120**	**0,7267**	**294%**	**599**	**33.614,25**	**111.316,32**	**294**

Tabla 4. Resultados prueba 12. Parametrizaciones preliminares

Los valores recogidos en la tabla 4 nos permiten extraer algunas conclusiones de esta parametrización:

- El porcentaje medio de mejora en la probabilidad acumulada a lo largo de las iteraciones ronda el 300%.
- Se ha llevado a cabo un promedio de cambios que ronda los 300 y además el último se produce en la iteración 599 (en una parametrización con 600 generaciones). Lo que permite deducir que **el algoritmo no ha terminado de converger**

Tal y como se ha comentado antes, los valores promedio serán los que se utilizarán para comparar los resultados obtenidos en cada una de las pruebas realizadas y nos permitirá determinar la bondad de cada una de las parametrizaciones.

Por otro lado, el algoritmo también almacena la representación gráfica de la trayectoria solución para poder hacer una valoración "visual" de los resultados obtenidos.

A continuación se muestran las gráficas del resultado obtenido en la ejecución 5, que es en la que se ha obtenido la mejor probabilidad de detección, con el menor consumo y la menor distancia recorrida.

Primero representaremos la dispersión de los punto de paso de la mejor trayectoria. En esta grafica podemos observar que estos puntos se encuentran principalmente situados sobre las zonas en las que se encuentran las dos gaussianas iniciales.

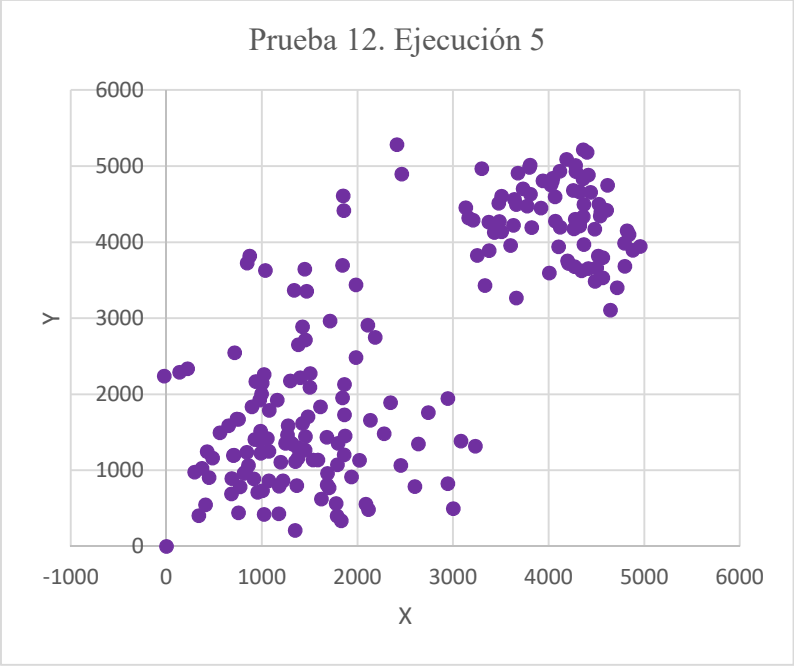


Figura 15. Solución Parametrización 12 - Ejecución 5. Puntos de la trayectoria

A continuación se representa la trayectoria generada sobre el escenario básico de referencia. En ella podemos observar como el UAV se centra en recoger primero la probabilidad de una de las dos gaussianas para desplazarse a continuación a recoger la

probabilidad de la otra. Además, la solución claramente no resulta óptima, ya que la trayectoria que existe entre ambas zonas de probabilidad visita múltiples puntos de probabilidad nula. Teniendo en cuenta que acabamos de observar que el algoritmo no parece haber convergido todavía en esta parametrización, observando la solución obtenida parece clara, diferentes zonas de mejora de la solución.

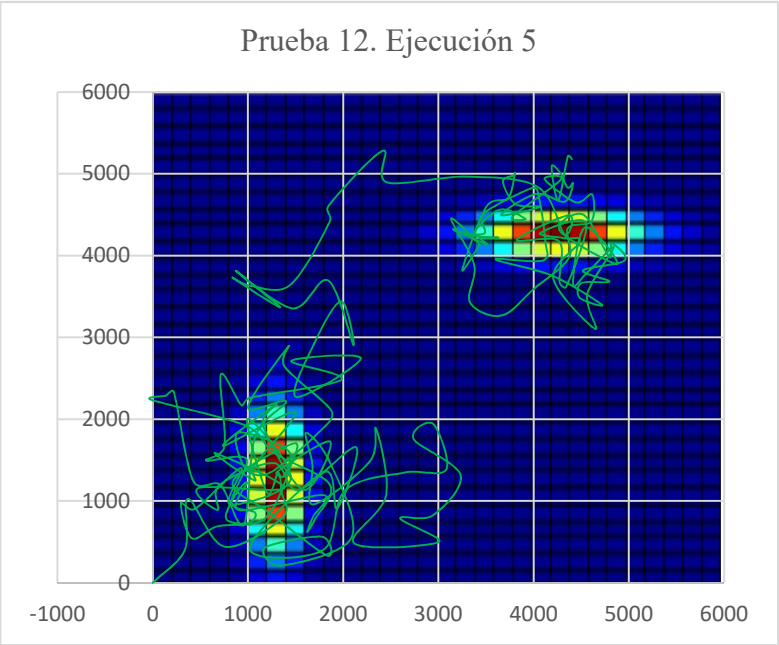


Figura 16. Solución Parametrización 12 - Ejecución 5. Trayectoria sobre escenario básico

Ahora que ya se ha explicado cómo se va a proceder con la comparativa de los resultados, se analizarán los resultados promedio obtenidos sobre las 25 parametrizaciones preliminares descritas anteriormente. Para cada una de las parametrizaciones se han realizado 10 ejecuciones del algoritmo.

Los resultados obtenidos se muestran a continuación:

P	wp	POB_Ini	OPert	OBase	Pini	Pfin	%mejora	Iteración	Consumo	Nº cambios
1	80	10	1	1	0,19	0,36	108,97%	535	15.407,38	247
2	120	20	1	1	0,31	0,46	50,37%	53	21.551,38	25
3	120	40	1	1	0,31	0,40	27,72%	30	22.703,99	17
4	200	40	1	1	0,30	0,60	133,62%	585	26.149,88	378
5	80	10	1	2	0,21	0,26	26,57%	16	16.716,85	13
6	120	20	1	2	0,27	0,34	26,38%	22	23.866,70	20
7	120	40	1	2	0,30	0,41	36,08%	27	24.034,54	25
8	200	40	1	2	0,41	0,58	41,60%	104	43.181,82	63
9	80	10	2	1	0,24	0,37	61,30%	275	16.383,67	155
10	120	20	2	1	0,27	0,47	74,08%	600	21.097,25	349
11	120	40	2	1	0,29	0,57	98,19%	599	18.479,77	291
12	200	40	2	1	0,21	0,73	293,67%	599	33.614,25	294
13	80	10	2	2	0,21	0,25	16,09%	17	17.101,54	15
14	120	20	2	2	0,29	0,37	28,03%	34	23.774,71	33
15	120	40	2	2	0,33	0,40	21,25%	36	25.954,79	36
16	200	40	2	2	0,44	0,55	25,13%	31	37.448,72	30
17	80	10	3	1	0,21	0,34	70,10%	487	16.064,95	242
18	120	20	3	1	0,24	0,57	145,17%	599	25.773,99	202
19	120	40	3	1	0,30	0,61	107,44%	596	23.762,29	135
20	200	40	3	1	0,38	0,76	100,51%	593	37.292,98	112
21	80	10	3	2	0,21	0,26	23,09%	31	18.883,10	26
22	120	20	3	2	0,26	0,38	48,12%	55	26.397,93	51
23	120	40	3	2	0,32	0,48	51,82%	123	26.031,15	112
24	200	40	3	2	0,37	0,52	43,44%	124	42.361,04	111
25	200	40	2	1	0,33	0,68	122,82%	599	29.761,73	301

Tabla 5. Resultados de parametrizaciones preliminares 1-25

Se observa que las pruebas sombreadas en color verde son las que proporcionan mejores resultados. De dichos resultados se pueden extraer las siguientes conclusiones:

- Se obtienen los mejores resultados cuando el número de waypoints es 200, ya que es el número mínimo de celdas que deben ser visitadas para alcanzar una probabilidad acumulada de 1. Si se emplean menos waypoints es imposible que se puedan alcanzar probabilidades objetivo razonables.
- Parece que el algoritmo funciona mejor cuando la selección de la base es aleatoria, independientemente de la opción de perturbación empleada. Es posible que en las

parametrizaciones en las que se elige la base en función de la aptitud de los candidatos el algoritmo se quede "atrapado" en óptimos locales.

- En cuanto a la función de perturbación, las que proporcionan mejores soluciones son aquéllas en las que se perturba cada solución o incluso los diferentes parámetros de la solución (OPert =2, OPert=3). Ahora bien en el caso de la perturbación tipo 3 (prueba 20), parece que la convergencia es más rápida y el número de cambios necesarios es menor.

A continuación, analizaremos por ejemplo, las 3 mejores ejecuciones (de las 10 realizadas) de la parametrización 20:

Ejecución	Pini	Pfin	%mejora	iteracion	consumo	distancia	nº cambios
1	0,4029	0,7631	89%	580	36.750,37	120.399,16	123
2	0,3866	0,7782	101%	598	38.469,31	121.715,56	106
4	0,4236	0,8426	99%	600	36.690,55	120.908,32	120

Tabla 6. Resultados prueba 20. Parametrizaciones preliminares

Los porcentajes de mejora son menores que los de la prueba 12, pero esto no implica que la solución sea peor. Teniendo en cuenta que la probabilidad inicial es la que obtenemos del mejor candidato tras la primera iteración, lo que realmente quiere decir es que, con este tipo de perturbación, obtenemos probabilidades más altas ya desde las primeras iteraciones.

Si a esto le sumamos que hay menos cambios, y que el promedio del número de iteración en el que se obtiene el último cambio disminuye, podemos pensar que esta parametrización converge más rápido que la que emplea la perturbación tipo 2. Sin embargo presenta un consumo y distancia más alta. Veamos qué sucede con el análisis visual.

A continuación mostramos la trayectoria de la ejecución 4 de la parametrización 20:

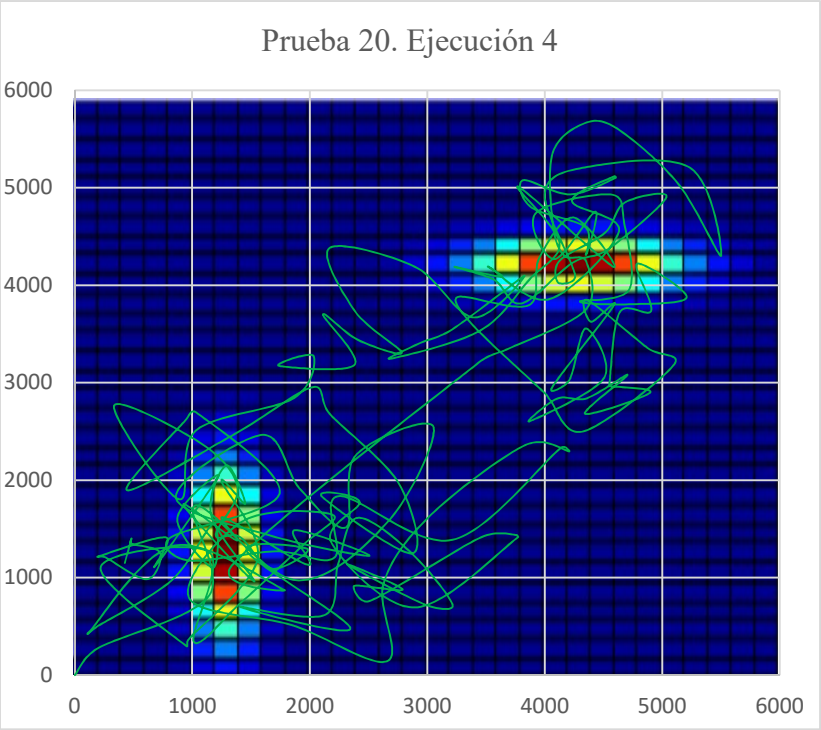
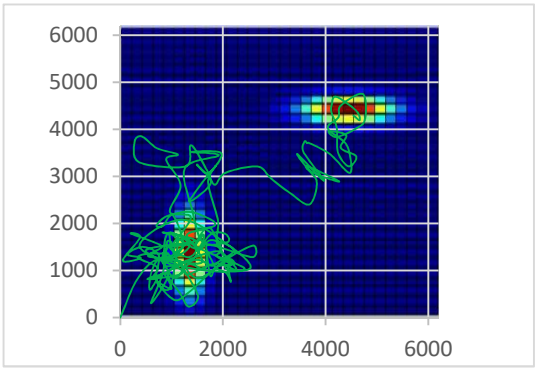


Figura 17. Solución Parametrización 20 - Ejecución 4. Trayectoria

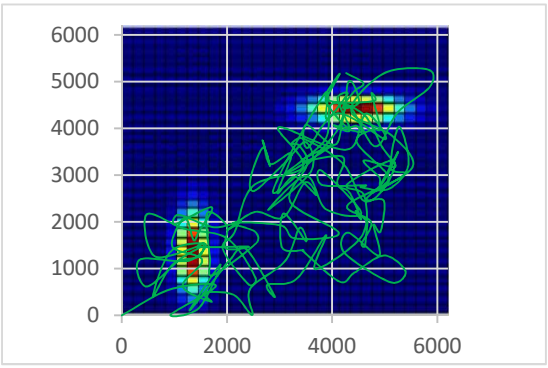
Como se puede deducir del análisis visual, el tipo de perturbación 3, al perturbar de forma diferente a cada parámetro de la solución, genera demasiado "ruido" en la solución final, a pesar de que la probabilidad acumulada sea mayor. Por tanto, podemos concluir que su funcionamiento dista más de la solución óptima que se podría implementar en un UAV real.

A continuación se muestran las mejores ejecuciones de cada una de las parametrizaciones sombreadas en verde (4, 8, 12, 16, 20 y 24), para la realización del análisis visual.



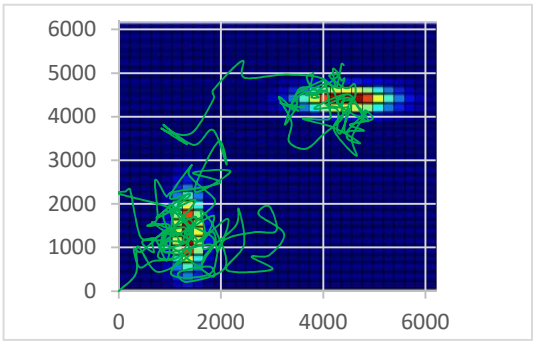
Parametrización 4. Ejecución 5

OPert=1, OBase=1



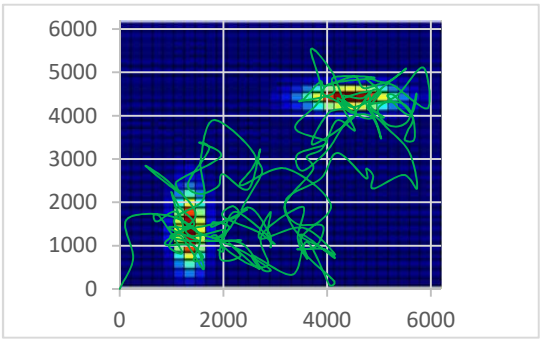
Parametrización 8. Ejecución 2

OPert=1, OBase=2



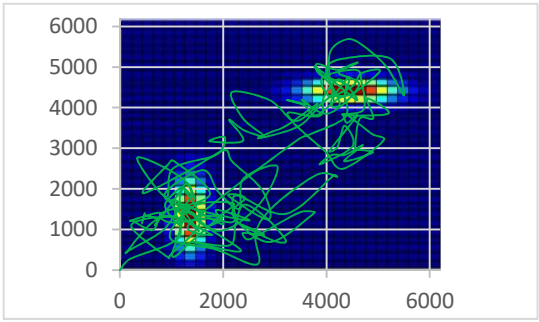
Parametrización 12. Ejecución 5

OPert=2, OBase=1



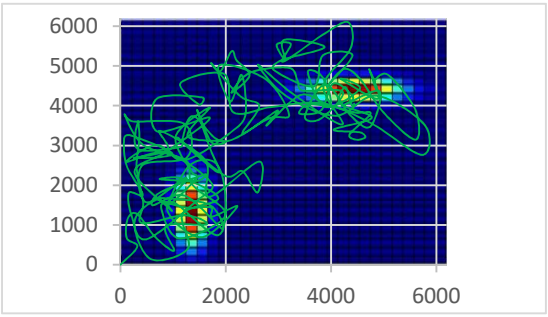
Parametrización 16. Ejecución 5

OPert=2, OBase=2



Parametrización 20. Ejecución 4

OPert=3, OBase=1



Parametrización 24. Ejecución 3

OPert=3, OBase=2

Figura 18. Trayectorias mejores resultados de las parametrizaciones preliminares 1-25

Como se puede deducir del análisis visual, las perturbaciones de tipo 2 y 3 (parametrizaciones 12, 16, 20 y 24) son las que nos facilitan trayectorias capaces de detectar mejor las distintas áreas de probabilidad. Además, la perturbación 3, tal y como se ha comentado anteriormente, nos facilita trayectorias más "ruidosas" o "erráticas" (parametrizaciones 20 y 24).

La perturbación tipo 1 (pruebas 4 y 8) parece no tener un buen comportamiento en áreas con varias zonas "calientes" y que queda atrapada en óptimos locales. Hecho que se puede confirmar al ver que en la mayor parte de los casos, la iteración en la que se produce el último cambio es bastante baja, y cuando es alta el cambio apenas es significativo en el candidato solución (es prácticamente idéntico a su predecesor).

4.2 Pruebas 26-73

Teniendo en cuenta los resultados anteriores, las mejores parametrizaciones tienen como elemento común que la selección de la base es aleatoria (OBase=1). En cuanto al tipo de perturbación, la respuesta ya no es tan contundente, aunque la tipo 2 (OPert=2), parece ser la que mejores resultados proporciona.

Por todo ello, analizaremos los tres tipos de perturbación con la selección de la base aleatoria, modificando otros parámetros como los valores que puede adoptar el factor F, el número de individuos de la población inicial, el número de iteraciones e incluso el factor diferencial que determina la generación de las variables de decisión en cada candidato.

La definición de cada grupo de parametrizaciones que se muestran a continuación en las tablas 7.1 y 7.2, ha sido el resultado de un trabajo evolutivo. En función de los resultados obtenidos en algunas configuraciones se ha decidido implementar las siguientes, con el único objeto de hacer un estudio completo del comportamiento de nuestro algoritmo.

P	wp	Iter	Prob	Pob_Ini	Auton	TPert	TBase	Dif	Fconst	F
26	200	600	0,9	40	50000	1	1	5	0.25+0.75*rand	0.25+0.75*rand
27	200	600	0,9	60	50000	1	1	5	0.25+0.75*rand	0.25+0.75*rand
28	200	600	0,9	40	50000	2	1	5	0.25+0.75*rand	0.25+0.75*rand
29	200	600	0,9	60	50000	2	1	5	0.25+0.75*rand	0.25+0.75*rand
30	200	600	0,9	40	50000	3	1	5	0.25+0.75*rand	0.25+0.75*rand
31	200	600	0,9	60	50000	3	1	5	0.25+0.75*rand	0.25+0.75*rand
32	200	600	0,9	40	50000	1	1	5	rand	rand
33	200	600	0,9	60	50000	1	1	5	rand	rand
34	200	600	0,9	40	50000	2	1	5	rand	rand
35	200	600	0,9	60	50000	2	1	5	rand	rand

Tabla 7.1. Parametrizaciones 26-35

P	wp	Iter	Prob	Pob_Ini	Auton	TPert	TBase	Dif	Fconst	F
36	200	600	0,9	40	50000	3	1	5	rand	rand
37	200	600	0,9	60	50000	3	1	5	rand	rand
38	200	1000	0,9	40	50000	1	1	5	0.5+1.5*rand	0.5+1.5*rand
39	200	1000	0,9	60	50000	1	1	5	0.5+1.5*rand	0.5+1.5*rand
40	200	1000	0,9	40	50000	2	1	5	0.5+1.5*rand	0.5+1.5*rand
41	200	1000	0,9	60	50000	2	1	5	0.5+1.5*rand	0.5+1.5*rand
42	200	1000	0,9	40	50000	3	1	5	0.5+1.5*rand	0.5+1.5*rand
43	200	1000	0,9	60	50000	3	1	5	0.5+1.5*rand	0.5+1.5*rand
44	200	3000	0,9	40	50000	2	1	5	0.25+0.75*rand	0.25+0.75*rand
45	200	3000	0,9	60	50000	2	1	5	0.25+0.75*rand	0.25+0.75*rand
46	300	600	0,9	40	50000	2	1	5	0.25+0.75*rand	0.25+0.75*rand
47	300	600	0,9	60	50000	2	1	5	0.25+0.75*rand	0.25+0.75*rand
48	300	3000	0,9	40	50000	2	1	5	0.25+0.75*rand	0.25+0.75*rand
49	200	600	0,9	40	50000	2	1	5	0.25+0.75*rand	0.25+0.75*rand
50	200	600	0,9	60	50000	2	1	5	0.25+0.75*rand	0.25+0.75*rand
51	300	600	0,9	40	50000	2	1	5	0.25+0.75*rand	0.25+0.75*rand
52	200	600	0,9	40	50000	2	1	10	0.25+0.75*rand	0.25+0.75*rand
53	200	1500	0,9	40	50000	2	1	10	0.25+0.75*rand	0.25+0.75*rand
54	200	600	0,9	40	50000	2	1	3	0.25+0.75*rand	0.25+0.75*rand
55	200	1500	0,9	40	50000	2	1	3	0.25+0.75*rand	0.25+0.75*rand
56	300	1500	0,9	40	50000	2	1	3	0.25+0.75*rand	0.25+0.75*rand
57	300	1500	0,9	40	50000	3	1	3	0.25+0.75*rand	0.25+0.75*rand
58	230	3000	0,9	40	50000	2	1	3	rand	rand
59	230	2000	0,9	40	50000	2	1	3	rand	rand
60	230	1500	0,9	40	50000	2	1	3	rand	rand
61	230	600	0,9	40	50000	2	1	3	rand	rand
62	200	600	0,9	40	50000	2	1	3	rand	rand
63	200	600	0,9	40	50000	2	2	3	rand	rand
64	200	600	0,9	40	50000	3	1	3	rand	rand
65	200	600	0,9	40	50000	3	2	3	rand	rand
66	200	600	0,9	40	50000	1	1	3	rand	rand
67	200	600	0,9	40	50000	1	2	3	rand	rand
68	200	600	0,9	40	50000	1	1	3	rand	rand
69	200	600	0,9	40	50000	2	1	3	rand	rand
70	200	600	0,9	40	50000	3	1	3	rand	rand
71	200	600	0,9	40	50000	1	1	2	rand	rand
72	200	600	0,9	40	50000	2	1	2	rand	rand
73	200	600	0,9	40	50000	3	1	2	rand	rand

Tabla 7.2. Parametrizaciones 36-73

Los resultados promedio obtenidos de las pruebas anteriores son los que se muestran en las siguientes tablas:

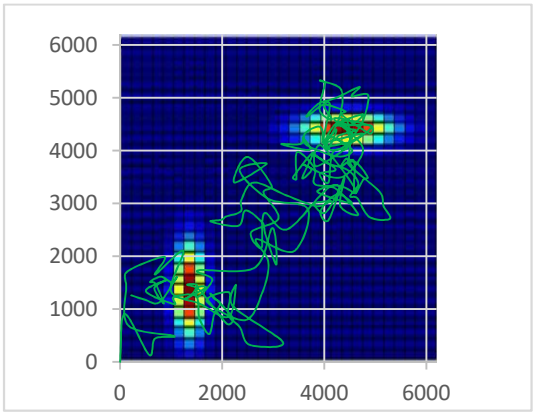
P	wp	POB_Ini	OPert	OBase	Pini	Pfin	%mejora	Iteración	Consumo	Nº cambios
26	200	40	1	1	0,39	0,60	54,70%	439	28.965,81	297
27	200	60	1	1	0,37	0,79	114,48%	597	27.644,72	91
28	200	40	2	1	0,37	0,72	99,08%	594	31.763,15	171
29	200	60	2	1	0,37	0,79	118,00%	599	28.243,89	162
30	200	40	3	1	0,42	0,77	84,64%	598	36.508,61	88
31	200	60	3	1	0,28	0,77	290,14%	582	35.762,60	65
32	200	40	1	1	0,37	0,63	70,03%	359	30.923,61	218
33	200	60	1	1	0,33	0,81	215,01%	591	30.812,86	71
34	200	40	2	1	0,35	0,70	112,36%	599	29.146,78	289
35	200	60	2	1	0,32	0,67	143,85%	600	29.061,74	271
36	200	40	3	1	0,39	0,75	98,35%	596	38.146,30	113
37	200	60	3	1	0,37	0,77	129,37%	593	36.347,57	79
38	200	40	1	1	0,34	0,69	115,30%	991	43.111,39	130
39	200	60	1	1	0,30	0,79	184,55%	995	27.758,19	128
40	200	40	2	1	0,32	0,68	161,43%	984	32.643,16	93
41	200	60	2	1	0,35	0,70	133,28%	995	32.693,23	81
42	200	40	3	1	0,36	0,47	33,20%	631	43.671,86	63
43	200	60	3	1	0,26	0,48	117,99%	535	45.919,21	45
44	200	40	2	1	0,25	0,69	207,79%	2999	28.162,53	1066
45	200	60	2	1	0,32	0,80	158,84%	2996	29.495,66	802
46	300	40	2	1	0,27	0,67	190,95%	597	36.393,93	201
47	300	60	2	1	0,30	0,77	180,92%	598	38.015,82	146
48	300	40	2	1	0,35	0,72	116,05%	2998	37.303,99	1036
49	200	40	2	1	0,37	0,76	109,76%	599	29.869,67	160
50	200	60	2	1	0,31	0,66	135,13%	598	29.325,08	154
51	300	40	2	1	0,30	0,87	204,99%	596	38.227,66	155
52	200	40	2	1	0,24	0,48	125,74%	595	43.582,36	169
53	200	40	2	1	0,22	0,56	182,33%	1499	42.497,35	464
54	200	40	2	1	0,46	0,80	77,09%	599	20.581,09	180
55	200	40	2	1	0,39	0,72	87,34%	1499	19.376,98	497
56	300	40	2	1	0,54	0,83	55,05%	1499	27.678,87	498
57	300	40	3	1	0,54	0,87	61,12%	1291	33.845,11	172
58	230	40	2	1	0,43	0,79	83,79%	2183	23.263,73	1101
59	230	40	2	1	0,43	0,77	79,85%	1999	21.194,57	984
60	230	40	2	1	0,47	0,74	62,18%	1499	23.065,66	786
61	230	40	2	1	0,43	0,72	68,02%	599	22.678,11	306
62	200	40	2	1	0,44	0,73	67,94%	600	22.329,05	286
63	200	40	2	2	0,45	0,60	32,05%	34	24.556,54	33

Tabla 8.1. Resultados parametrizaciones 26-63

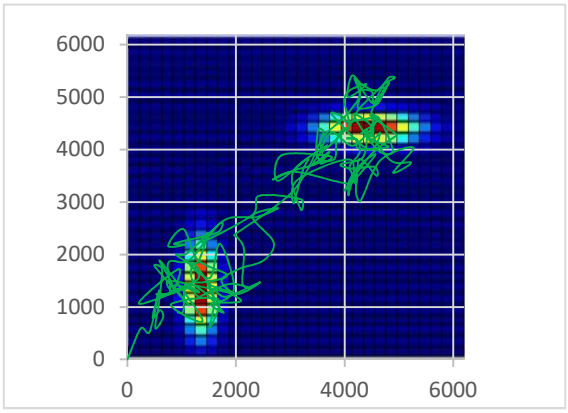
P	wp	POB_Ini	OPert	OBase	Pini	Pfin	%mejora	Iteración	Consumo	Nº cambios
64	200	40	3	1	0,43	0,80	86,28%	588	25.363,35	107
65	200	40	3	2	0,44	0,56	27,53%	147	27.701,08	134
66	200	40	1	1	0,43	0,70	66,80%	407	20.466,71	241
68	200	40	1	1	0,42	0,70	68,59%	596	28.088,87	71
69	200	40	2	1	0,44	0,74	70,09%	599	20.104,83	298
70	200	40	3	1	0,40	0,77	90,51%	598	24.338,55	112
71	200	40	1	1	0,43	0,48	12,85%	44	14.569,89	24
72	200	40	2	1	0,42	0,55	32,93%	599	12.960,51	308
73	200	40	3	1	0,42	0,76	82,16%	595	16.694,73	116

Tabla 8.2. Resultados parametrizaciones 64-73

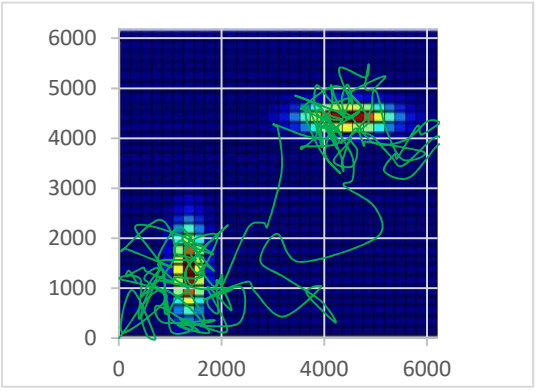
A continuación se muestran algunas de las trayectorias obtenidas en estas parametrizaciones, sobre las que plantearemos algunas conclusiones:



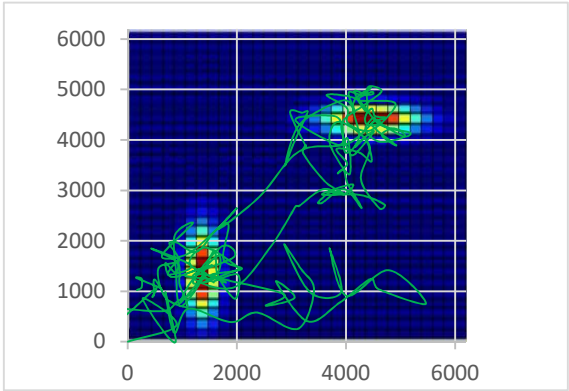
Parametrización 27. OPert=1, OBase=1, F=rand, PobIni=60, dif=5



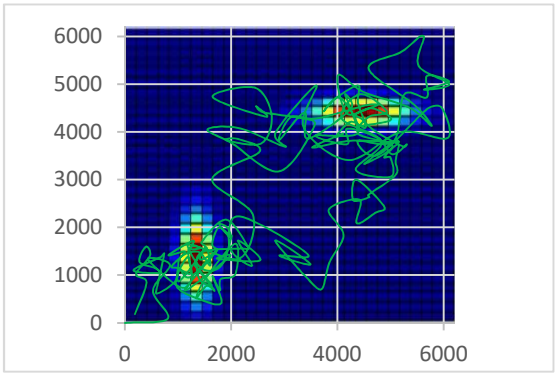
Parametrización 29. OPert=2, OBase=1, F=rand, dif=5



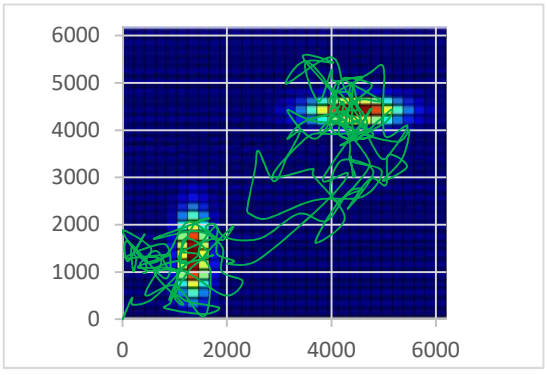
Parametrización 31. OPert=3, OBase=1, F=rand, dif=5



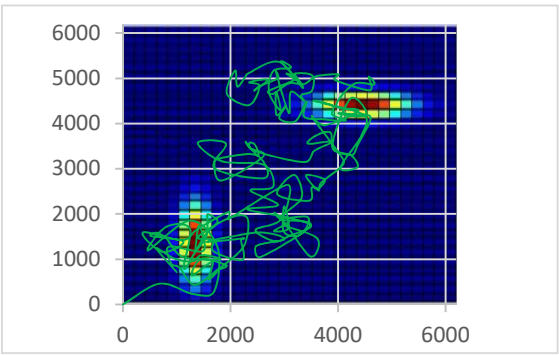
Parametrización 33. OPert=1, OBase=1, F=(0.25+0.75·rand), PobIni=60, dif=5



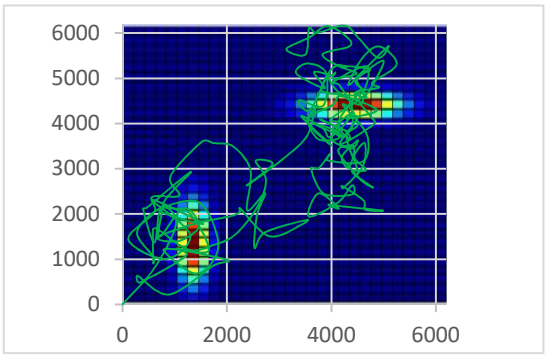
Parametrización 35. OPert=2, OBase=1 F=(0.25+0.75·rand), dif=5



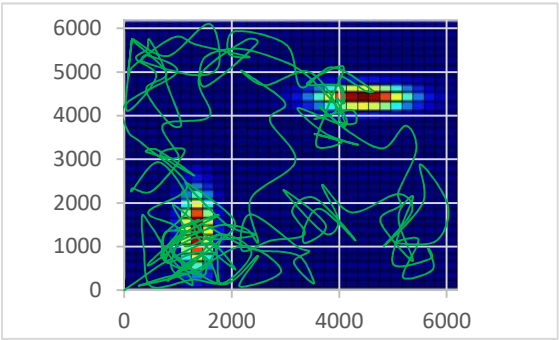
Parametrización 37. OPert=3, OBase=1 F=(0.25+0.75·rand), dif=5



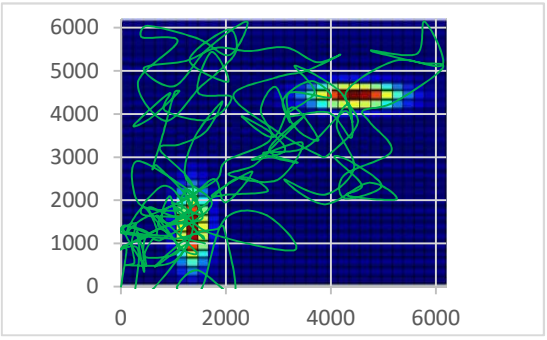
Parametrización 26. OPert=1, OBase=1 F=rand, PobIni=40, dif=5



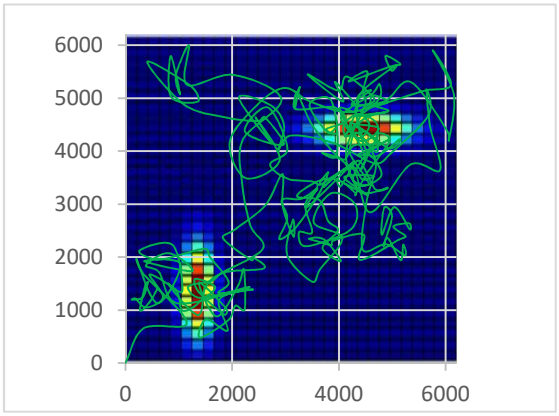
Parametrización 32. OPert=1, OBase=1 F=(0.25+0.75·rand), PobIni=40, dif=5



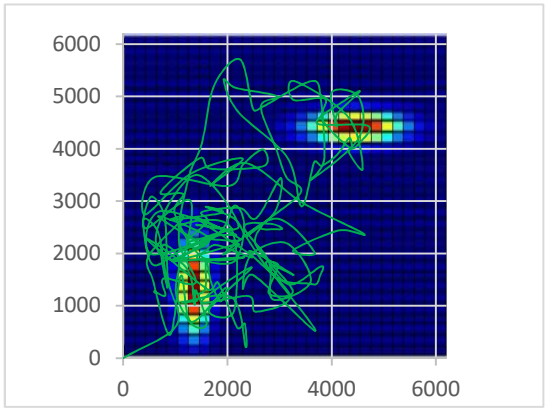
Parametrización 42. OPert=3, OBase=1 F=(0.5+1.5·rand), iter=1000, dif=5



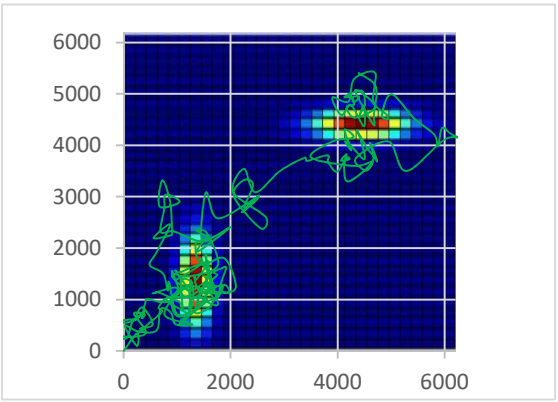
Parametrización 43. OPert=3, OBase=1 F=(0.5+1.5·rand), iter=1000, dif=5



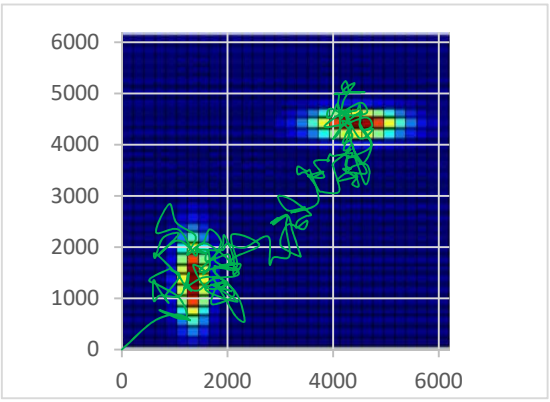
Parametrización 51. OPert=2, OBase=1 wp=300



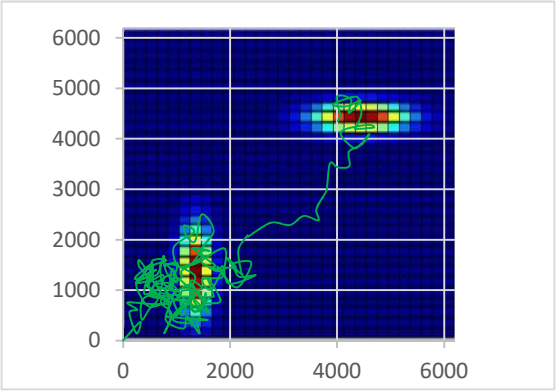
Parametrización 52. OPert=2, OBase=1 dif=10



Parametrización 58. OPert=2, OBase=1 dif=3



Parametrización 61. OPert=2, OBase=1 dif=3



Parametrización 62. OPert=2, OBase=1 dif=3

Figura 19. Selección trayectorias de pruebas preliminares 26-73

Tras el análisis de las pruebas realizadas, se pueden obtener las siguientes conclusiones:

- En las parametrizaciones de la 26 a la 37, evaluamos cómo se comportan las versiones de *opción perturbación* y *opción base*, modificando los valores que

puede adoptar F. F podrá adoptar un valor aleatorio entre (0,1) en un caso (pruebas de la 26 a la 31), y aleatorio entre (0.25,1) en otro (pruebas de la 32 a la 37). Se puede observar que incide principalmente sobre las perturbaciones de tipo 2, en cuanto a la probabilidad de detección acumulada. En general, en cualquiera de las parametrizaciones se genera mayor "distorsión" en las combinaciones donde F puede tomar valores menores (F=rand). A este respecto, es conveniente observar las gráficas de las parametrizaciones 27, 29, 31, 33, 35, 37

- También en estas pruebas comprobamos cómo afecta el número de individuos de la población inicial. Esto tampoco supone un cambio significativo salvo para la configuración de perturbación tipo 1. Tiene sentido, porque al ser el tipo de perturbación que menos facilita la exploración de nuevas áreas del mapa, el hecho de incrementar la población incide directamente en mejorar la exploración y por consiguiente el resultado final. A este respecto, es conveniente observar las gráficas de las parametrizaciones 26, 27, 32, 33.
- En las pruebas de las parametrizaciones 38 a la 43 incrementamos notablemente las iteraciones y el factor F. Apenas se producen cambios respecto a las configuraciones anteriores salvo en el caso de la perturbación tipo 3, donde se reduce la probabilidad acumulada obtenida, seguramente debido a incrementar demasiado el "ruido" en la exploración del mapa. A este respecto, es conveniente observar las gráficas de las parametrizaciones 42 y 43.
- El incremento de iteraciones (de 600 a 2000/3000) mejora ligeramente el resultado obtenido (apenas unas centésimas), en la probabilidad de detección acumulada, pero los tiempos de computación se disparan considerablemente. Por ello podemos afirmar que no resulta "rentable" el incremento en la probabilidad de detección frente al tiempo empleado en su cálculo.
- El incremento del número de waypoints, sí muestra una relación directa con los resultados obtenidos, lo cual tiene sentido teniendo en cuenta que el número seleccionado inicialmente coincide con el número de celdas que definen nuestra área de búsqueda. Pero genera bastante "distorsión" en la trayectoria. A este respecto, es conveniente observar las gráficas de las parametrizaciones 51.
- Incrementar el valor del diferencial, empeora los resultados. Parece existir una relación entre dicho valor y el tamaño de nuestras áreas de probabilidad. De hecho, en el caso del mapa básico de referencia, los mejores resultados se obtienen para

valores de diferencial entre 3 y 5. Nótese que las gaussianas tienen unas dimensiones medias de 3 y 5 celdas, según la dirección. A este respecto, es conveniente observar las gráficas de las parametrizaciones 52, 58, 61 y 62.

De todo lo visto hasta el momento, podemos concluir que las parametrizaciones de nuestro algoritmo que proporcionan unos resultados razonablemente buenos en un espacio de tiempo corto son los siguientes:

WP	Iteraciones	POB_Ini	O_Pert	O_Base	Fconst	F	Diferencial
200	600	40	2	1	rand	rand	3
200	600	40	3	1	rand	rand	3

Tabla 9. Parametrizaciones mejores para el escenario básico de referencia

Si observamos gráficamente las trayectorias obtenidas, las que tienen OPert=3 son más "ruidosas" que las OPert=2. Por ello, teniendo en cuenta que las probabilidades son similares podemos decir que funciona mejor OPert=2.

Para poder analizar visualmente con más detalle las soluciones obtenidas con las parametrizaciones elegidas, en el anexo del presente documento se muestra la evolución comparativa de las parametrizaciones de la tabla 9 en distintas iteraciones.

A continuación, debemos verificar si la bondad de la configuración sufre una dependencia operativa con el escenario o si por el contrario es adecuada independientemente del mismo.

4.3 Pruebas sobre otros escenarios

Tal y como se ha indicado anteriormente, analizaremos otros dos escenarios. En este caso son mapas de creencia con sólo un área "caliente", de diferente tamaño y no siguen una distribución de probabilidad conocida.

4.3.1 Escenarios 2 y 3

Los nuevos escenarios son los que se muestran a continuación:

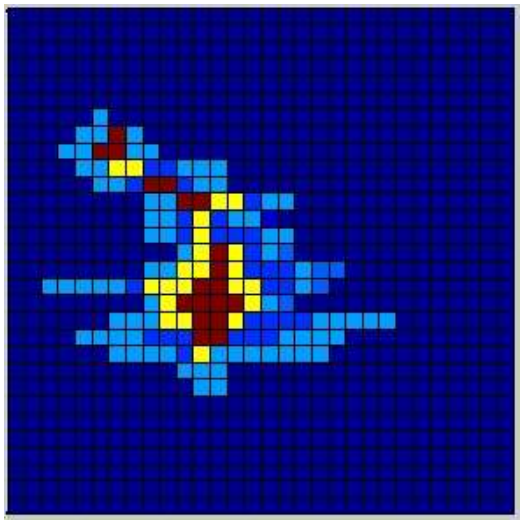


Figura 20. Escenario 2

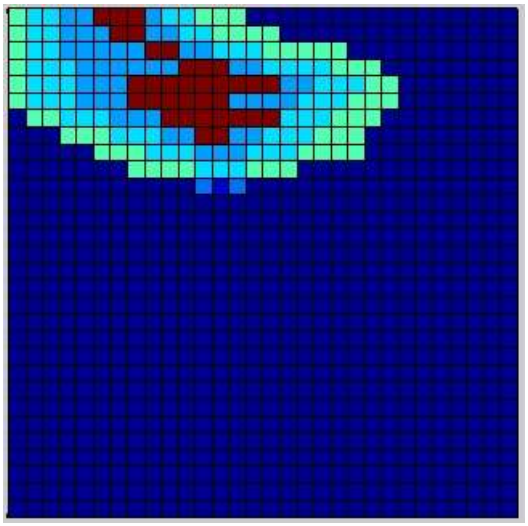


Figura 21. Escenario 3

Del análisis particular de los escenarios, se deduce que el número mínimo de waypoints a considerar para garantizar que se puedan alcanzar probabilidades aceptables son 150 para el escenario 2 y 200 para el escenario 3.

Se implementa un análisis básico del algoritmo teniendo en cuenta las mejores parametrizaciones obtenidas para el escenario básico de referencia, con las siguientes consideraciones:

- Número de waypoints, definido por el mapa de probabilidad (wp=150 para el escenario 2, wp=200 para el escenario 3)
- Población inicial ≥ 40
- Selección de la base aleatoria (Obase=1)
- Diferencial ≤ 5

  Iteraciones ~ 600

Las parametrizaciones analizadas en esta sección son las siguientes:

Prueba	wp	Iterac	Probab	PobIni	Auton	OPert	OBase	Diferencial	Fconst	F
1	150/200	600	0,9	40	50000	2	1	3	rand	rand
2	150/200	600	0,9	40	50000	2	1	2	rand	rand
3	150/200	600	0,9	40	50000	2	1	5	rand	rand
4	150/200	600	0,9	40	50000	3	1	3	rand	rand
5	150/200	600	0,9	40	50000	3	1	2	rand	rand
6	150/200	600	0,9	40	50000	3	1	5	rand	rand
7	150/200	600	0,9	40	50000	1	1	3	rand	rand
8	150/200	600	0,9	40	50000	1	1	2	rand	rand
9	150/200	600	0,9	40	50000	1	1	5	rand	rand
10	150/200	600	0,9	40	50000	2	2	3	rand	rand
11	150/200	600	0,9	40	50000	2	2	2	rand	rand
12	150/200	600	0,9	40	50000	2	2	5	rand	rand

Tabla 10. Parametrización en escenarios 2 y 3

En este caso, y por limitaciones de tiempo de cómputo, se han realizado 5 ejecuciones de cada parametrización y escenario, para poder extraer unas conclusiones preliminares. Además se han implementado 3 parametrizaciones (con 5 ejecuciones) con la selección de la base, según la aptitud del candidato, para verificar si el comportamiento del algoritmo es peor que con la opción de selección aleatoria, tal y como ocurría en el escenario básico.

A continuación se muestran los resultados obtenidos

Escenario 2

P	wp	Pob Ini	OPert	OBase	Pini	Pfin	%mejora	Iteración	Consumo	Nº cambios
1	150	40	2	1	0,37	0,61	67,50%	600	13.692,10	302
2	150	40	2	1	0,37	0,57	55,46%	599	10.015,26	305
3	150	40	2	1	0,30	0,66	123,91%	600	22.953,12	295
4	150	40	3	1	0,35	0,66	89,28%	600	17.834,48	137
5	150	40	3	1	0,37	0,64	75,23%	597	14.069,60	129
6	150	40	3	1	0,32	0,63	97,03%	599	30.162,85	130
7	150	40	1	1	0,40	0,67	67,02%	600	16.399,79	169
8	150	40	1	1	0,32	0,47	48,33%	38	12.413,50	20
9	150	40	1	1	0,32	0,70	119,70%	598	19.582,89	167
10	150	40	2	2	0,42	0,50	18,23%	40	19.546,10	39
11	150	40	2	2	0,38	0,43	14,75%	30	13.491,34	30
12	150	40	2	2	0,37	0,49	32,60%	39	29.290,37	38

Tabla 11. Resultados de las parametrizaciones analizadas sobre el escenario 2

Escenario 3

P	wp	Pob Ini	OPert	OBase	Pini	Pfin	%mejora	Iteración	Consumo	Nº cambios
1	200	40	2	1	0,44	0,57	30,41%	600	19.934,32	288
2	200	40	2	1	0,34	0,49	43,67%	599	16.261,13	278
3	200	40	2	1	0,36	0,58	66,92%	600	32.644,68	304
4	200	40	3	1	0,36	0,60	69,13%	599	23.480,50	122
5	200	40	3	1	0,33	0,59	77,56%	593	17.949,92	115
6	200	40	3	1	0,32	0,65	134,67%	599	39.965,03	113
7	200	40	1	1	0,36	0,45	24,27%	56	25.213,79	29
8	200	40	1	1	0,36	0,58	61,05%	597	18.712,54	64
9	200	40	1	1	0,40	0,64	60,20%	595	25.678,82	120
10	200	40	2	2	0,35	0,44	24,15%	30	24.880,96	29
11	200	40	2	2	0,32	0,41	26,16%	33	21.144,95	31
12	200	40	2	2	0,42	0,54	29,79%	37	32.779,13	36

Tabla 12. Resultados de las parametrizaciones analizadas sobre el escenario 3

De los resultados obtenidos podemos extraer las siguientes conclusiones:

Para la obtención de buenos resultados es importante **el tamaño del área de probabilidades**. El valor del diferencial debe facilitar a nuestro algoritmo explorar la zona caliente. En este caso, al tratarse de áreas con distancias de extremo a extremo elevadas, el diferencial de 5 permite obtener mejores resultados (frente a diferenciales de 2 y 3). Se han hecho pruebas posteriores incrementando el valor del diferencial y observamos que el nivel de "agitación" es tan alto que el algoritmo no llega a converger.

A continuación se muestran los resultados de la siguiente parametrización con diferencial de 10:

WP	Iter	Prob	PobIni	Auton	OPert	OBase	Ejecuciones	Diferencial	Fconst	F
200	600	0,9	40	50000	2	1	10	10	rand	rand

Tabla 13. Parametrización escenario 3 con diferencial de 10

Ejecución	Pini	Pfin	Iteración	Consumo	Distancia	Nº cambios
1,00	0,13	0,59	600	46.881,10	156.624,54	314
2,00	0,18	0,28	599	46.544,93	163.196,80	291
3,00	0,15	0,49	600	47.744,57	151.494,38	297
4,00	0,16	0,57	599	46.734,56	152.600,95	299
5,00	0,22	0,52	600	43.369,86	146.225,58	324
6,00	0,21	0,13	600	48.672,05	170.197,13	298
7,00	0,25	0,68	597	45.523,42	153.532,88	287
8,00	0,13	0,56	600	45.476,28	155.383,96	307
9,00	0,31	0,25	600	48.654,22	165.925,53	322
10,00	0,29	0,30	600	49.044,68	167.259,10	296

Tabla 14. Resultados de la Parametrización escenario 3, con diferencial de 10

También es relevante **la forma de nuestra zona caliente.** Mientras que en el primer escenario analizamos distribuciones gaussianas, en este caso las distribuciones son irregulares y esto implica que en perturbaciones de tipo 3 del valor del diferencial sea menos significativo que en el caso de las perturbaciones de tipo 1, donde el emplear un diferencial proporciona mayor "agitación" y facilita explorar nuevas áreas de búsqueda. El funcionamiento es mejor cuanto más parecidas son las dimensiones del área de probabilidad.

Por último, es importante **la distancia a la que se encuentre la zona de detección de la base del UAV**. Cuanto más grande, menor es la probabilidad detección acumulada para el mismo número de waypoints. Esto es debido a que la amplitud del diferencial es la misma para toda la trayectoria y nos obliga a tener un número de waypoints en áreas fuera de la zona caliente. Este número de waypoints crece con la distancia a dicha zona. De ahí que en el escenario 3 las probabilidades de detección acumuladas son significativamente menores que para el escenario 2.

Además, el tipo de cruce implementado, tampoco facilita el solventar este problema.

A continuación se presentan los resultados visuales de diferentes parametrizaciones.

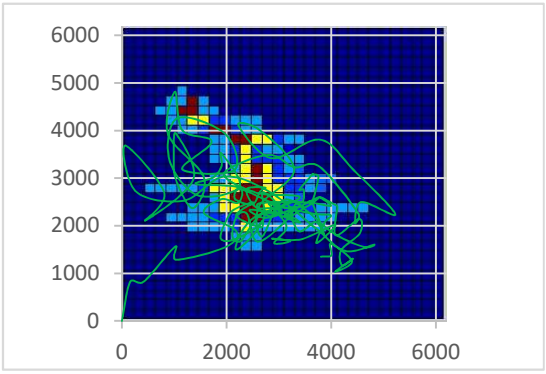
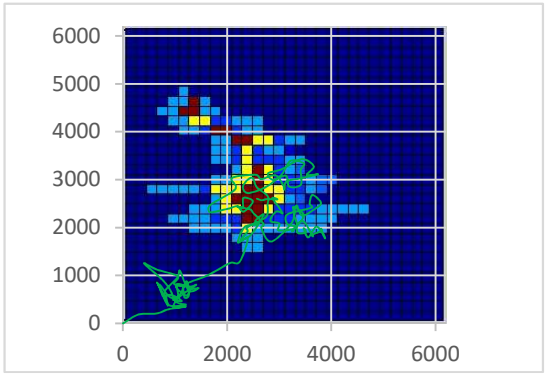
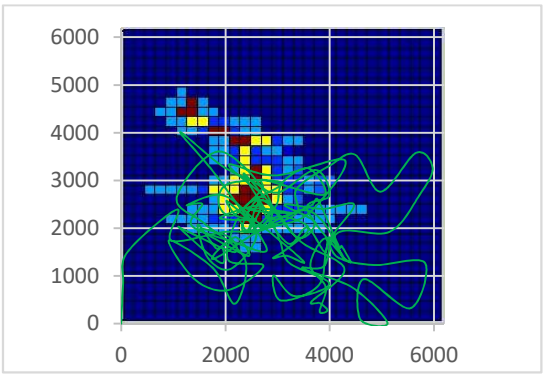
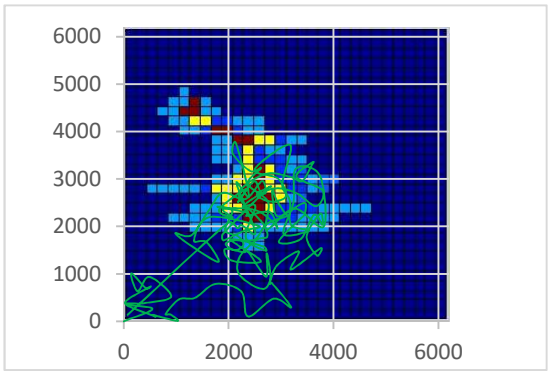
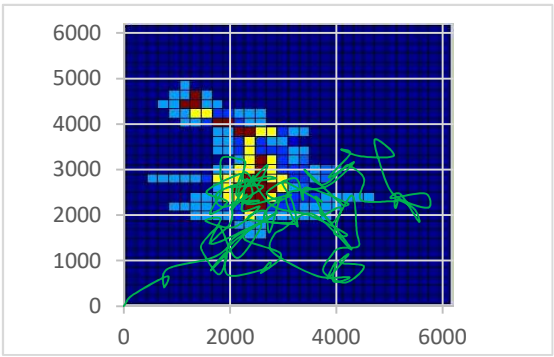
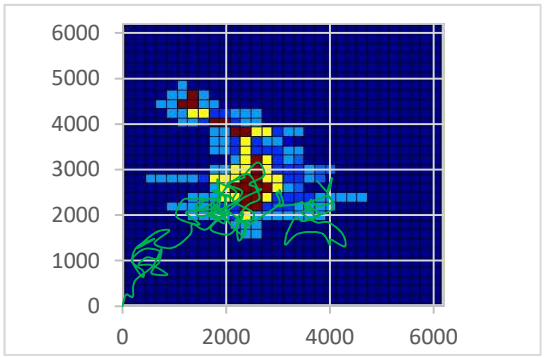
Escenario 2	
	
Parametrizacion 3. **OPert=2, OBase=1. Dif=5**	**Parametrizacion 2.** **OPert=2, OBase=1. Dif=2**
	
Parametrizacion 6. **OPert=3, OBase=1. Dif=5**	**Prueba 4. OPert=3, OBase=1. Dif=3**
	
Parametrizacion 9. **OPert=1, OBase=1. Dif=5**	**Parametrizacion 8.** **OPert=1, OBase=1. Dif=2**

Figura 22. Comparativa de resultados en escenario 2, según valor diferencial

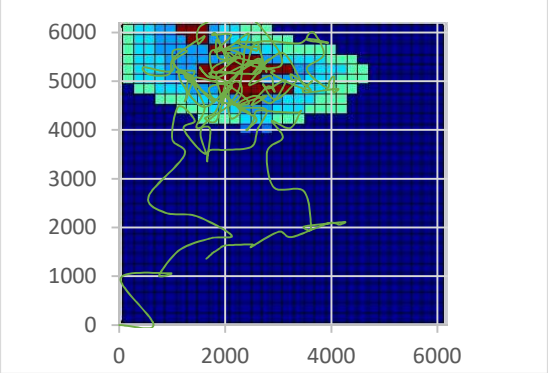
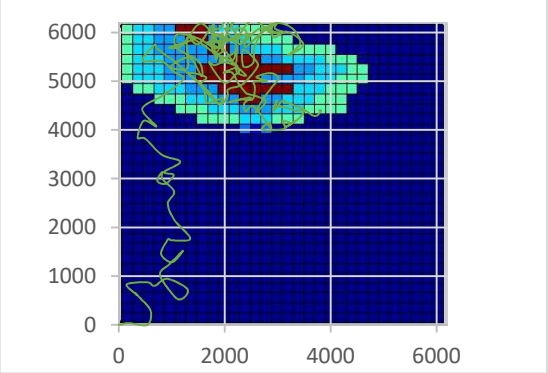
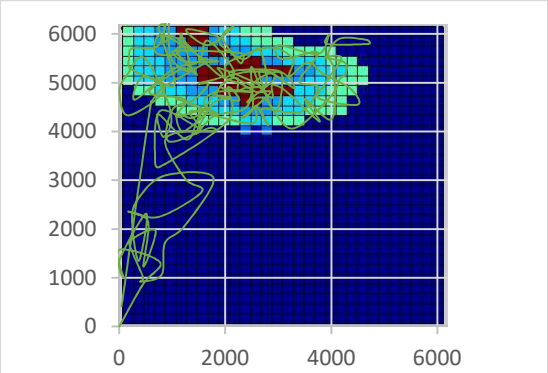
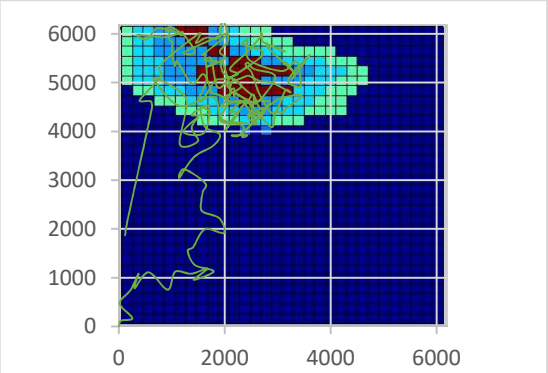
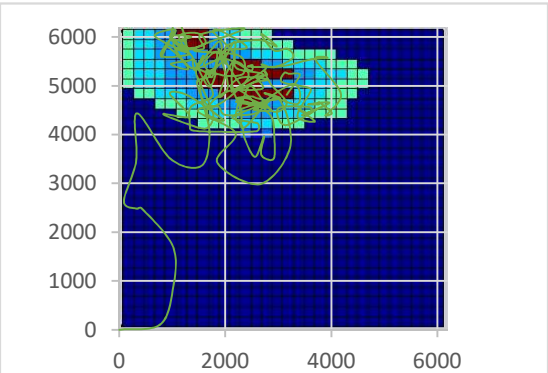
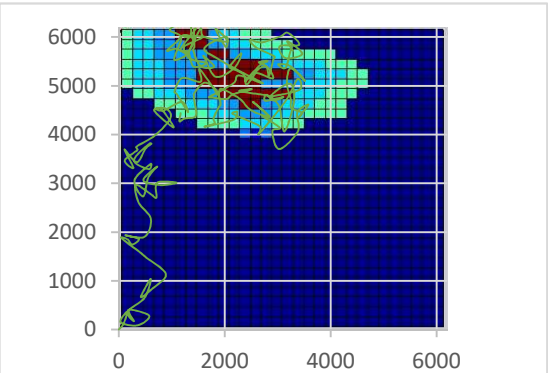
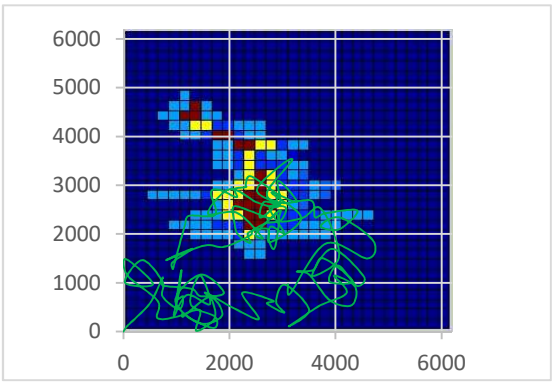
Escenario 3	
	
Parametrización 3. **OPert=2, OBase=1. Dif=5**	**Parametrización 2.** **OPert=2, OBase=1. Dif=2**
	
Parametrización 6. **OPert=3, OBase=1. Dif=5**	**Parametrización 5.** **OPert=3, OBase=1. Dif=2**
	
Parametrización 9. **OPert=1, OBase=1. Dif=5**	**Parametrización 8.** **OPert=1, OBase=1. Dif=2**

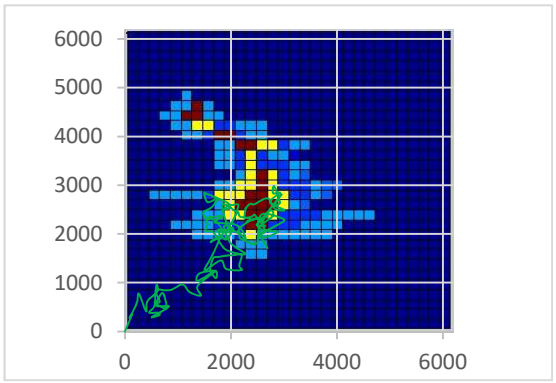
Figura 23. Comparativa de resultados en escenario 3, según valor diferencial

Tal y como vimos en el escenario básico, confirmamos que si se aplica la selección de base en función de la aptitud del candidato (parametrizaciones 10, 11 y 12), los resultados obtenidos son peores y el algoritmo se queda "atrapado" en un óptimo local. Además, no se producen cambios después de las primeras 40 iteraciones.

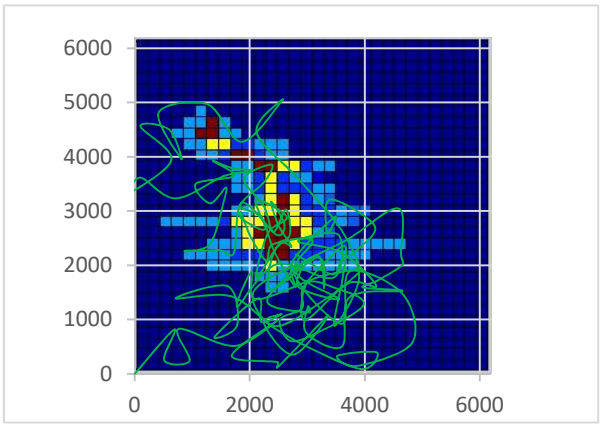
Escenario 2



Parametrización 10.
OPert=2, OBase=2. Dif=3



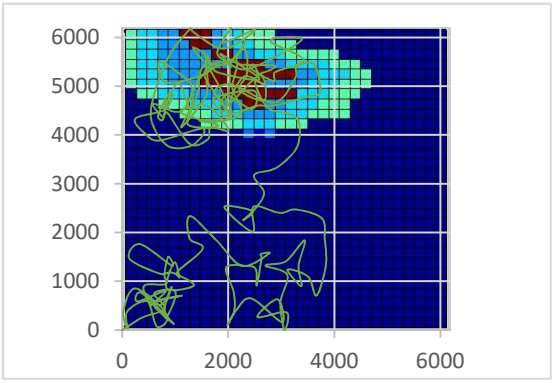
Parametrización 11.
OPert=2, OBase=2. Dif=2



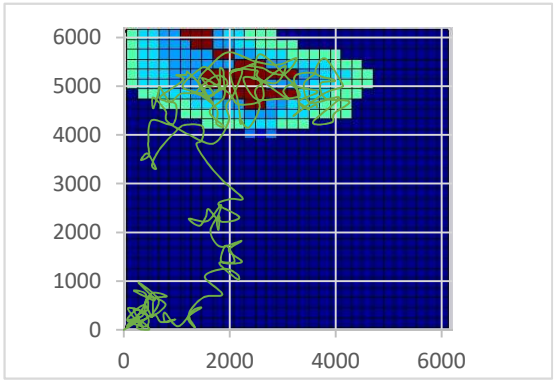
Parametrización 12.
OPert=2, OBase=2. Dif=5

Figura 24. Trayectorias escenario 2. Parametrizaciones 10, 11 y 12.

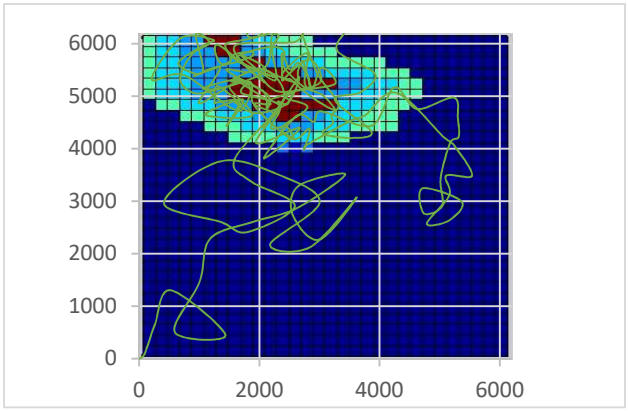
Escenario 3



Parametrizaciones 10.
OPert=2, OBase=2. Dif=3



Parametrizaciones 11.
OPert=2, OBase=2. Dif=2



Parametrizacines 12.
OPert=2, OBase=2. Dif=5

Figura 25. Trayectorias escenario 3. Parametrizaciones 10, 11 y 12.

Tras analizar las diferentes parametrizaciones y escenarios, se puede afirmar que la selección de la base aleatoria es mejor que una selección según la aptitud del candidato, que en la mayor parte de los casos condena al algoritmo a quedar "atrapado" en algún óptimo local.

También se puede afirmar que el tipo de perturbación 2, la perturbación que modifica el factor F para cada solución, es la que nos permite obtener una mejor relación *Tiempo empleado/Probabilidad acumulada*.

El resto de parametrizaciones varían según el escenario, lo que nos permite hacer la siguiente afirmación: **existe cierta dependencia de nuestro algoritmo con el escenario de búsqueda**.

Tal y como hemos ido observando con la realización de las múltiples pruebas en los diferentes escenarios, el número mínimo de waypoints vendrá determinado por el número de celdas que integren el área probabilidad de detección del objeto de estudio. Ahora bien, este número también estará condicionado por la distancia de la base de operaciones desde la que saldrá nuestro agente a la zona de probabilidad.

Las dimensiones del área de búsqueda en ambos ejes cartesianos inciden en la selección del valor diferencial. Este parámetro es fundamental en la generación de las trayectorias, ya que determina el rango de valores que pueden tomar los waypoints. Pero hay que respetar una relación de compromiso entre el incremento del diferencial y la distancia a la zona de probabilidad, ya que a mayor distancia, el diferencial debe ser más moderado porque genera una distorsión muy elevada en el tramo inicial de la trayectoria.

El número de áreas de probabilidad de detección y su posición también son factores a tener en cuenta. Hemos visto que nuestro algoritmo ha funcionado mejor en el primer escenario. Seguramente, esto último es debido al tipo de cruce implementado (cruce simple por un punto). Una de las áreas de búsqueda se encuentra próxima a la base de operaciones del UAV y el cruce por un punto facilita la obtención de soluciones perturbadas que son combinación de los tramos inicial y final de los padres. Sin embargo en escenarios donde sólo hay un área de búsqueda y, especialmente, en aquéllas que se encuentran alejadas de la base de operaciones del agente (escenario 3), se "desperdician" waypoints en el trayecto hacia la zona de detección de probabilidad, y el cruce por un punto dificulta el obtener rutas "directas" hacia la zona objetivo.

En el anexo del presente documento se muestra la evolución comparativa iterativa de parametrizaciones con distinto tipo de perturbación en ambos escenarios.

5. Conclusiones

En este capítulo final se abordarán tres aspectos relacionados con el trabajo realizado y la documentación aportada. Por un lado, se valorará el grado de consecución de los objetivos propuestos en el capítulo 1 de la presente memoria. A continuación se expondrán las conclusiones fruto del análisis de resultados. Y por último, se comentarán de forma breve las líneas de trabajo futuras propuestas.

5.1 Consecución de objetivos

Recopilando los objetivos propuestos en el capítulo 1 de la presente memoria, se considera que éstos han sido alcanzados completamente:

- Se ha llevado a cabo un análisis del modelado de los problemas de búsqueda de objetivos de forma probabilística mediante algoritmos heurísticos evolutivos. Se han descrito los elementos principales del problema y su planteamiento, desde los datos conocidos, hasta las variables de decisión, la función objetivo y las restricciones. También se ha determinado cuál es el papel que juega cada uno de los operadores del problema en su parametrización.
- Se ha recopilado información sobre diferentes métodos estudiados para la resolución de este tipo de problemas y los diferentes enfoques de las líneas de investigación: programación con restricciones, algoritmos basados en el método del gradiente, algoritmos basados en redes neuronales, algoritmos genéticos, etc.
- Se ha propuesto un algoritmo basado en evolución diferencial (DE), donde la principal diferencia respecto a otros trabajos de investigación es que las soluciones no se centran en la dinámica del UAV, sino que son los waypoints los que definen la trayectoria en el área de búsqueda. Dichos waypoints se implementan como desplazamientos diferenciales en los ejes de coordenadas de nuestro mapa de búsqueda. Los desplazamientos se pueden realizar a diferentes velocidades, limitadas por las velocidades de operación de UAVs reales. El consumo del UAV es una restricción que debe cumplirse garantizando el regreso del agente a la base. La

función a optimizar es la relación entre el tiempo empleado y la probabilidad de detección acumulada.

- Se ha probado el comportamiento del algoritmo en un escenario básico de referencia con dos áreas de probabilidad de detección, caracterizadas por sendas gaussianas, para definir una parametrización básica del algoritmo. Posteriormente se han probado parametrizaciones equivalentes en otros escenarios con número de áreas de detección y distribuciones diferentes. En todos los casos, tanto el objetivo como los escenarios son estáticos a lo largo del tiempo de búsqueda.
- Por último, teniendo en cuenta los resultados obtenidos en el capítulo 4 de evaluación de resultados, se han extraído las principales conclusiones sobre el trabajo realizado y líneas futuras de trabajo. Éste último punto es el contenido de los siguientes apartados del presente capítulo.

5.2 Análisis de resultados y conclusiones

De los resultados recogidos en el capítulo 4 sobre los diferentes escenarios propuestos obtenemos las siguientes conclusiones de carácter general:

- Una selección de la base aleatoria en el operador de perturbación es mejor que una selección según la aptitud del candidato (ya que esta última lleva al algoritmo a quedar "atrapado" en un óptimo local).
- El tipo de perturbación 2, aquélla en la que s*e emplea el mismo factor F para cada nuevo elemento de la población en todas sus variables de decisión,*, es la que nos permite obtener una mejor relación *Tiempo empleado/Probabilidad acumulada* durante la búsqueda del objetivo.
- Existe una cierta dependencia de nuestro algoritmo con el área de probabilidad de detección bajo estudio.

Además, se requiere cierto conocimiento previo para adecuar la parametrización del algoritmo al escenario de búsqueda propuesto y, por tanto, su aplicación está limitada a que se disponga de dicha información con carácter previo.

Son varios los aspectos del escenario a considerar para establecer la parametrización:

- El tamaño del área de búsqueda y su distancia a la base de operaciones del agente incide directamente sobre el número mínimo de waypoints de nuestro algoritmo.

- La selección del valor diferencial está condicionada por el compromiso entre las dimensiones del área de búsqueda en ambos ejes cartesianos y la distancia entre la zona de probabilidad y la base de operaciones.
- El número de áreas de probabilidad de detección y su posición en el mapa de búsqueda determinarán la selección del tipo de combinación más adecuada. De hecho esta es una de las líneas de trabajo futuras que se propone en el siguiente apartado.

5.3 Líneas de trabajo futuras

De acuerdo con lo expuesto anteriormente, se proponen ampliaciones de este Trabajo de Fin de Máster, que permitirían ahondar en las cuestiones ya planteadas:

- *Implementación de otras funciones para la generación de poblaciones.*

  El uso de diferenciales para la definición de los waypoints proporciona trayectorias poco "realistas" que puedan ser directamente aplicadas en UAVs y escenarios reales. Se podría implementar una función en la que la generación de los waypoints estuviese basada en coordenadas polares, donde se puede "controlar" el ángulo de giro de un waypoint al siguiente. El objetivo es la obtención de trayectorias menos erráticas que las obtenidas en el presente trabajo.
- *Ampliación de la familia de operadores del algoritmo: selección de la base, operador de combinación, etc.*

  Se podría analizar el comportamiento del algoritmo al emplear, por ejemplo, el método de torneo, seleccionando un subconjunto de individuos de la población inicial y eligiendo el mejor candidato de este subconjunto.

  Al mismo tiempo se podría llevar a cabo un estudio comparativo del comportamiento del algoritmo en diferentes escenarios, empleando distintas formas de combinación (cruce por N puntos, uniforme, aritmético, etc.), y evaluando la idoneidad de cada una según el mapa y las áreas de probabilidad de detección.

  También se podría sustituir la función objetivo propuesta, que es la razón entre el tiempo de búsqueda y la probabilidad acumulada durante la trayectoria, por otra en la que se comparasen ambas funciones de forma Pareto. Alternativamente, para optimizar el tiempo de búsqueda también se podría probar cómo funciona el método desarrollado cuando la función objetivo es el tiempo esperado (medio) de búsqueda

de la trayectoria, ya que esta función optimiza tanto el tiempo como la acumulación de la probabilidad de detección cuanto antes.

- *Escenarios más realistas.*

  Se podría evaluar cómo funciona el algoritmo sobre escenarios que presentan obstáculos, teniendo en consideración un parámetro adicional como es la altura de vuelo y un mapa que registre la información sobre la altura del terreno.

  También se podría analizar el comportamiento considerando que el mapa no es estático, situación muy común en escenarios marítimos. En este caso sería necesario disponer del modelo de transición del mapa de probabilidad del objetivo.

Bibliografía

[1] https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle

[2] Monográfico Sistemas Aéreos No Tripulados. Revista Bit-COIT. Autor: Mª Jesús Llorden.

[3] https://www.theuav.com/

[4] https://www.dji.com/

[5] Farzad Kamrani and Rassul Ayani. "UAV Path Planning in Search Operations".

[6] Víctor San Juan, Matilde Santos and José Manuel Andújar. "Intelligent UAV Map Generation and Discrete Path Planning for Search and Rescue Operations". (2018)

[7] Pérez Carabaza, S., Besada Portas, E., López Orozco, J. A., De la Cruz, J. M., A Real World MultiUAV Evolutionary Planner for Minimum Time Target Detection, Proceedings of the Genetic and Evolutionary Computation Conference 2016, (2016)

[8]Eva Besada-Portas, Luis de la Torre, Jesús M. de la Cruz, Member, IEEE, and Bonifacio de Andrés-Toro "Evolutionary Trajectory Planner for Multiple UAVs in Realistic Scenarios"

[9] Verónica González, Concepción A. Monje y Carlos Balaguer. "Planificación de trayectorias con vehículos aéreos no tripulados en un entorno aeroportuario". Dpto Ing Sistemas y Automática, Universidad Carlos III Madrid (2015)

[10] Berger, J., Barkaoui, M., Genetic Algorithm for In-Theatre Military Logistics Search-and-Delivery Path Planning, International Journal of Computer, Electrical, Automation, Control and Information Engineering, (2013)

[12] Sara Pérez Carabaza, Eva Besada Portas, José Antonio López Orozco, Jesús Manuel de la Cruz. "Resolución del problema de Búsqueda en Tiempo Mínimo mediante Colonias de Hormigas" Departamento de Arquitectura de Computadores y Automática. Universidad Complutense de Madrid

[13] Francisco Fernández Ramírez, David Sánchez Benitez, Eva Besada Portas, José A. López Orozco. "Coordinated sea rescue system based on unmanned air vehicles and surface vessels". (2011)

[14] Juan Carlos Rubio, Juris Vagners, Rolf Rysdyk. "Adaptive Path Planning for Autonomous UAV Oceanic Search Missions". (2004)

[15] Arce, F., López Orozco, J. A., Besada Portas, E., "Búsqueda de objetivos estáticos en tiempo mínimo mediante programación genética", TFM, Universidad Complutense de Madrid, Dpto. Arquitectura de Computadores y Automática (2015)

[16]Wan-liXiang, Xue-leiMeng, Mei-qingAn, Yin-zhenLi and Ming-xiaGao. "An Enhanced Differential Evolution Algorithm Based on Multiple Mutation Strategies". School of Traffic&Transportation, Lanzhou Jiaotong University, Lanzhou, Gansu 730070, China (2015)

[17] Petr Pošík, Václav Klems, "JADE, an Adaptive Differential Evolution Algorithm, Benchmarked on the BBOB Noiseless Testbed". Czech Technical University in Prague

[18] Lanillos, P., Besada Portas, E., López Orozco, J. A., De la Cruz, J. M., Minimum Time Search in Uncertain Dynamic Domains with Complex Sensorial Platforms, Sensors (Basel) 2014 August 4;14(8):14131-14179 (2014)

[19] Lanillos, P., Besada Portas, E., Pajares, G., Ruz, J. J., Minimum time search for lost targets using cross entropy optimization. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pages 602-609. (2012).

[20] Lanillos P., Gan S. K., Besada Portas, E., Pajares, G., Sukkarieh, S., Multi-UAV target search using decentralized gradient-based negotiation with expected observation, Information Sciences Journal, Volume 282, October 2014, p. 92–110 (2014)

[21] Lanillos, P., Yañez, J., Ruz, J. J., Besada-Portas, E., A bayesian approach for constrained multiagent minimum time search in uncertain dynamic domains, Proc. of the 15th Conf. on Genetic and Evolutionary Computation (GECCO), p. 391-398, July (2013)

[22]Apuntes Asignatura Optimización Heurística. Temas 1 y 4.

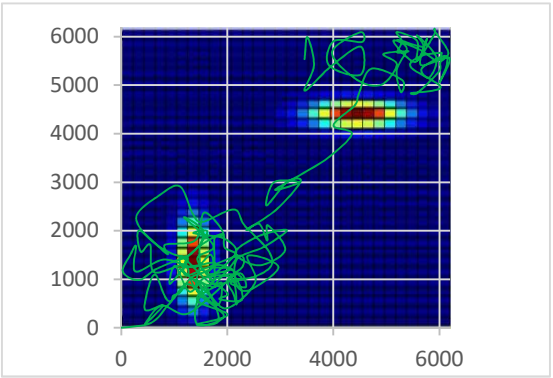
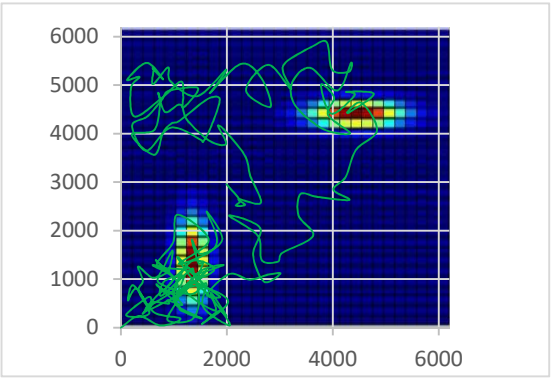
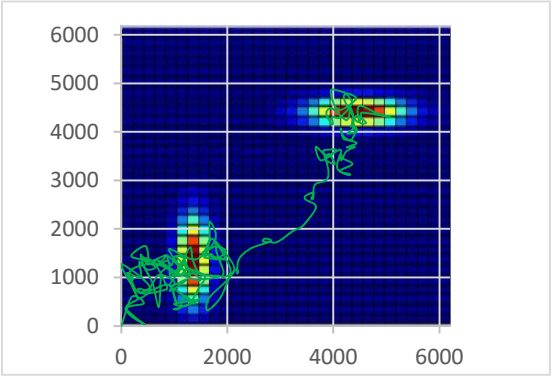
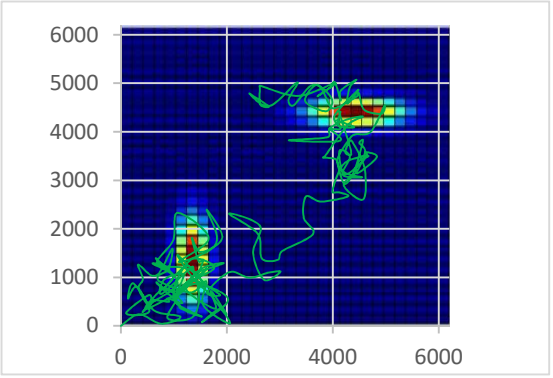
[23] Judith Manso "Búsqueda de objetivos estáticos en tiempo mínimo mediante redes neuronales". TFM 2013

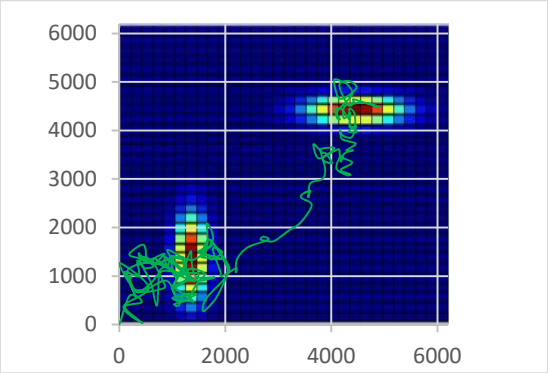
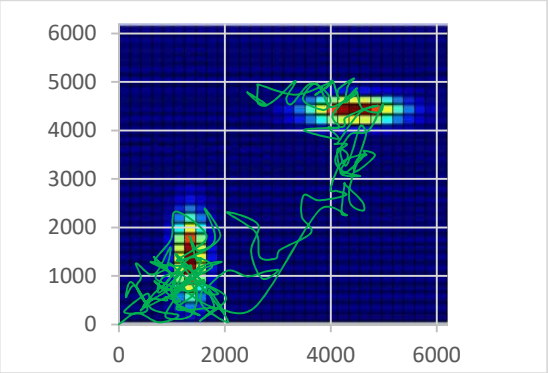
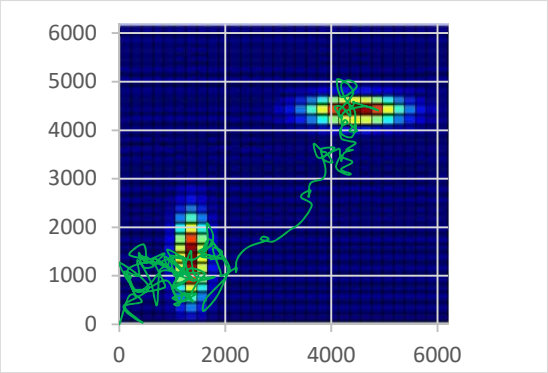
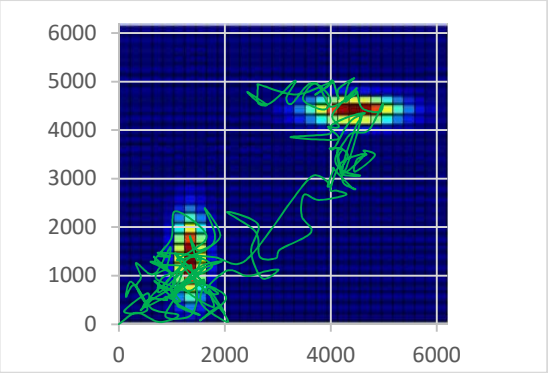
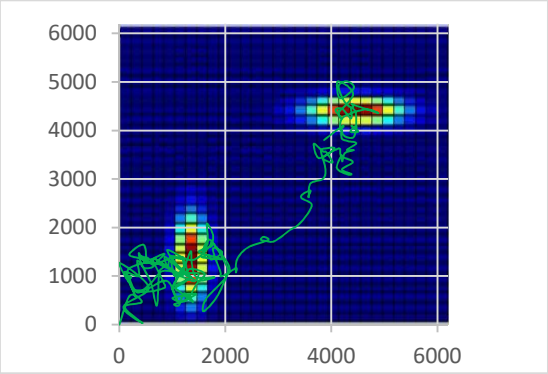
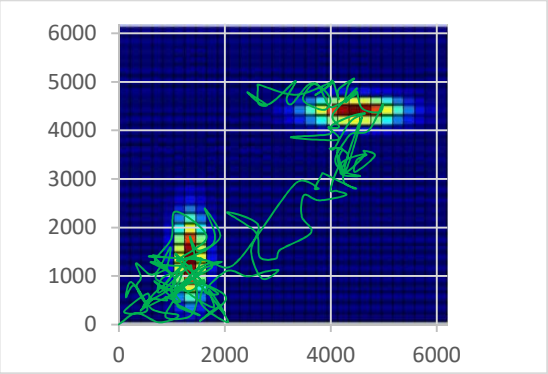
[24] Francisco Javier Huertos. "Optimización de trayectorias de UAVs para la rápida localización de objetivos mediante algoritmos de colonias de hormigas". TFM 2016

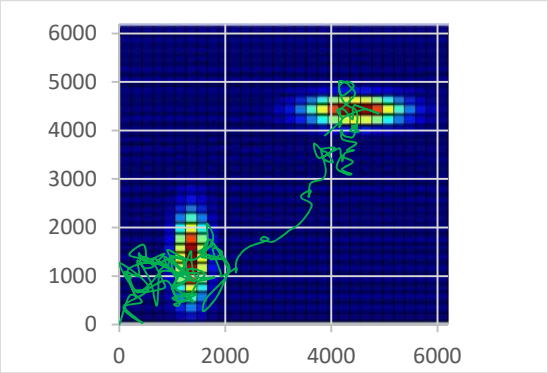
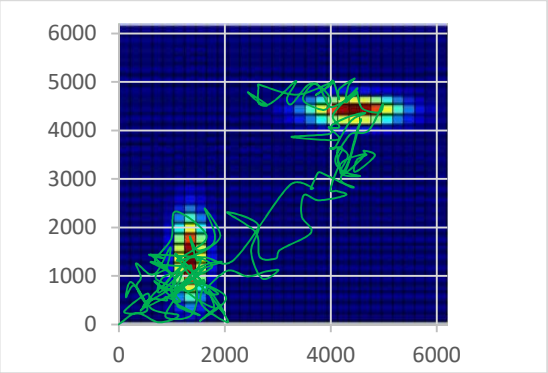
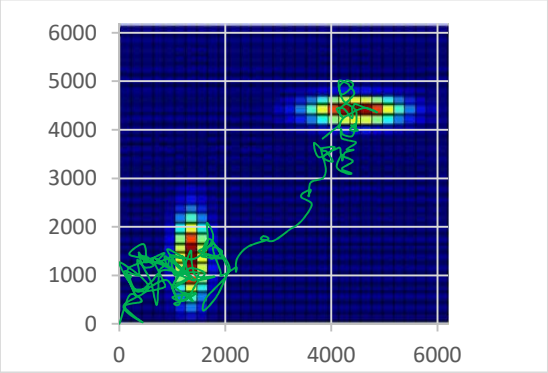
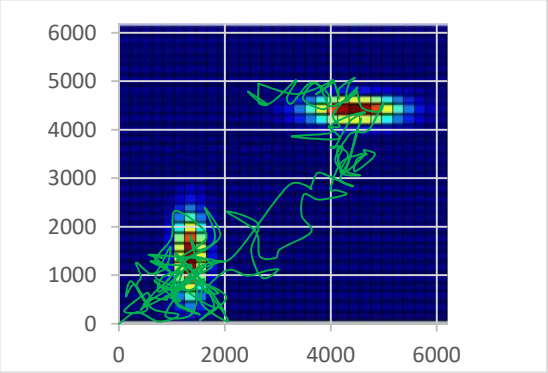
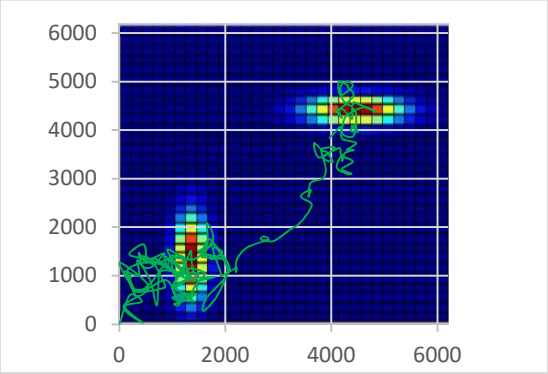
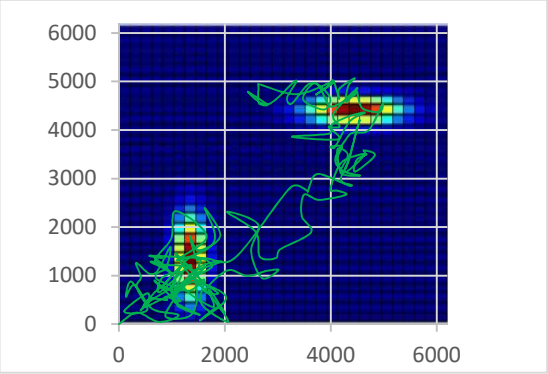
[25] Francisco Javier Yáñez. "Técnicas heurísticas para la búsqueda en tiempo mínimo (Comparativa)". TFM 2014

Anexo

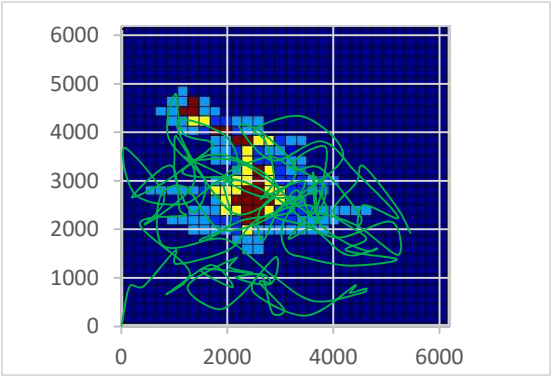
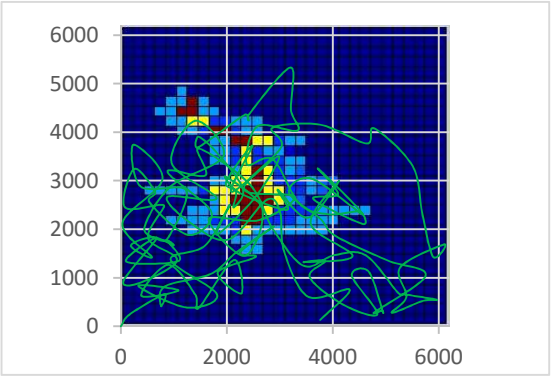
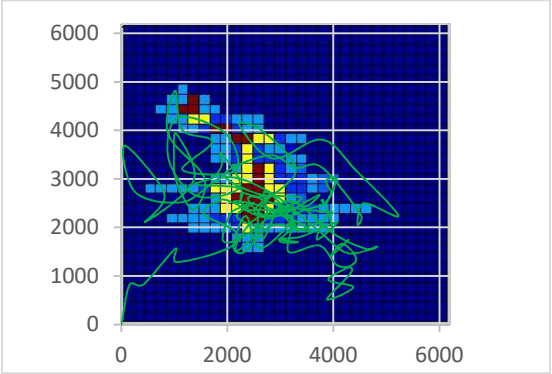
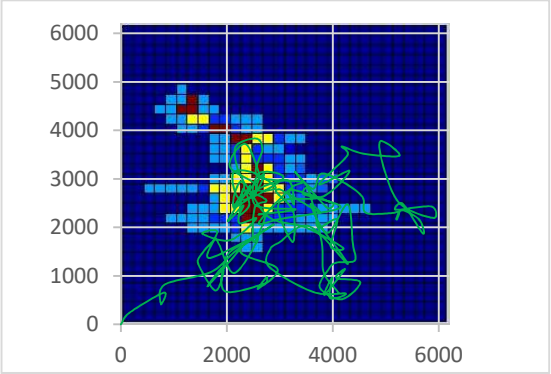
Muestra de representación evolutiva (iteraciones) de parametrizaciones óptimas para el escenario básico de referencia

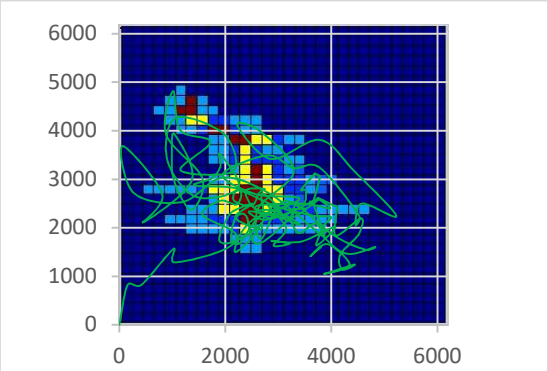
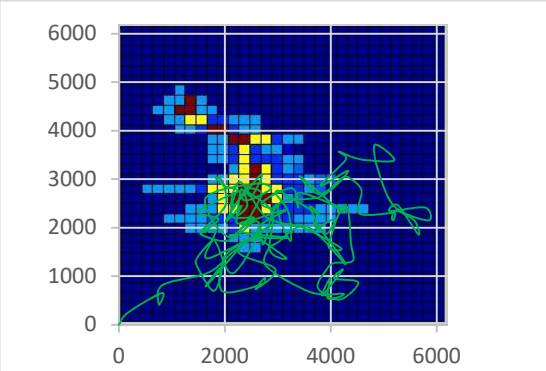
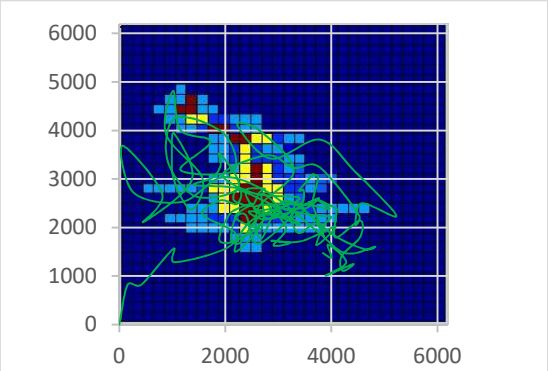
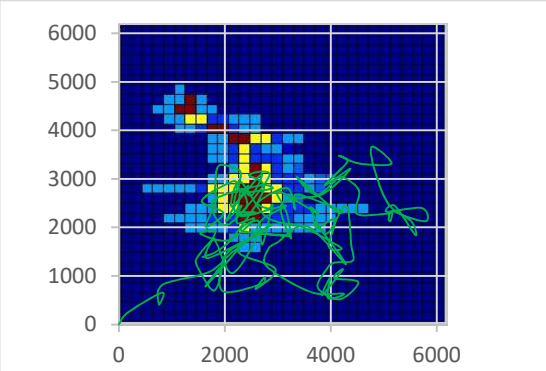
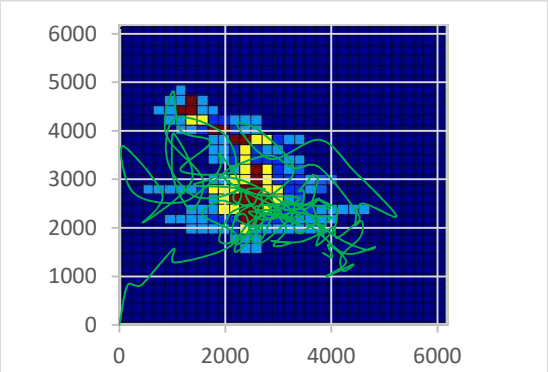
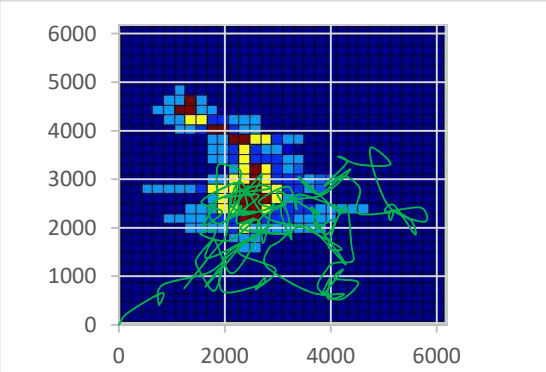
Parametrización **Opert=2, Obase=1, Diferencial=3**	**Parametrización** **Opert=3, Obase=1, Diferencial=3**
	
Iteración 1	**Iteración 1**
	
Iteración 50	**Iteración 50**

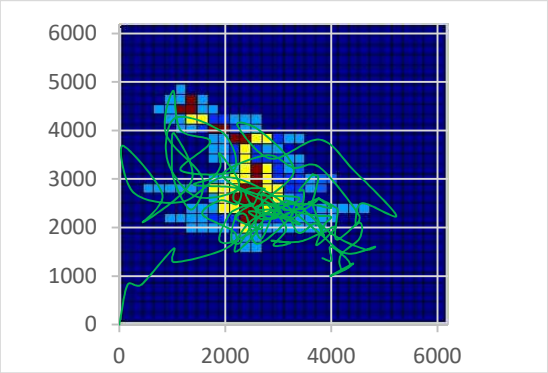
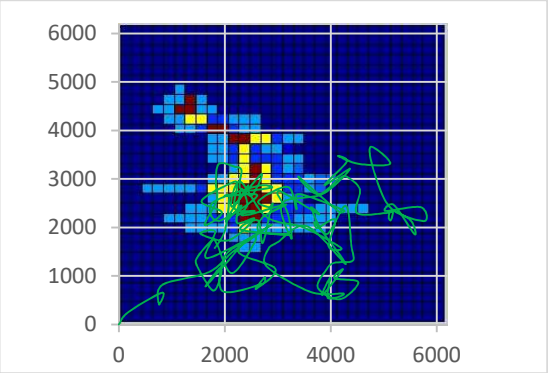
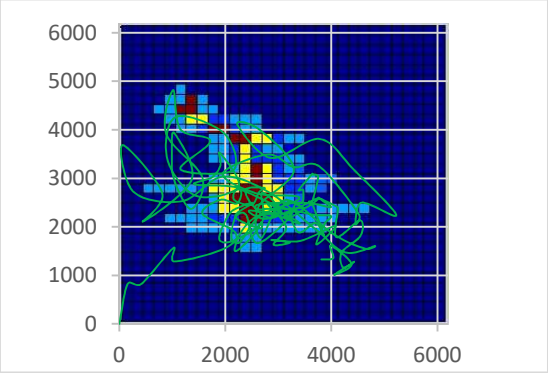
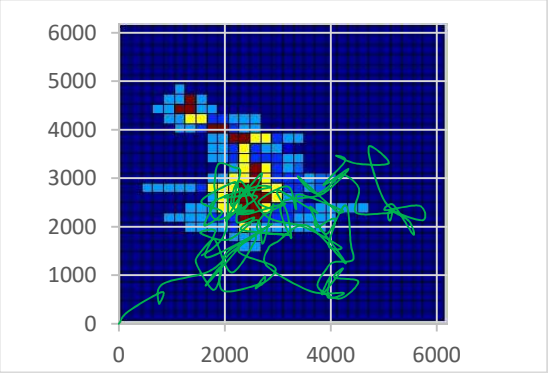
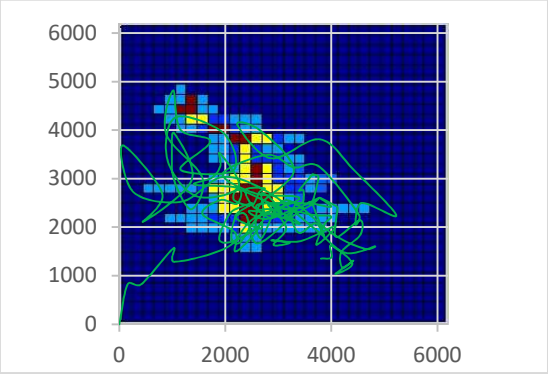
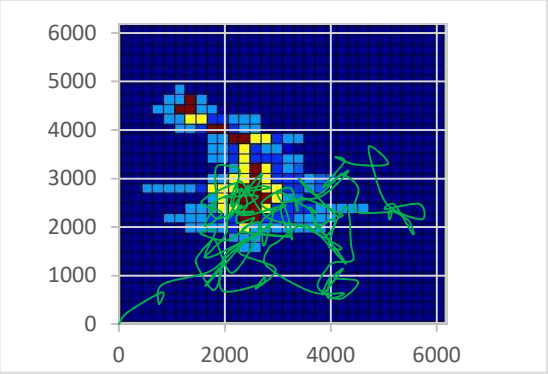
	
Iteración 100	**Iteración 100**
	
Iteración 200	**Iteración 200**
	
Iteración 300	**Iteración 300**

	
Iteración 400	**Iteración 400**
	
Iteración 500	**Iteración 500**
	
Iteración 600	**Iteración 600**

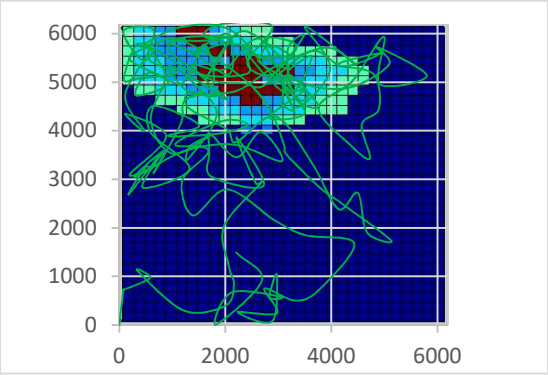
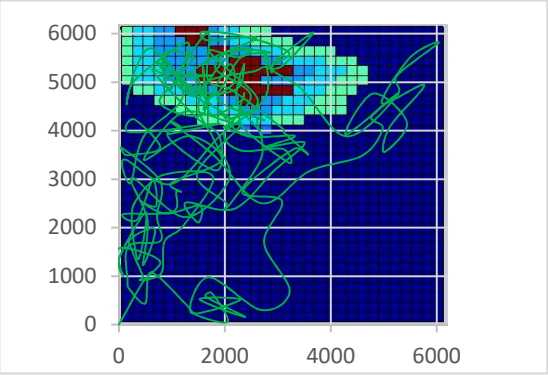
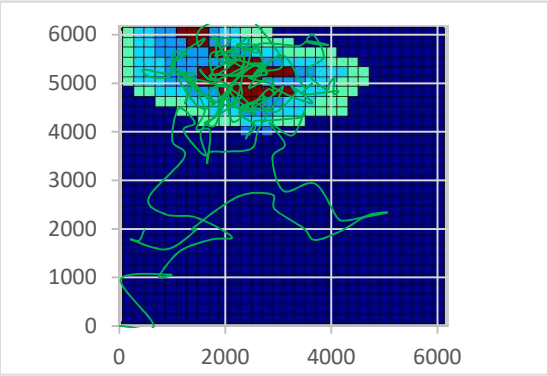
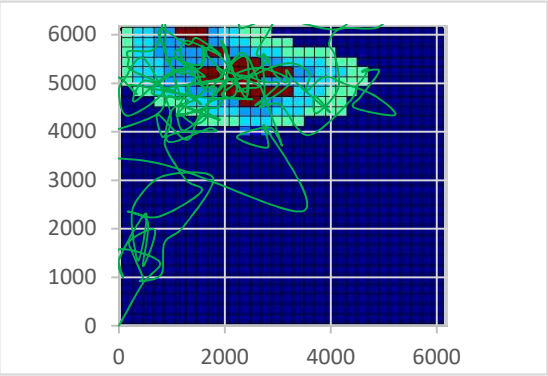
Muestra de representación evolutiva (iteraciones) de parametrizaciones óptimas para el escenario 2

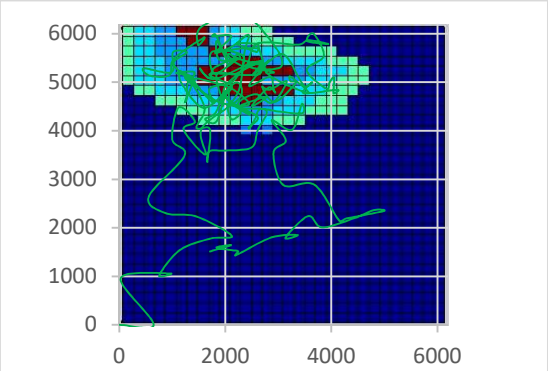
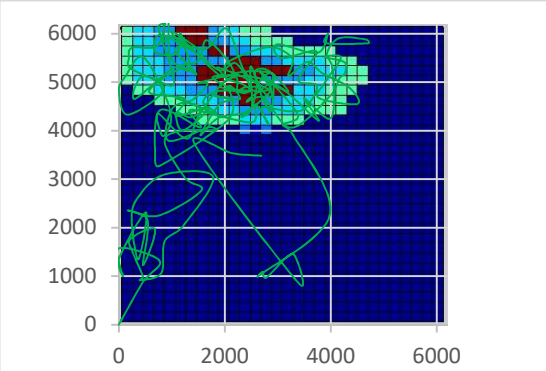
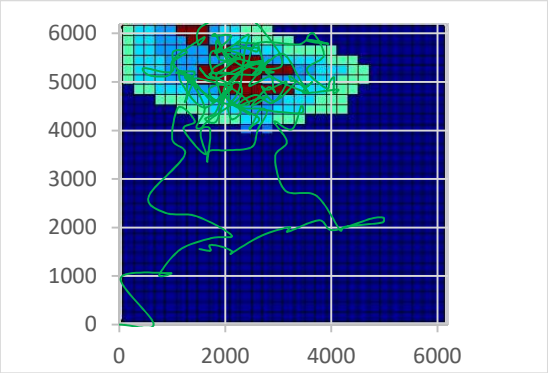
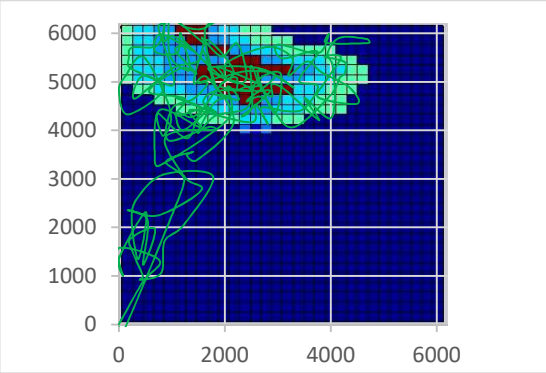
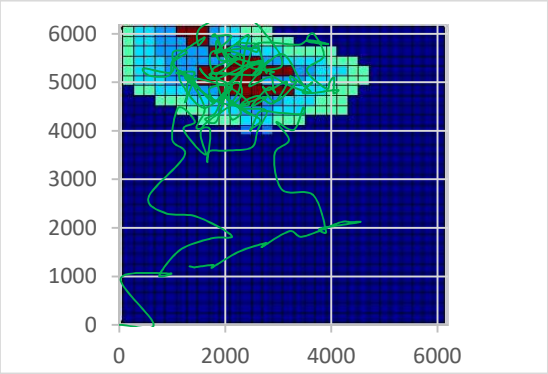
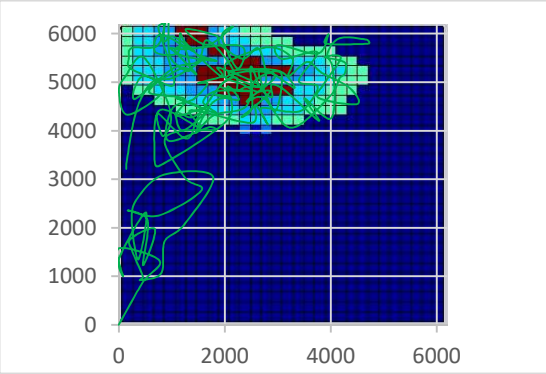
Parametrización **Opert=2, Obase=1, Diferencial=5**	**Parametrización** **Opert=1, Obase=1, Diferencial=5**
	
Iteración 1	**Iteración 1**
	
Iteración 50	**Iteración 50**

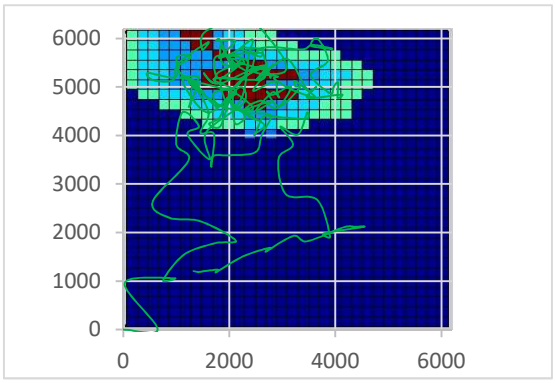
	
Iteración 100	**Iteración 100**
	
Iteración 200	**Iteración 200**
	
Iteración 300	**Iteración 300**

	
Iteración 400	**Iteración 400**
	
Iteración 500	**Iteración 500**
	
Iteración 600	**Iteración 600**

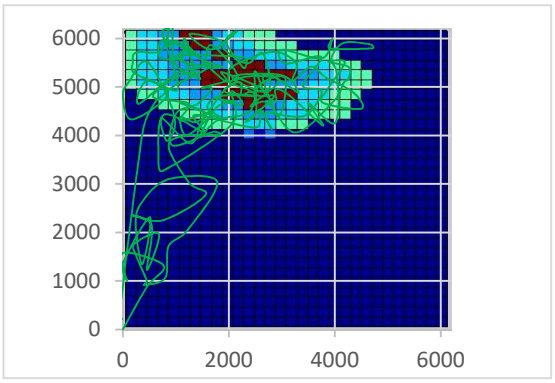
Muestra de representación evolutiva (iteraciones) de parametrizaciones óptimas para el escenario 3

Parametrización **Opert=2, Obase=1, Diferencial=5**	**Parametrización** **Opert=3, Obase=1, Diferencial=5**
 **Iteración 1**	 **Iteración 1**
 **Iteración 50**	 **Iteración 50**

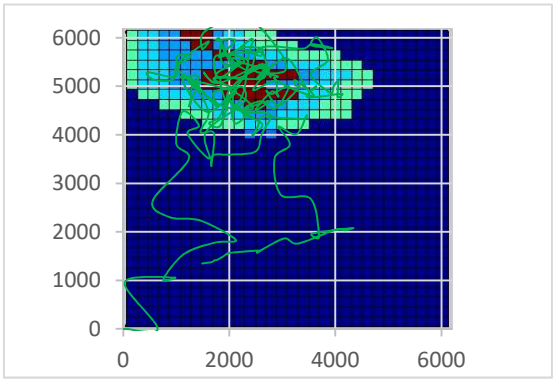
	
Iteración 100	**Iteración 100**
	
Iteración 200	**Iteración 200**
	
Iteración 300	**Iteración 300**



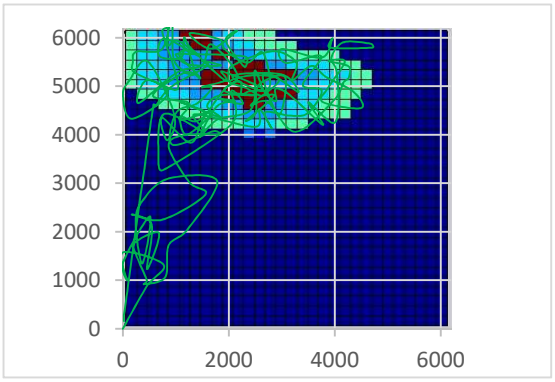
Iteración 400



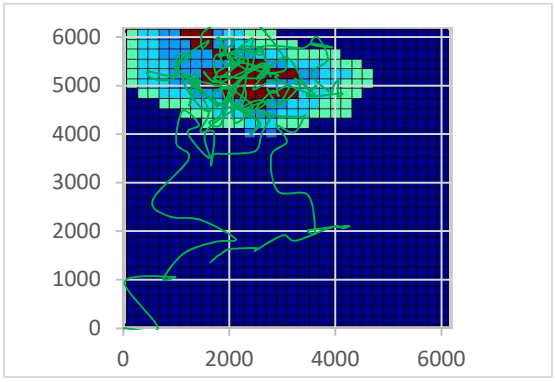
Iteración 400



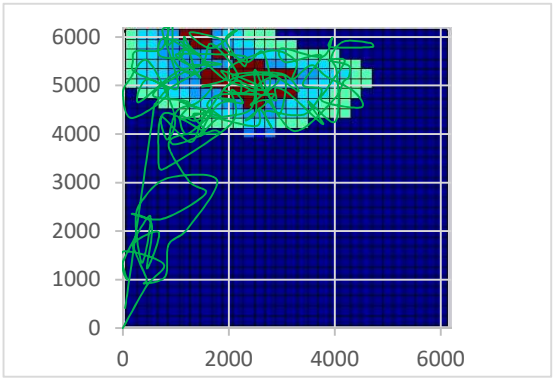
Iteración 500



Iteración 500



Iteración 600



Iteración 600