UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE INFORMÁTICA
Departamento de Arquitectura de Computadores y Automática

Máster en Ingeniería de Sistemas y Control

Proyecto Final de Máster

Optimización de trayectorias de UAVs para la rápida localización de objetivos mediante algoritmos de colonias de hormigas

ALUMNO**: FRANCISCO JAVIER HUERTOS IZQUIERDO**

DIRECTORES: **EVA BESADA PORTAS y SARA PEREZ CARABAZA**

Curso académico 2015/2016

Convocatoria de defensa: Septiembre

Máster en Ingeniería de Sistemas y Control

Optimización de trayectorias de UAVs para la rápida localización de objetivos mediante algoritmos de colonias de hormigas

ALUMNO**: FRANCISCO JAVIER HUERTOS IZQUIERDO**

DIRECTORES: **EVA BESADA PORTAS y SARA PEREZ CARABAZA**

Curso académico 2015/2016

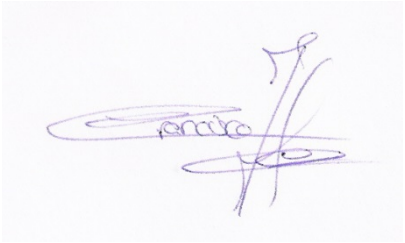
Convocatoria de defensa: Septiembre

Autorización

Autorizamos a la Universidad Complutense de Madrid (UCM) y a la Universidad Nacional de Educación a Distancia (UNED) 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:

Firma del alumno



Resumen

En este trabajo se presenta una adaptación del algoritmo Ant Colony Extended (ACE) para la resolución del problema de búsqueda en tiempo mínimo (MTS). En concreto, se diseña un planificador para múltiples UAVs con el fin de encontrar un objetivo de ubicación desconocida en mínimo tiempo. Este planificador debe determinar la secuencia de acciones que permite a los agentes desplazarse por la región de búsqueda mientras realizan observaciones. En este sentido, a partir del algoritmo ACE, se han desarrollado dos tipos de planificadores diferentes, uno para el problema MTS discreto con 360 acciones y otro para el problema MTS con acciones continuas. Se ha conseguido diseñar un algoritmo que trabaja con acciones continuas manteniendo la arquitectura de ACE original.

Ambos algoritmos incluyen una heurística de rumbo, desarrollada también en este trabajo, que les permite trazar rutas hacia las zonas de máxima probabilidad de detección del objetivo mientras esquivan zonas prohibidas de vuelo. La eficacia de ambos algoritmos ha sido comprobada frente a varios escenarios. Los resultados obtenidos para ambos algoritmos son muy similares en todos los escenarios probados. La versión de ACE con acciones continuas también ha sido comparada con otros algoritmos continuos, sobre diferentes escenarios. El estudio realizado muestra que ACE tiene una mayor rapidez de convergencia y presenta soluciones de mayor calidad que los otros dos métodos.

Palabras clave: búsqueda en tiempo mínimo, optimización por colonias de hormigas, planificación multi-UAV,

Índice General

1 Introducción

Los últimos años los vehículos aéreos no tripulados (UAVs en sus siglas en inglés) se han popularizado convirtiéndose en pieza muy útil en numerosas aplicaciones, tanto militares como civiles. Entre otras destaca su uso en tareas como inspección, exploración, mantenimiento, detección de objetivos, etc. favorecido por la capacidad de este tipo de vehículos para acceder con mínimo esfuerzo a áreas inaccesibles por otros medios.

A pesar de todo, hoy en día muchos de estos vehículos aéreos son todavía controlados de forma remota, lo cual restringe las posibles aplicaciones a áreas libres de obstáculos donde el vehículo se mantiene en todo momento visible al operador. El principal reto en el desarrollo de UAVs inteligentes sigue siendo la planificación de caminos en entornos adversos: vuelos externos a baja altitud en entornos con vegetación o entre montañas, o en zonas de vuelo amplias donde la conexión directa con el UAV podría verse comprometida. Este tipo de tareas requieren de un mayor nivel de autonomía por parte de los vehículos.

La planificación de caminos ha sido una de las principales ramas de investigación de la robótica móvil de las últimas décadas [20]. Ante la gran cantidad de posibles caminos que se le presentan al UAV, la selección del mejor camino cumpliendo con todos las limitaciones es un factor crítico. La planificación de caminos puede ser descrita como el procedimiento por el que el UAV encuentra un camino libre de colisiones desde una posición específica a otra de destino de acuerdo a ciertos criterios (función objetivo), tales como mínima distancia, máxima seguridad o mínimo tiempo de búsqueda. Este último factor resulta especialmente interesante en misiones donde el objetivo debe ser localizado en el menor tiempo posible.

A lo largo de estas décadas son numerosos los métodos presentados para la resolución de problemas relativos a la planificación de caminos. En casos sencillos, los métodos clásicos (mapa de carreteras, descomposición en celdas, campo de potencial, etc.) pueden resultar efectivos. Sin embargo, en casos más complejos presentan problemas de estancamiento en mínimos locales, evitando que el robot alcance el objetivo [15]. Más aún, la planificación de caminos de un robot móvil en presencia de obstáculos es considerada un problema NP-duro [4]. Es decir, se considera que no es posible encontrar un algoritmo eficiente (con tiempo polinomial) que pueda resolverlo de forma óptima. El problema resulta incluso más complejo en presencia de obstáculos dinámicos, lo que prácticamente excluye a los métodos clásicos [29].

Normalmente este tipo de problemas son resueltos con métodos aproximados o heurísticos, que permiten encontrar soluciones aproximadas (i.e., soluciones buenas, pero probablemente no óptimas) en unos tiempos de cómputo razonables. El desarrollo de estos métodos ha recibido muchísima atención en los últimos años; entre otras destacan técnicas tales como algoritmos genéticos (Genetic Algorithms GA), optimización por enjambre de partículas (Particle Swarm Optimization, PSO), evolución diferencial (Differential Evolution, DE), optimización por colonias de hormigas (Ant Colony Optimization, ACO), temple simulado (Simulated Annealing, SA) y búsqueda tabú (Tabu Search, TS). Este tipo de técnicas se presenta como la solución a los grandes costes computacionales y complejidades de los métodos clásicos, particularmente en entornos complejos.

En concreto, en este Trabajo de Fin de Máster se plantea el diseño de un planificador para múltiples UAVs con el fin de encontrar un objetivo de ubicación desconocida en mínimo tiempo o, en otras palabras, estudia y plantea una solución de la Búsqueda de Tiempo Mínimo (Minimum Time Search, MTS, [27], [16]) con múltiples agentes. Este planificador deberá obtener la secuencia de acciones que permita a los agentes desplazarse por la región de búsqueda mientras realizan observaciones que permitan la detección del objetivo en un tiempo mínimo. La Figura 1 muestra una representación esquemática del problema.

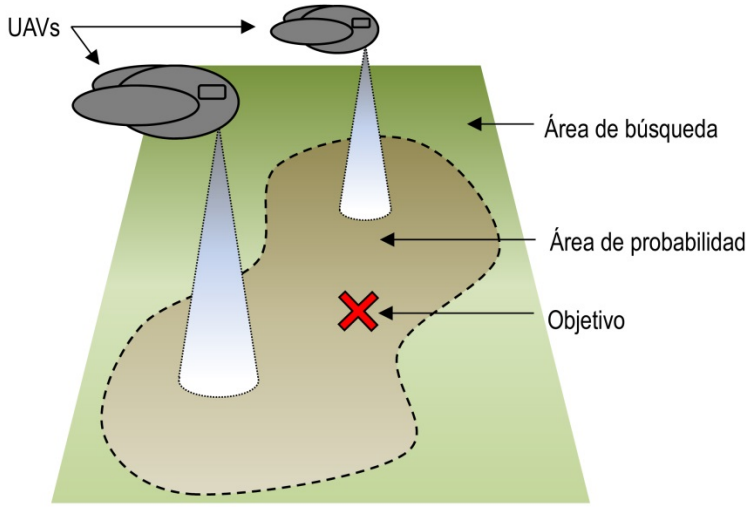


Figura 1: Esquema representativo del problema de búsqueda en MTS

Un aspecto que dificulta la resolución del problema MTS es la presencia de incertidumbre en la información disponible. Por lo tanto, el planificador que se debe desarrollar no está pensado para entornos deterministas, sino que debe ser capaz de manejar las fuentes de incertidumbre del problema, modeladas a través de modelos probabilísticos relacionados con la localización inicial del objetivo, el desplazamiento del mismo y los errores existentes en la información sensorial. Tal y como se ha mencionado anteriormente, la gran complejidad de este problema obliga a la utilización de métodos de optimización heurísticos o aproximados.

En este sentido, a lo largo de los últimos años se han presentado diversos planificadores que pueden ser englobados en dos familias diferentes. Por un lado, un conjunto de planificadores que aborda el problema MTS desde un punto de vista discreto, limitando las acciones de los UAVs a las 8 direcciones relacionadas con los puntos cardinales([24], [30], [16], [17]). Esto permite la obtención de planes de alto nivel no directamente realizables por los UAVs. Existe también una segunda familia, de creación más reciente, que proponen trayectorias factibles para los UAVs, a partir de valores continuos en las acciones realizables ([25], [18]).

Este trabajo pretende extender los prometedores resultados obtenidos mediante el algoritmo ACO discreto [30] a problemas MTS de optimización con acciones continuas. Para ello, se ha utilizado el algoritmo Ant Colony Extended (ACE), desarrollado por Escario [11]. ACE busca completar la meta-heurística ACO mediante la inclusión de una representación en espacio de estados y de elementos de las colonias de abejas en la organización de una colonia de hormigas. Esta nueva técnica de optimización ha sido aplicada satisfactoriamente en la resolución de problemas discretos como la planificación de maniobras para barcos [9] o la resolución del problema TSP [10]. En este trabajo se

ha adaptado ACE para la resolución de problemas MTS y se ha desarrollado una nueva heurística basada en campos de potencial que permite al UAV trazar rutas libres de colisiones.

Desde la aparición de ACO como una herramienta de optimización combinatorial, se han realizado distintos intentos para utilizarlo en problemas continuos. Una de las últimas adaptaciones es el algoritmo ACO_R propuesto por Socha [26]. Siguiendo esta línea, en este trabajo también se plantea una adaptación de ACE para la optimización de problemas continuos en base a las modificaciones propuestas por Socha dentro de su algoritmo de hormigas.

1.1 Motivación

Las técnicas de optimización aparecen en multitud de ámbitos de la vida real con objetivos tan diversos como minimizar el coste y consumo de energía, maximizar la seguridad o eficiencia o incluso minimizar el tiempo de búsqueda de un objetivo. Se han convertido por tanto en una herramienta básica debido a las limitaciones existentes en la mayoría de los procesos en lo que se refiere a tiempo, diseño, dinero, recursos, etc.

Son incontables los casos reales en los que se deben encontrar uno o varios objetivos lo más rápidamente posible, ya sea debido a accidentes, rescates, etc. En estos casos, resulta imprescindible obtener el mejor conjunto de acciones posible que determinen la trayectoria a seguir, permitiendo a los agentes desplazarse por diferentes áreas geográficas para encontrar el objetivo en el tiempo mínimo posible.

La incertidumbre existente en la información relativa al problema que nos atañe añade un plus de complejidad a la búsqueda de soluciones óptimas. La mayoría de estos problemas se consideran NP-duros, por lo que la búsqueda de soluciones óptimas requiere de un tiempo exponencial. Es por ello que en este tipo de problemas se habla de solución aproximada o sub-óptima.

A lo largo de las últimas décadas se han presentado multitud de técnicas de optimización para tratar de solucionar los problemas de NP-duros. Entre otras, destacan la familia de las heurísticas, dentro de la cual se encuentra el algoritmo ACO. Este algoritmo ha resultado ser muy útil en la búsqueda de caminos hacia el objetivo. En este sentido, recientes publicaciones lo presentan como un algoritmo eficiente en la resolución del problema MTS con 8 acciones cardinales [24]. Los problemas MTS requieren de la identificación de un camino que posibilite la detección de un objetivo en el menor tiempo posible. Sin embargo, la combinación de incertidumbre en relación al objetivo y el factor tiempo dificulta la resolución del mismo.

Por su parte, el algoritmo Ant Colony Extended (ACE), extensión del algoritmo ACO, también ha sido aplicado satisfactoriamente en la resolución de problemas discretos [9], en este caso con 360 acciones discretas diferentes. Una de las principales ventajas de ACE con respecto a ACO, es que el primero es un algoritmo que trabaja en grandes espacios de estados discretos, lo cual facilita su posible aplicación a problemas con acciones continuas.

Guiados por los buenos resultados obtenidos por los algoritmos basados en colonias de hormigas, este trabajo pretende, en primer lugar, adaptar el algoritmo ACE para la optimización de problemas MTS para una configuración de 360 acciones discretas. Para a continuación plantear una adaptación

de ACE al problema MTS continuo, realizada para permitir la optimización de las trayectorias del UAV codificadas a partir de una secuencia de acciones que toma valores continuos.

Finalmente, también analizaremos los resultados obtenidos con los nuevos algoritmos y los compararemos con otras técnicas como los algoritmos genéticos y los métodos de optimización de estimación de distribuciones.

1.2 Objetivos

Partiendo del conocimiento adquirido por trabajos presentados en el Departamento de Arquitectura de Computadores y Automática de la Universidad Complutense de Madrid en relación a algoritmos basados en colonias de hormigas, el principal objetivo de este TFM es adaptar el algoritmo Ant Colony Extended (ACE) [11] para la resolución de problemas MTS de optimización con acciones continuas. La búsqueda en tiempo mínimo (MTS) consiste en localizar un objetivo (elemento a encontrar) en el menor tiempo posible.

En este trabajo se considera un objetivo estático, cuya posición no varía con el tiempo. Sin embargo, dicho objetivo tiene una localización desconocida (estado) dentro de una zona limitada, de la cual se conocen las zonas de máxima probabilidad de localización. Los agentes (UAVs en este caso) se desplazan sobre la zona de búsqueda siguiendo las acciones de control determinadas por la estrategia de búsqueda, mientras realizan una serie de observaciones de la zona de búsqueda con ayuda de los sensores que incorporan.

El objetivo de la estrategia es determinar el mejor conjunto de N acciones para que, probabilísticamente, encuentren el objetivo en el menor tiempo posible; todo ello, teniendo en cuenta la incertidumbre asociada a la localización del objetivo e información sensorial.

Además, se debe definir una heurística de rumbo para ACE para los UAVs, que trate la naturaleza probabilística del problema MTS.

Finalmente, se analizará el funcionamiento del nuevo algoritmo de hormigas y se comparará los resultados obtenidos por éste con otras técnicas heurísticas.

1.3 Organización del Trabajo

El resto de esta memoria se divide en 5 bloques, cada uno recogido en uno de los capítulos que la conforman.

El capítulo 2 describe los elementos principales del problema MTS de detección de objetivos. En los primeros apartados, se definen los modelos matemáticos necesarios para englobar la información relativa al problema, así como la función objetivo a minimizar. Posteriormente, en los dos últimos apartados, se realiza un análisis del problema MTS desde un planteamiento con acciones discretas y acciones continuas.

En el capítulo 3 se realiza una pequeña descripción del algoritmo ACO, funcionamiento e implementación, que sirve de introducción del algoritmo ACE utilizado en este trabajo. A continuación, se detallan las principales características del algoritmo ACE. El apartado comienza describiendo las diferencias entre ACO y ACE, para luego mostrar un breve resumen de la implementación de este último en la resolución de problemas discretos.

El capítulo 4 muestra el proceso seguido durante este trabajo para resolver el problema MTS mediante ACE. El primer apartado describe el uso de ACE sobre un MTS discreto con 8 acciones cardinales. El siguiente paso consiste en extender el problema a 360 acciones discretas diferentes, para lo cual se ha desarrollado una nueva heurística que permite tratar la naturaleza probabilística del problema MTS. Por último, el apartado final plantea una versión de ACE sobre acciones continuas. Para ilustrar el funcionamiento de cada versión de ACE, el capítulo también recoge los resultados obtenidos por cada versión del algoritmo ACE desarrollado sobre algunos escenarios de búsqueda.

El Capítulo 5 muestra los estudios experimentales realizados con las dos versiones de ACE desarrolladas: discreto con 360 acciones diferentes y con acciones continuas. Inicialmente, se realiza una comparativa entre ambos planteamientos sobre diferentes escenarios de búsqueda. A continuación, se compara la versión de ACE sobre acciones continuas con otros dos algoritmos continuos como son un algoritmo genético (GA) y un algoritmo de optimización de estimación de distribuciones cuyo soporte es una mezcla de gaussianas, sobre diferentes escenarios.

Finalmente, en el último capítulo, se presentan las conclusiones de este Trabajo Fin de Máster y se proponen algunas líneas de trabajo futuro.

2 Búsqueda en Tiempo Mínimo

En este capítulo se describen los elementos principales del problema MTS de detección de objetivos, en el cual participan dos elementos bien diferenciados: los agentes móviles con capacidades sensoriales y el objetivo, de localización desconocida, que se pretende encontrar en el menor tiempo posible. En este caso, los agentes son UAVs que realizan observaciones sobre la zona de búsqueda mientras se desplazan por la misma.

2.1 Búsqueda de Tiempo Mínimo

La búsqueda de tiempo mínimo (MTS) consiste en encontrar, en el menor tiempo posible, un objeto que está posicionado en algún lugar desconocido del espacio de búsqueda. El principal reto para solucionar el problema MTS reside en la dinámica de los agentes y en la incertidumbre asociada a la localización y dinámica del objetivo y a los errores que se producen en las observaciones de los sensores.

Una forma de solucionar el problema MTS es mediante la minimización del tiempo esperado [27]. Definiendo el tiempo para encontrar el objetivo como una variable aleatoria T, la solución al problema del MTS consiste en minimizar el tiempo esperado para encontrar el objetivo antes del instante k:

$$min(E\{T \leq k\}) \tag{1}$$

En concreto, a partir de la información disponible, los agentes deben tomar las decisiones óptimas para encontrar el objetivo dentro de un tiempo con tendencia al mínimo. El problema de optimización para un equipo de M agentes es encontrar la mejor secuencia de N acciones $u_{1:M}^{0:N-1}$ que cumplan:

$$arg\ \min_{u_{1:M}^{0:N-1}}(E\{T \leq k\}) \tag{2}$$

En nuestro escenario particular, disponemos de agentes móviles con capacidades sensoriales (UAVs) y el objetivo a detectar, todos ellos posicionados en un espacio de búsqueda delimitado. La ubicación del objetivo no es conocida, pero los agentes disponen de información probabilística inicial de la posición del objetivo. A partir de esta información, los agentes se desplazan por el espacio de búsqueda de acuerdo con su dinámica y siguiendo las acciones de control obtenidas por una estrategia de búsqueda, realizando observaciones para detectar el objetivo.

El objetivo de la estrategia de búsqueda es determinar el mejor conjunto de acciones que permita a las agentes detectar el objetivo lo antes posible, teniendo en cuenta la incertidumbre asociada a la localización y dinámica del objetivo, y a la información sensorial.

2.2 Localización del Objetivo

Definimos el espacio de búsqueda delimitado de la misión $\Omega \subseteq \mathbb{R}^2$, donde se encuentran ubicados los agentes y el objetivo. Los agentes parten de una posición conocida y disponen de información probabilística inicial de la posición y dinámica del objetivo, su localización concreta es desconocida.

La información sobre la localización del objetivo se define a partir de un modelo probabilístico $\mathrm{b}(\tau^0) = \mathrm{P}(\tau^0)$ y la relativa a su dinámica viene definida por el modelo probabilístico $\mathrm{P}(\tau^{\mathrm{k}}|\tau^{\mathrm{k}-1})$. El objetivo se mueve independientemente de las acciones de los agentes. Un ejemplo de $\mathrm{b}(\tau^0)$ se puede observar en la siguiente figura.

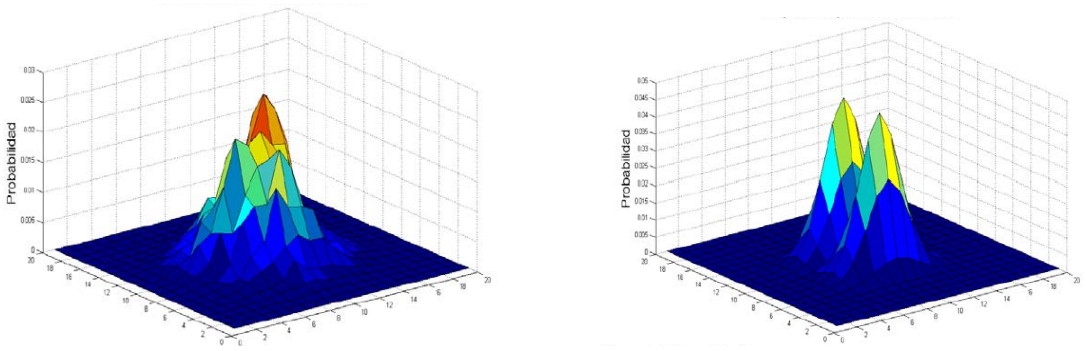


Figura 2: Mapas probabilísticos a) con una única concentración de probabilidad o b) dos concentraciones de probabilidad

Mediante ambos modelos probabilísticos los UAVs pueden determinar las áreas más prometedoras de detección del objetivo y así con la ayuda del algoritmo de optimización determinar sus mejores trayectorias. A su vez, la aplicación de estimadores bayesianos permite ir actualizando el modelo de posición inicial $\mathrm{b}(\tau^0)$, con la información obtenida tras nuevas observaciones de los sensores y el modelo dinámico del objetivo.

2.3 Modelo del sensor

Cada agente dispone de un sensor que le permite realizar observaciones según se desplaza. En cada instante planificado k, el equipo de agentes (M UAVs) realiza observaciones $z_{1:M}^k$ sobre la zona de búsqueda, lo que permite actualizar el mapa de probabilidad hasta el instante k con las medidas realizadas hasta ese instante $\mathrm{b}(\tau^{\mathrm{k}}) = \mathrm{P}(\tau^{\mathrm{k}}|\mathrm{z}_{1:\mathrm{M}}^{1:\mathrm{k}}, \mathrm{s}_{1:\mathrm{M}}^{1:\mathrm{k}})$, a lo largo de sus trayectorias $\mathrm{s}_{1:\mathrm{M}}^{\mathrm{k}}$. De esta manera los agentes pueden actualizar la información que disponen en relación a la localización del objetivo. La actualización se lleva a cabo con la fusión de la información percibida por los sensores y la existente en el mapa. El modelo del sensor viene definido por la siguiente expresión [18]:

$$P\left(z_i^k = D|\tau_i^k, s_i^k\right) \tag{3}$$

Esta posibilidad de observación es la probabilidad de detectar un objetivo cuando la posición del mismo es τ_i^k y la del sensor es s^k. Se considera que la posición del sensor coincide con la del UAV.

A pesar de que la implementación del algoritmo permite el uso de cualquier tipo de sensor, por simplificación, el modelo de sensor utilizado en este trabajo viene definido por un modelo de sensor binario ideal (Figura 3), el cual devuelve 1 cuando el sensor puede detectar al objetivo, porque el objetivo y el sensor se encuentran sobre la misma celda, y 0 en el caso contrario.

$$P\left(z_i^k = D|\tau_i^k, s_i^k\right) = \begin{cases} 1 & i^k = s_i^k \\ 0 & i^k \neq s_i^k \end{cases} \tag{4}$$

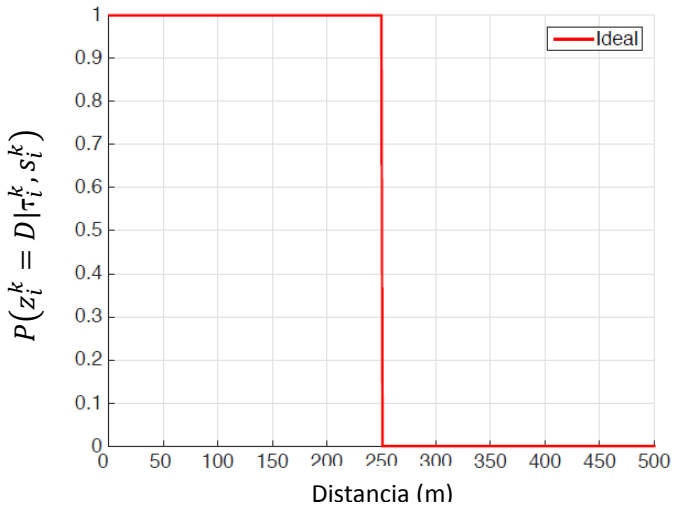


Figura 3: Representación del sensor ideal [23] en la zona de búsqueda

2.4 Función Objetivo

La solución al MTS se ha diseñado como un proceso de decisión, donde cada vez que los agentes realizan un desplazamiento, también realizan una nueva observación que modifica el modelo probabilístico del objetivo $\mathrm{b}(\tau^k)$, es decir la creencia de la posición del objetivo. De ello se deduce que el orden de las acciones realizadas por los agentes es importante. El tiempo esperado sobre N pasos sucesivos de búsqueda (ET) se puede calcular en función de los modelos $\mathrm{b}(\tau^0)$, $\mathrm{P}(\tau^{\mathrm{k}}|\tau^{\mathrm{k-1}})$ y $P(z_i^k = D|\tau_i^k, s_i^k)$ mediante el siguiente conjunto de fórmulas [23]:

$$\mathrm{ET}(s_{1:M}^{0:N}) = ET(s_{1:M}^{0:N-1}) + \sum_{\tau^N=1}^{w_x * w_y} \tilde{b}(\tau^N, s_{1:M}^{0:N}) \tag{5}$$

$$\tilde{b}(\tau^N, s_{1:M}^{0:N}) = \prod_{i=1}^{M} P(z_i^N = \overline{D}|\tau^N, s_i^N) \sum_{\tau^{N-1}=1}^{w_x * w_y} P(\tau^N|\tau^{N-1}) \tilde{b}(\tau^{N-1}, s_{1:M}^{0:N-1}) \tag{6}$$

$$\mathrm{ET}(s_{1:M}^{0}) = \sum_{\tau^0=1}^{w_x * w_y} b(\tau^0) = 1 \tag{7}$$

$$\tilde{b}(\tau^0, s_{1:M}^{0}) = b(\tau^0) \tag{8}$$

De la formulación empleada se desprende que la evaluación de ET se calcula mediante la suma de probabilidades de no detección $P(z_i^k = D|\tau_i^k, s_i^k)$ desde la posición inicial hasta el horizonte de decisión N, en lugar de hasta el infinito por lo que el valor ET se ve truncado.

Tras introducir la formulación matemática del problema en las siguientes secciones presentamos las dos variantes habituales y describimos brevemente las heurísticas utilizadas para resolverlos.

2.5 Problema MTS con acciones discretas

El objetivo del proceso de optimización del problema MTS consiste en obtener la mejor secuencia de acciones de control (que inducirán una trayectoria sobre el UAV), de acuerdo a la función de

optimización escogida (MTS) y teniendo en cuenta las observaciones realizadas por los UAVs ($i \in \{1: M\}$) en los diferentes instantes k. Para ello, disponemos de la posición inicial de los vehículos, y la información relativa a la localización y dinámica del objetivo.

En el problema MTS con acciones discretas, el espacio de búsqueda Ω es discretizado mediante una malla G de $\omega_x * \omega_y$ celdas. De esta manera cada celda del mapa lleva asignada la probabilidad de que el objetivo se encuentre en ella. En la Figura 4 se muestra un ejemplo de un mapa de creencia de 20x20 celdas. El mapa viene coloreado en altura, donde los colores más calientes (altos) representan las zonas con una mayor probabilidad de encontrar el objetivo. Al contrario los colores más fríos representan una baja probabilidades de dar con el objetivo en esas áreas.

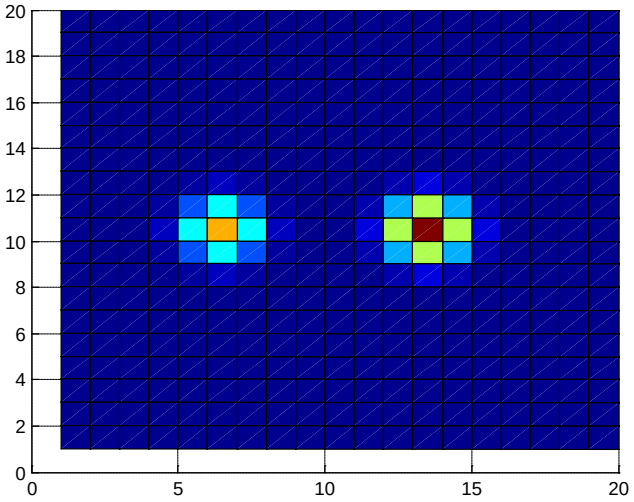


Figura 4: Mapa de creencia $b(\tau^0)$

Esta discretización permite identificar cada celda como un punto discreto conectado con los 8 puntos (celdas) adyacentes. La conexión entre celdas contiguas, definida por las 8 direcciones cardinales (N, NE, E, SE, S, SW, W, NW), hace que un UAV se desplace de celda a celda por el espacio de búsqueda. Es decir, la secuencia de direcciones cardinales definen el conjunto de acciones necesario para guiar al vehículo. A modo de ejemplo en la Figura 5 [18] se ha representado un escenario discretizado ($\omega_x * \omega_y$) y la trayectoria de un agente cuando sigue una serie de acciones cardinales.

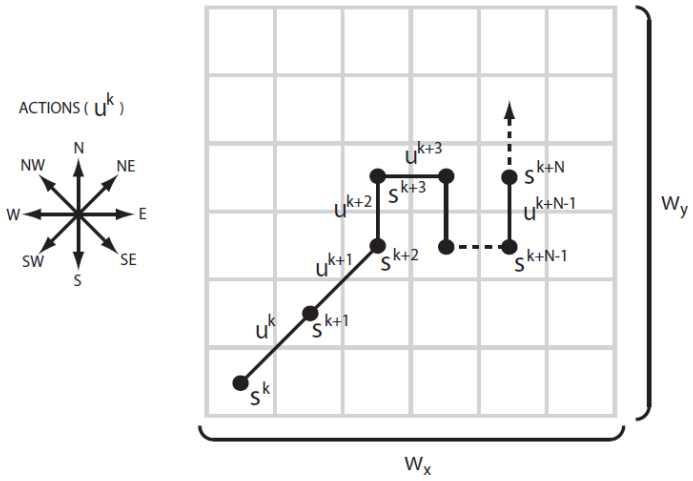


Figura 5: Escenario discreto donde cada punto (celda) está conectado a los puntos adyacentes.

La posición s_i^{k+1} de cada agente i está relacionada con la posición anterior del agente s_i^k por medio de la acción de control u_i^k. Por tanto, la secuencia de N acciones de control $v_i^k = \{u_i^k, \ldots, u_i^{k+N-1}\}$, hace que el agente i se desplace del estado s_i^k a $\{s_i^{k+1}, \ldots, s_i^{k+N}\}$ según avanza el tiempo.

De esta manera se consigue transformar el problema MTS en un problema de optimización combinatorial (CO). Los problemas CO, $P = (S, f)$ son un problema de optimización en el que se dan una serie finita de estados en S (también llamado espacio de búsqueda) y una función objetivo $f: S \rightarrow \mathbb{R}^+$ (en nuestro caso el $ET(s_{1:M}^{0:k})$) que asigna un valor de coste positivo a cada uno de los estados $s \in S$ (en nuestro caso la secuencia de posiciones o de acciones de los UAVs). El objetivo es encontrar un valor de coste mínimo (o máximo, dependiendo del tipo de problema).

Esta variante del problema MTS, que permite determinar trayectorias de alto nivel para diferentes tipos de agentes, ha sido tratada en los últimos años con diferentes algoritmos heurísticos. Entre ellos destacan la optimización por entropía cruzada (CEO, [16]), el algoritmo de optimización bayesiana (BOA, [17]), diferentes algoritmos genéticos (GA, [30]), y más recientemente, un algoritmo de hormigas clásico (ACO, [24]). Una comparativa reciente de los mismos [23] muestra a ACO como la solución más eficiente computacionalmente en esta variante del problema, seguido por una variante de GA.

La eficiencia computacional de cada técnica sobre el problema MTS debe estar ligada al funcionamiento intrínseco de cada algoritmo. Los dos primeros (CEO y BOA) son algoritmos que buscan la solución óptima del problema a través del aprendizaje y muestreo del modelo probabilístico que define las zonas donde se concentran las mejores soluciones obtenidas en cada iteración, el tercero (GA) es una heurística que modifica soluciones de iteraciones previas mediante cruces y mutaciones, y el más reciente (ACO) es un algoritmo que combina el aprendizaje de un modelo probabilístico con el muestreo de soluciones de dicho modelo y de una heurística propia del problema. Es decir, tres algoritmos (CEO, BOA, ACO) aprenden un modelo probabilístico, uno modifica soluciones existentes con operaciones básicas (GA) y uno (ACO) utiliza una heurística especialmente diseñada para el problema MTS para acelerar la búsqueda. Por lo tanto, el uso combinado de la heurística con el aprendizaje del modelo probabilístico parece ser el motivo fundamental para la mejor eficiencia computacional de ACO, mientras que la sencillez de las operaciones de GA es preferible al aprendizaje de modelos probabilísticos sin ninguna guía adicional de CEO, BOA.

2.6 Problema MTS con acciones continuas

Sin embargo, en la variante del problema en el que las acciones toman valores continuos (i.e. planificación de movimientos de un UAV real) no es posible establecer la conexión entre los puntos/celdas que se usa en la formulación con acciones discretas, ya que ésta viene impuesta por la dinámica del modelo continuo del UAV a estudiar. Más aún, a priori se desconoce el conjunto de celdas alcanzable ya que este depende del estado actual y la dinámica del modelo. La dinámica específica de los vehículos aéreos impone las correspondientes restricciones a su maniobrabilidad.

Por lo tanto en el caso continuo, el estado s_i^{k+1} de cada UAV viene determinado por el estado del UAV en el instante anterior s_i^k, la acción de control u_i^k y el entorno ϵ, integrados en un modelo cinemático diferencial $\dot{s}_i = f(s_i^{k-1}, u_i^{k-1}, \epsilon)$. De esta forma, resulta posible calcular los estados $s_{1:M}^{1:N}$

de los UAVs durante las siguientes N observaciones $z_{1:M}^{1:N}$, a partir de los estados iniciales de UAVs $s_{1:M}^{0}$ y la secuencia de señales de control $u_{1:M}^{0:N-1}$.

Para resolver el problema en esta variante con acciones continuas, aproximamos la optimización global trasformando el problema en una optimización lineal a tramos, donde la misma acción de control se utiliza durante un intervalo constante de tiempo. Por lo tanto, cada agente dispone de un vector de acciones compuesto de N trozos de acciones lineales $\mathrm{v_i^k} = \{\mathrm{u_i^k}, \ldots, \mathrm{u_i^{k+N-1}}\}$, que hacen que éste se desplace del estado $\mathrm{s_i^k}$ a $\{\mathrm{s_i^{k+1}}, \ldots, \mathrm{s_i^{k+N}}\}$ según avanza el tiempo (Figura 6).

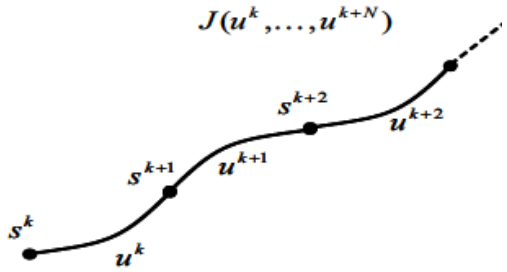


Figura 6: Representación de una trayectoria a partir de N trozos de acciones lineales $v_i^k = \{u_i^k, \ldots, u_i^{k+N-1}\}$, que hacen que el UAV se desplace del estado s_i^k a $\{s_i^{k+1}, \ldots, s_i^{k+N}\}$ [18]

En este trabajo se ha utilizado el mismo modelo no-lineal empleado en [25] para representar la función de movimiento $\dot{s}_i = f(s_i^{k-1}, u_i^{k-1}, \epsilon)$ de los UAVs. El modelo de Simulink se muestra en la Figura 7, donde las variables de estado, representadas en verde a la derecha de la figura, son la localización en 3D de los vehículos (x_i, y_i, h_i), sus velocidades $(\dot{x}_i, \dot{y}_i, \dot{h}_i)$, guiñada (θ_i), ángulo de dirección, velocidad del aire, velocidad respecto a tierra y el consumo de combustible. Por otro lado, las variables de control, representadas en azul a la izquierda de la figura, son la velocidad del aire $(\mathrm{v_i^{c,k}})$, guiñada $(\mathrm{\theta_i^{c,k}})$, y la altura $(\mathrm{h_i^{c,k}})$. La influencia del entorno viene definida por la velocidad del viento y la dirección, representados en amarillo. Los bloques en azul se emplean para modelar la dinámica en altura, los verdes el viento, naranja el combustible, gris para la velocidad el aire y los blancos para las dinámicas laterales. Además el modelo incluye las limitaciones con respecto a la velocidad del aire, altura, y guiñada.

Esta variante del problema MTS, que permite generar trayectorias realistas para los UAVs, ha sido resuelta con menos algoritmos que la variante con acciones discretas. En concreto, en [18] se usa un modelo dinámico muy sencillo del UAV y un algoritmo de optimización basado en gradiente descendente y en [25] se usa el modelo elegido para este TFM y un algoritmo genético. En este último caso, es importante destacar que aunque el modelo permite manipular la posición de los UAVs mediante tres tipos de acciones (control de guiñada, altura y velocidad), únicamente se manipula la guiñada, manteniéndose constante las otras dos variables de control.

En este trabajo seguiremos la misma aproximación (manipulación sólo del ángulo de guiñada) y estudiaremos si es posible utilizar los beneficios observados en el ACO existente sobre la variante discreta del problema sobre un algoritmo de hormigas que permita el uso de acciones de guiñada con valores continuos (o discretos con una granularidad superior a la existente en la variante en la que únicamente se consideran 8 acciones cardinales).

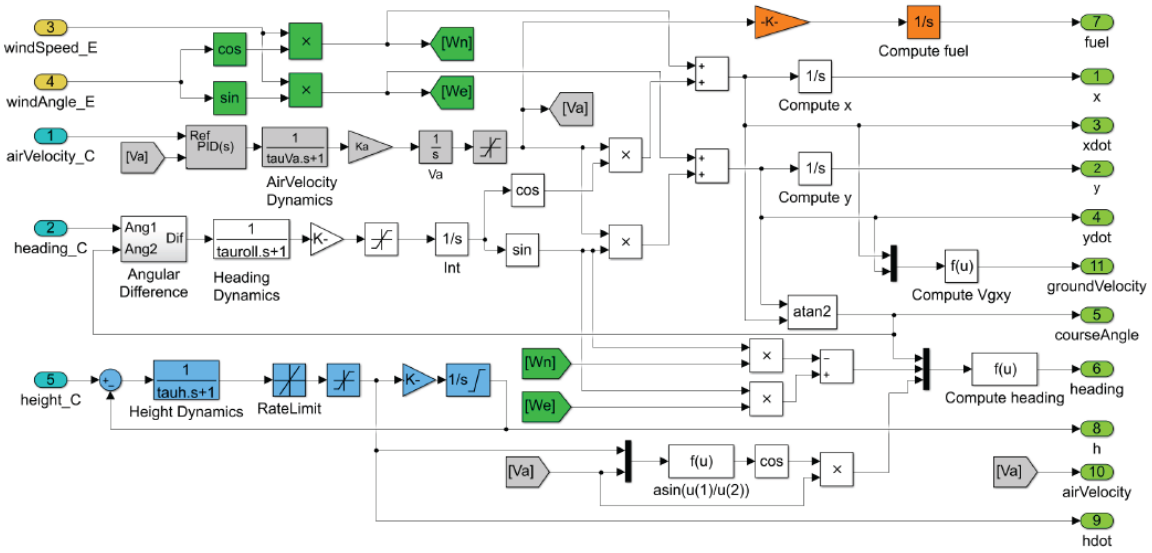


Figura 7: Modelo Simulink de un UAV

3 Algoritmos basados en colonias de hormigas

Los sistemas multi-agentes se usan frecuentemente para modelar sistemas complejos. En lugar de realizar un sistema excesivamente complicado, se diseñan partes más sencillas que se relacionan/coordinan entre sí, logrando así capacidades inteligentes. Estas capacidades no están centralizadas si no distribuidas entre las partes que lo componen.

El objetivo de las *swarm intelligence* (Inteligencia de enjambre) es precisamente el diseño de sistemas multi-agente inteligentes inspirados en el comportamiento colectivo de insectos descentralizados y auto-organizados tales como hormigas, termitas, abejas y otras sociedades de animales, como por ejemplo las *aves en vuelo.* El algoritmo ACO es una de las técnicas más exitosas de entre las técnicas denominadas inteligencia de enjambre [3]. Este grupo también lo componen otros algoritmos tales como la optimación mediante enjambre de partículas (PSO) o el algoritmo de colonia de abejas (Artificial Bee Colony, ABC).

Figura 8: Foto de Hormigas

A pesar de los buenos resultados obtenidos por el algoritmo ACO original [5], en las últimas décadas se han presentado nuevas extensiones y mejoras del mismo, cuya principal diferencia reside en el método empleado para actualizar la tabla de feromonas. Entre otros, destacan Ant Conlony System (ACS [6]), MAX-MIN Ant System (MMAS [28]). El algoritmo Ant Colony Extended (ACE) es también otra adpatación de ACO [11].

En este capítulo se realiza una pequeña descripción del algoritmo ACO, que sirve de introducción del algoritmo ACE utilizado en este trabajo. Además se detallan las diferencias entre ACO y ACE, y se muestra un breve resumen de la implementación de este último para la resolución de problemas discretos [11].

3.1 Ant Colony Optimization (ACO)

En primer lugar se realiza una breve introducción del funcionamiento del algoritmo ACO, para a continuación describir su implementación clásica [6].

3.1.1 Introducción a ACO

El algoritmo ACO, desarrollado por Dorigo [5], es un modelo inspirado en el comportamiento de colonias de hormigas reales. Su principal característica reside en imitar el trabajo cooperativo que realiza una colonia de hormigas para solucionar problemas de optimización de tipo combinatorial dentro de un tiempo razonable.

Durante el recorrido que realizan las hormigas entre el nido y la comida, éstas depositan una sustancia química denominada feromona, la cual les permite comunicarse entre ellas. Una hormiga deposita cierta cantidad de feromona para marcar el trayecto recorrido, haciéndolo de esta manera más atractivo al resto de la colonia.

Inicialmente, una hormiga de forma aislada se mueve de manera aleatoria, pero al encontrar el *camino de feromonas*, decide si lo sigue o no. La probabilidad de que una hormiga elija un camino en lugar de otros viene determinada por la cantidad de feromona en el potencial camino de interés. Con una continua acción de la colonia, el camino más corto es más frecuentemente visitado y se convierte en más atractivo para el resto de hormigas (Figura 9). A su vez, la feromona depositada también se va evaporando con el paso del tiempo, debilitando aquellos caminos menos transitados. En definitiva, un grupo de hormigas, utilizando un simple mecanismo indirecto de comunicación a través de la feromona depositada, es capaz de encontrar el camino más corto entre dos puntos eligiendo el camino según los niveles de feromona. Se habla de comunicación indirecta ya que una hormiga sabe que por ese mismo punto han pasado otras hormigas de su especie, pero desconoce el destino final de las mismas.

Figura 9: Imagen representativa de un camino de feromonas

Las características principales de este proceso son una retroalimentación positiva- la probabilidad con la que una hormiga escoge un camino aumenta con el número de hormigas que previamente hayan elegido el mismo camino-, la computación distribuida y el uso de una heurística de *búsqueda voraz (greedy)* constructiva. La retroalimentación positiva permite un rápido descubrimiento de buenas soluciones. La computación distribuida evita una convergencia prematura, y la *búsqueda voraz* permite encontrar soluciones aceptables en los instantes iniciales del proceso de búsqueda.

Como ya se ha mencionado anteriormente, la metodología ACO se ha aplicado en problemas de optimización combinatorial (CO). Tal y como su nombre indica, la optimización combinatorial trata de

encontrar combinaciones o permutaciones de elementos disponibles en el problema. Por lo tanto, el problema a resolver debe poder dividirse en un conjunto de elementos finitos para que el algoritmo de optimización intente encontrar la combinación óptima de los mismos.

Muchas aplicaciones de la vida real pueden ser representadas de manera sencilla como problemas combinatoriales. De ahí que la metodología ACO haya sido satisfactoriamente probada en multitud de problemas discretos, entre otros problemas de ordenamiento secuencial (Sequential Ordering Problem [12]), enrutamiento de redes (Telecommunications Networks [22]), etc. A su vez, otra aplicación muy estudiada por este tipo de algoritmos ha sido la determinación de rutas de vehículos (Vehicle Routing Problem [13]), donde los resultados proporcionados por estos algoritmos han sido satisfactorios.

3.1.2 Implementación de ACO

El primer problema donde se aplicó el algoritmo ACO fue el Problema del Viajante (Traveling Salesman Problem), instancia de un problema NP-duro, que además incluye la resolución de un camino mínimo: dado un conjunto de ciudades, un viajante debe encontrar la ruta más corta visitando todas las ciudades una sola vez. El problema se representa mediante un grafo estático, donde un conjunto de aristas/uniones conectan una serie de nodos (ciudades a visitar). Cada nodo – o cada unión entre dos nodos, dependiendo de la representación del problema específico – representa un elemento del conjunto de soluciones.

El pseudo-código de la meta-heurística ACO se muestra en la Figura 10 [26]. Como se puede apreciar, el algoritmo se compone principalmente de 3 fases: construcción de la solución, actualización de feromonas y acciones daemon. Cada una de las fases se detalla a continuación

- Construcción de la solución: dirige la búsqueda de soluciones llevada a cabo por cada hormiga mediante un modelo de feromona. Este modelo es una distribución de probabilidad parametrizada sobre el espacio de soluciones.
- Actualización de feromonas: las soluciones encontradas en el procedimiento anterior son empleadas para modificar los valores de feromona, con el objetivo de concentrar la búsqueda en regiones del espacio que contengan soluciones de alta calidad.
- El objetivo del último procedimiento, acciones daemon, es implementar computaciones centralizadas, en caso de que las hubiera, que no pueden ser llevadas a cabo por una hormiga en solitario. A modo de ejemplo este procedimiento podría decidir depositar feromona extra en los elementos de la mejor solución obtenida hasta el momento. Las acciones daemon están incluidas en la Figura 10 como un procedimiento opcional.

```
Ant Colony Optimization metaheuristic
  while termination conditions not met do
    ScheduleActivities
      AntBasedSolutionConstruction( )
      PheromoneUpdate( )
      DaemonActions( )      {optional}
    end ScheduleActivities
  endwhile
```

Figura 10: Meta-heurística ACO [26]

Construcción de la solución

Las hormigas artificiales viajan de nodo a nodo construyendo posibles soluciones de manera probabilística, guiándose por un rastro de feromona artificial y por una información construida a priori de manera heurística. En la representación en grafo, cada arista recibe un peso, de esta manera, una hormiga situada en un nodo s_i^k decide a qué nodo moverse usando una elección al azar proporcional a los pesos asociados a cada arista. La probabilidad $p_{\{s_i^k,j\}}$ de escoger un nodo $j \epsilon Q_{s_i^k}$, donde $Q_{s_i^k}$ es el vecindario del nodo s_i^k, viene definida por la siguiente ecuación:

$$p_{\{s_i^k,j\}} = \frac{\left[\varphi_{\{s_i^k,j\}}\right]^{\alpha}\left[\eta_{\{s_i^k,j\}}\right]^{\beta}}{\sum_{l \epsilon Q_{s_i^k}}\left[\varphi_{\{s_i^k,l\}}\right]^{\alpha}\left[\eta_{\{s_i^k,l\}}\right]^{\beta}} \tag{9}$$

donde α y β son parámetros que controlan la influencia de las feromonas $\varphi_{\{s_i^k,j\}}$ y la heurística $\eta_{\{s_i^k,j\}}$ respectivamente.

Los algoritmos ACO son procesos iterativos. En cada iteración se *lanza* una colonia de H hormigas y cada una de las hormigas de la colonia construye una solución al problema. Cada hormiga dispone de una estructura de memoria básica privada, donde almacena la ruta realizada y el coste de la misma.

Actualización de feromonas

Una vez que las hormigas hayan construido sus soluciones, siempre de acuerdo a la calidad de las mismas, se actualiza la tabla de feromonas. Dicha actualización es lo que permite que vaya variando el conjunto de soluciones. La función heurística es habitualmente un componente estático, no varía a lo largo del proceso de búsqueda.

La actualización de la tabla de feromonas se realiza en primer lugar disminuyendo en un valor constante ρ $(0 \leq \rho \leq 1)$ todos los niveles de feromona mediante un proceso denominado evaporación y en segundo lugar, incrementando los niveles de feromona de aquellas aristas por las que haya transitado la hormiga h en su recorrido L.

El proceso de evaporación, implementado mediante la ecuación 7, evita una rápida convergencia del algoritmo, favoreciendo la exploración de nuevas áreas en el espacio de búsqueda:

$$\varphi_{\{s_i^k,j\}} \leftarrow (1-\rho)\varphi_{\{s_i^k,j\}} \quad \forall (s_i^k, j) \in A \tag{10}$$

siendo A el conjunto de aristas que conectan los nodos.

Tras la evaporación, los depósitos de feromonas de las aristas transitadas por la hormiga h en su recorrido L se incrementan $\Delta\varphi^h$, en función de la calidad (valor de la función objetivo) de la solución, tal y como muestra la siguiente expresión:

$$\varphi_{\{s_i^k,j\}} \leftarrow \varphi_{\{s_i^k,j\}} + \Delta\varphi^h \quad \forall (s_i^k, j) \in L^h \tag{11}$$

3.2 ACE: Extensión del algoritmo ACO

Como se vio en la introducción, el algoritmo de colonia de hormigas extendido (Ant Colony Extended, ACE) es una adaptación de la meta-heurística ACO: técnica de inteligencia artificial basada en el estudio del comportamiento colectivo de un sistema descentralizado (auto-organizados) como son las colonias de hormigas, las cuales comparten el conocimiento que se obtiene de la búsqueda de cada agente.

El algoritmo ACO es por tanto un sistema multi-agente, cada hormiga representa un agente, donde los agentes se coordinan mediante la información compartida en una estructura de datos común, la tabla de feromonas. Esta característica en una de las principales razones por las que ACE utiliza la meta-heurística ACO como esquema de partida [11].

La otra característica es que en ACO cada hormiga debe construir su propia solución o secuencia, transitando de nodo a nodo de manera probabilística. Las soluciones se construyen paso a paso, escogiendo el siguiente elemento en la secuencia de manera independiente, lo cual define a ACO como una meta-heurística constructiva. Este tipo de meta-heurísticas presenta un esquema donde conocimiento y solución son elementos separados, lo que otorga una mayor flexibilidad, a diferencia de las meta-heurísticas de búsqueda y evolutivas donde no existe una separación definida entre la generación de soluciones y el conocimiento. En este caso, el conocimiento viene representado por una colección de soluciones y una nueva solución se obtiene por transformaciones de anteriores.

ACE busca completar la meta-heurística ACO mediante la inclusión de la representación en espacio de estados y de nuevas propiedades *observadas* en colonias de abejas reales [11]. En esta sección se detallan las principales diferencias entre los algoritmos ACO y ACE, y la implementación de este último.

3.2.1 Representación en Espacio de estados

En general el algoritmo ACO, desarrollado originalmente para solucionar problemas de optimización combinatorial, utiliza una representación en grafo. A pesar de que muchos problemas reales pueden ser representados mediante un grafo, existe un importante conjunto de problemas para los que esta representación puede no ser la mejor alternativa: problemas de optimización que requieren de la obtención de variables continuas. Esto ocurre en problemas continuos (si todas las variables son continuas) o en problemas de variables mixtas, si únicamente algunas variables son continuas.

Otro de los principales inconvenientes de la representación en grafo empleada por ACO es la necesidad de construirlo antes de la aplicación del algoritmo, lo que dificulta su aplicación en problemas donde se desconoce el espacio de búsqueda y por lo tanto, debe ser descubierto según se realiza la búsqueda. Además, si el grafo a construir es de un tamaño considerable puede generar problemas de rendimiento y de convergencia [11].

Una de las modificaciones de ACE con respecto al algoritmo original ACO es que utiliza una representación de espacio de estados. A diferencia de la representación en grafo, éste no necesita construirse previamente, sino que se va descubriendo. Esto implica que ACE arranca con una tabla de feromonas vacía que va llenando según avanza la búsqueda. De esta manera, se pueden evitar fácilmente problemas de rendimiento en problemas complejos (tiempo de computo).

Además, el algoritmo ACE únicamente guarda la información de aquellos estados visitados para obtener una solución particular. Es decir, no se almacena información de aquellos estados que no han sido visitados durante la búsqueda de una trayectoria, a diferencia de lo que ocurre en ACO. Esto reduce considerablemente el tamaño de la tabla de feromonas

El resultado es un nuevo algoritmo de hormigas, Ant Colony Extended (ACE), que ayuda a extender la aplicación de ACO a problemas que son difíciles de representar mediante un grafo. A pesar de estos cambios el algoritmo ACE sigue siendo válido para problemas combinatoriales [10], ya que la representación en grafo no deja de ser una representación particular de la representación en espacio de estados.

3.2.2 Colonia de hormigas + Colonia de abejas

ACE se inspira en estudios de investigadores biológicos, e incorpora a la meta-heurística ACO ideas de la toma de decisiones colectivas de las colmenas de abejas [11].

Ambos sistemas (hormigas y abejas) difieren en la forma en la que almacenan la información. Mientras en un modelo inspirado en las abejas la información se almacena en los agentes, en el caso de las hormigas ésta es común y reside en la tabla de feromonas. Las abejas aportan por tanto un componente temporal, mientras las hormigas añaden un componente local de la información [11]. Al combinar la inspiración de ambos modelos, almacenamiento en frecuencia de las abejas más fragmentación en función del estado típico de las hormigas, se obtiene un sistema que permite almacenar un número superior de secuencias. La novedad radica en el uso de la tabla para incorporar el componente típico de las colmenas de las abejas, lo cual permite que distintos tipo de agentes trabajen de manera asíncrona con información diferente y busquen la convergencia hacia soluciones distintas.

El algoritmo ACE implementa la combinación de ambos modelos asignando a los agentes, que como en ACO siguen siendo hormigas, diferentes roles asociados a diferentes tareas. De esta manera, se pueden distinguir dos tareas en las actividades de búsqueda en las hormigas. La primera consiste en la localización de las fuentes de comida y la segunda en la explotación de las fuentes localizadas. Estas dos tareas están coordinadas: la explotación de fuentes de comida no empieza a menos que la tarea de localización haya finalizado con éxito. Las hormigas que localizan una fuente de comida, estimulan otras hormigas para empezar la recolección de la misma. Una vez que la tarea de explotación ha comenzado, el éxito en la recolección de comida controla la ejecución de la tarea. Una hormiga que transporta una porción de comida a la colonia, no continúa recolectando comida, sino que esa hormiga pasa a un estado inactivo. Para volver a empezar a recolectar comida otra vez, ésta tiene que volver a ser activada por otra hormiga que vuelve a la colonia transportando comida.

Los agentes son por lo tanto de dos tipos:

- Patrollers: hormigas con una tarea asignada de localizar fuentes de comida, antes de que la actividad de búsqueda haya comenzado.
- Foragers: hormigas cuya tarea consiste en la explotación de las fuentes de comida descubiertas por las patrollers.

A diferencia de lo que ocurre en otros algoritmos ACO, esta división de tareas lleva a un uso diferente de la información de la feromona y la heurística; éstas no se combinan para la toma de decisiones en

la construcción de la solución. En ACE una hormiga puede construir una solución, usando tanto la información de la feromona como la de la heurística, según el rol de la misma. Nunca se combinan ambas fuentes de información. En función del tipo de agente, patroller o forager, éste tendrá tendencia a utilizar más a menudo un tipo u otro de información.

Por otro lado, en el proceso de búsqueda de una solución, el algoritmo debe mantener un equilibrio entre la exploración (búsqueda de nuevas soluciones) y la explotación (búsqueda de soluciones similares a la actual). La población de hormigas en los algoritmos ACO es homogénea, por lo que el balance entre exploración (heurística) y explotación (feromona) se realiza a través de las variables α y β de la ecuación 6.

Por su parte el algoritmo ACE propone una población heterogénea, y realiza el equilibrio entre exploración (patroller) y explotación (forager) en base a los dos tipos de hormigas. Para ello, implementa un mecanismo de auto-regulación, el cual controla el número y el rol de las hormigas involucradas en la búsqueda de las diferentes etapas. Este mecanismo se denomina dinámica de población y hace evolucionar en número de hormigas asignadas a cada tarea durante la búsqueda, de acuerdo al resultado logrado hasta el momento.

3.2.3 Implementación de ACE

El algoritmo ACE sigue un esquema similar al mostrado en la Figura 10 (El pseudo-código de la meta-heurística ACO). La construcción de la solución es implementada mediante el procedimiento *Paso de búsqueda*: cada una de las hormigas activas realiza un paso en la búsqueda con el objetivo de añadir un nuevo componente a la solución que está construyendo (la solución final se compone de N elementos).

Cada hormiga va construyendo su propia solución en cada paso de iteración y solo cuando haya completado la misma optará a poder actualizar la tabla de feromonas. Sin embargo, únicamente las búsquedas de los agentes que hayan tenido éxito, serán las adecuadas para modificar la feromona. El éxito (grado de calidad) se evalúa mediante un criterio de umbral. El grado de calidad se asigna comparando el coste de la solución construida, $J(S_a)$, con el coste medio μ calculado sobre soluciones de calidad conocidas hasta el momento, es decir encontradas por el algoritmo en iteraciones previas. En caso de que el agente no haya tenido éxito, se descarta la información de la búsqueda. La Figura 11 muestra el pseudo-código de la meta-heurística ACE [11]:

A diferencia de ACO (Figura 10), ACE añade un nuevo procedimiento, la dinámica de población. Una vez que una hormiga haya finalizado una búsqueda y ésta sea de calidad, puede reclutar nuevas hormigas. Una hormiga que esté realizando una cierta tarea, puede estimular a otra hormiga que se encuentre inactiva para que comience a desempeñar la misma tarea que ella realiza u otra distinta, (procedimientos *fracaso (a)* o *éxito (a)* en la Figura 11). Finalmente la hormiga es eliminada de la población. Cada vez que una hormiga activa haya completado una iteración, la población es actualizada mediante el procedimiento de Reclutamiento (). Esto completa la dinámica de población.

Es importante destacar que la arquitectura asíncrona de este algoritmo hace que no todas las hormigas completen la solución en el mismo ciclo (iteración) y que tampoco los procedimientos de actualización de feromona o de dinámica de población sean ejecutados cada ciclo. Es decir, una hormiga no tiene que esperar a que el resto de hormigas finalice la construcción de sus soluciones para ejecutar el resto de procedimientos.

```
1: procedure BUCLE PRINCIPAL
2:    for a ∈ Población do
3:       pasoBusqueda(a)
4:    end for
5:    for a ∈ Población do
6:       if finalizado(a) == T then
7:          if J(S) < μ then                    ▷ Criterio de umbral
8:             actualizaMedia(S,m)
9:             actualizaFeromona(Aq,Au,λ)
10:            exito(a)                         ▷ Dinámica de población
11:         else
12:            fracaso(a)                       ▷ Dinámica de población
13:         end if
14:      end if
15:   end for
16:   reclutamiento()                            ▷ Dinámica de población
17: end procedure
```

Figura 11: Pseudo-código ACE: bucle principal

En general, los algoritmos ACO utilizan una tabla de feromonas para modelar el grafo de construcción. Esta tabla contiene los valores de feromona (pesos) asignados a las uniones entre los nodos. Como ya hemos comentado, ACE no utiliza una representación en grafo, por lo que en este caso, la tabla de feromonas es una tabla asociativa que contiene información sobre estados visitados y las acciones llevadas a cabo en esos estados. La clave o entrada de la tabla son los estados, y los datos son pares de acción-probabilidad. Una vez que una hormiga alcanza uno de los estados de la tabla, ésta selecciona de forma aleatoria una acción de entre la lista de acciones disponibles, donde cada acción tiene una cierta probabilidad de ser escogida. El esquema de la tabla de feromonas viene representado a continuación:

Tabla 1: Estructura de la tabla de feromonas ACE: el símbolo s representa un estado, u una acción y φ un valor de probabilidad

Key	Value
s_i^0	$\left[\left(u_i^0, \varphi_0\right), \left(u_i^1, \varphi_1\right), \cdots\right]$
s_i^1	$\left[\left(u_i^1, \varphi_1\right), \left(u_i^2, \varphi_2\right), \cdots\right]$

La probabilidad de seleccionar una acción en función de un estado viene definida por la ecuación:

$$P_\varphi\left(u_i^k \middle| s_i^k\right) = \frac{\varphi_{u_i}^{s_n}}{\sum_{j \in N_\varphi} \varphi_{u_j}^{s_n}} \tag{12}$$

donde $\varphi_{u_i}^{s_n}$ es el valor de probabilidad de una acción u_i en el estado s_n y N_φ es el número de acciones disponibles. Solo aquellos estados previamente visitados por las hormigas forman la clave de la tabla de feromonas y solo aquellas acciones realmente ejecutadas en un estado son registradas en ella. La tabla de feromonas se inicializa vacía, y se va rellenando gradualmente según la información proveniente de las búsquedas con éxito [11].

A continuación se realiza una descripción detallada de los distintos procedimientos ACE, siguiendo el esquema mostrado en la Figura 11.

3.2.3.1 Construcción de la Solución

Los algoritmos ACO clásicos combinan la información de la feromona y la heurística para obtener la probabilidad de añadir un nuevo elemento a una solución en construcción. En cambio, el algoritmo ACE únicamente utiliza la información de la feromona o la información de la heurística para añadir un elemento a la solución, pero no ambas a la vez. En ACE la tabla de feromonas contiene directamente la probabilidad de la toma de decisión para la construcción de soluciones (Tabla 1). Al existir dos fuentes de información es necesario crear una política de control que regule el uso de una u otra información.

De esta forma, en función del tipo de hormiga (forager o patroller) que realiza la llamada al procedimiento *Paso de Búsqueda*, y también de la información (tabla de feromonas, heurística) disponible en ese momento, se selecciona una acción u otra. El procedimiento *Paso de Búsqueda* se muestra en la Figura 12 [11]. Si la hormiga que realiza la llamada es una forager, ésta utiliza la información de la tabla de feromonas y P_{φ} (ecuación 9) para seleccionar la siguiente acción. Solo en el caso de que la tabla de feromonas no contenga información, i.e. el estado actual no haya sido visitado anteriormente por ninguna otra hormiga, las hormigas foragers recurren a la información proveniente de la heurística.

Por otro lado, si la hormiga es una patroller, ésta puede utilizar ambas fuentes de información, tabla de feromonas o heurística, para seleccionar la próxima acción. Si una hormiga utiliza la información proveniente de la heurística, la probabilidad de seleccionar una acción $P_\eta(u_i|s_n)$, viene definida por una distribución de probabilidad derivada de algún tipo conocimiento inicial que se tenga del problema.

Para determinar qué tipo de información utilizará la hormiga patroller, el procedimiento implementa la siguiente política de decisión:

- Si la última información utilizada provenía de la tabla de feromonas (P_φ) o el estado actual es el estado inicial (s_0):

$$\begin{cases} \chi < 1-(1-\gamma_1)^{1/N} & \rightarrow P_\eta \\ \chi \geq 1-(1-\gamma_1)^{1/N} & \rightarrow P_\varphi \end{cases} \tag{13}$$

- Si la última información empleada provenía de la heurística$\left(P_\eta\right)$:

$$\begin{cases} \chi < \gamma_2 & \rightarrow P_\eta \\ \chi \geq \gamma_2 & \rightarrow P_\varphi \end{cases} \tag{14}$$

donde χ es un número aleatorio generado a partir de una distribución uniforme U(0,1). Los términos $\gamma_1 \in [0,1)$ y $\gamma_2 \in [0,1)$ son parámetro definidos por el usuario. Los símbolos "$\rightarrow P_\varphi$" y "$\rightarrow P_\eta$" indican el uso de la información de la feromona o de la heurística respectivamente. El termino N es el número de pasos que una hormiga debe realizar para construir una solución. Obviamente, este proceso tiene lugar siempre y cuando la tabla de feromonas tenga información disponible, sino la hormiga utiliza directamente la heurística (P_η).

Los parámetros γ_1 y γ_2 pueden ser utilizados para regular el uso de la información de la heurística, i.e. el ratio de exploración, búsqueda de patrollers. Para $\gamma_1 = 0$, el patroller no adquiriría información nueva, por lo que no habría diferencia entre una hormiga forager y otra patroller.

```
1: procedure PASO BUSQUEDA(a)
2:    if a es forager then
3:       Feromona pasoBusqueda(a)
4:    else
5:       decision ← politica de control
6:       if decision = true then
7:          Heurística pasoBusqueda(a)
8:       else
9:          Feromona pasoBusqueda(a)
10:      end if
11:   end if
12: end procedure
```

Figura 12: Descripción en pseudo-código ACE: procedimiento *Paso de Búsqueda*

3.2.3.2 Actualización de Feromona

Cuando una hormiga finaliza la construcción de una solución (N elementos), puede acceder a modificar la tabla de feromonas con la información de los estados visitados y las acciones realizadas en dichos estados.

Tal y como se indicaba anteriormente, la tabla de feromonas únicamente es modificada por hormigas que hayan tenido éxito en su búsqueda, i.e. hormigas que han encontrado una solución cuyo coste es menor que la media μ ($J(S_a) < \mu$), en la Figura 11. Cuando una hormiga obtiene una solución con coste mayor que μ, se considera que ha fracasado y la solución se descarta.

Sin embargo, antes de actualizar la tabla de feromonas, una hormiga con éxito actualiza el valor de umbral μ. El número de muestras anteriores involucradas en el cálculo de μ viene definido por el parámetro γ_3. Para el caso en el que $\gamma_3 = \infty$, esta media móvil sería simplemente la media "clásica": el valor ponderado de todas las soluciones conocidas. Para cualquier otro valor de γ_3, el umbral μ representa una media móvil. El procedimiento de actualización de media se muestra en la Figura 13 [11].

Aunque no se muestra en la descripción del pseudo-código, el valor de μ se inicializa a ∞. De este modo siempre se acepta la solución proveniente de la primera búsqueda.

```
1: procedure ACTUALIZA MEDIA(S, m)
2:     m ← m + 1
3:     L_μ ← add(J(S))
4:     if m = 1 then
5:         μ ← J(S)                          ▷ Inicialización de la media
6:     else
7:         if m < γ3 then
8:             μ ← μ + (J(S) − μ)/m
9:         else
10:            μ ← μ + J(S)/γ3 − L_μ(γ3)/γ3
11:            L_μ ← extraer(γ3)
12:            m ← m − 1
13:        end if
14:    end if
15: end procedure
```

Figura 13: Descripción en pseudo-código ACE: procedimiento para modificación del valor medio. El símbolo S indica una solución, m el número de muestras y L_μ representa la lista de muestras empleada para calcular la media.

El procedimiento para actualizar la tabla de feromonas se describe a continuación: para cada estado visitado, una hormiga con éxito modifica el contenido de la tabla modificando el valor asociado $\varphi(s_i^k)$ a cada acción. Dicho valor asociado refleja la frecuencia relativa: la probabilidad de utilizar una determinada acción en un determinado estado. Dicho de otra forma, se incrementa la probabilidad de las acciones ejecutadas, mientras que la probabilidad de las acciones no realizadas, pertenecientes a los estados visitados, disminuye. Este proceso de actualización se lleva a cabo mediante la ecuación 15 [7].

$$\begin{cases} \varphi_{u_i}^{s_v} \leftarrow \varphi_{u_i}^{s_v} + \lambda \cdot \left(1 - \varphi_{u_i}^{s_v}\right) & \forall u, s \in S \\ \varphi_{u_j}^{s_n} \leftarrow \varphi_{u_j}^{s_n} - \lambda \cdot \varphi_{u_j}^{s_n} & \forall u, s \notin S \end{cases} \tag{15}$$

donde S es la solución construida por una hormiga con éxito, φ_u^s es el valor de probabilidad al seleccionar la acción u en el estado s, u_i representa las acciones utilizadas por la hormiga para construir la solución S, u_j son acciones también disponibles en la tabla de feromonas para el estado s pero que no han sido utilizadas por la hormiga, s_n representa los estados visitados por la hormiga mientras iba construyendo la solución S. Finalmente, el término λ es la calidad asociada a la solución construida por la hormiga,

$$\lambda = \begin{cases} \frac{J(S)-\mu}{J_{best}-\mu} & J(S) < \mu \\ 0 & J(S) \geq \mu \end{cases} \tag{16}$$

La ecuación 13 normaliza la calidad asignada a una solución. Una solución S tendrá mayor calidad a medida que su coste $J(S)$ se acerque al coste de la mejor solución J_{best}, y calidad 0 si su coste es igual o inferior al valor umbral μ que indica si una solución es aceptable o no.

El contenido de la tabla de feromonas es reemplazado en mayor medida en función de λ. Es importante remarcar que este reemplazo puede darse porque λ tenga una valor próximo a 1, pero también puede darse si se acumula el efecto de sucesivas actualizaciones con valores menores de 1. El contenido se conservará únicamente si la acción utilizada es siempre la misma. La Figura 14 [11] muestra el procedimiento para actualizar la feromona.

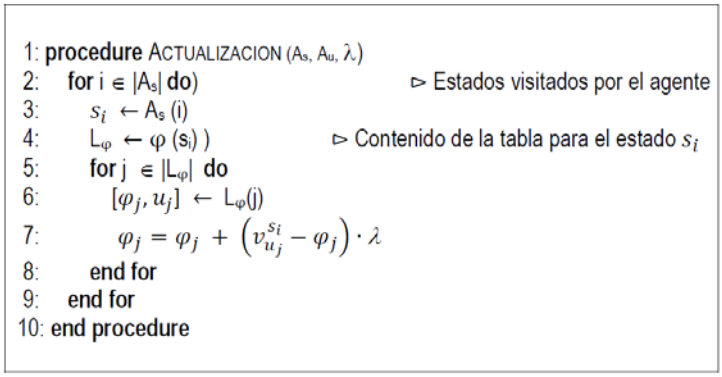


Figura 14: Descripción en pseudo-código ACE: procedimiento de actualización de feromona. Los símbolos A_s y A_u representan el número de estados visitados y el número de acciones ejecutadas por el agente, respectivamente

El uso de la media (μ) del coste de las soluciones conocidas hasta el momento para determinar el éxito o el fracaso de la búsqueda, permite mantener soluciones, que a pesar de ser *peores* que la mejor solución actual, podrían llevar el proceso de búsqueda al óptimo. Esta técnica conocida como *worsening* es empleada en algoritmos tales como Simulated Annealing para escapar de mínimos locales [14]

3.2.3.3 Procedimiento de dinámica de población

Tal y como se describía anteriormente, el algoritmo ACE contiene dos tipos de hormigas diferentes que desempeñan distintas tareas. La proporción del número de patrollers con respecto al número de foragers no es siempre la misma a la largo de la búsqueda, si no que el algoritmo la regula mediante el procedimiento denominado dinámica de población. Mediante este procedimiento, el algoritmo permite implementar un mecanismo que regula la exploración y explotación.

En una primera etapa, una vez que una hormiga ha finalizado la construcción de una solución, pasa a un estado inactivo. Dependiendo de si la hormiga ha tenido éxito o no, se activa el procedimiento Éxito o el procedimiento Fracaso (líneas 9 y 11 en la Figura 11). Estos procedimientos determinan el tipo y número de hormigas a ser reclutadas.

En una segunda fase, una vez que toda la población de hormigas activas ha sido procesada, se procede a reclutar el número de hormigas predeterminado mediante el procedimiento Reclutamiento() (línea 16 en la Figura 11). Estos tres procedimientos comparten el siguiente conjunto de variables (contadores):

- RP, número de patrollers a reclutar.
- RF, número de foragers a reclutar.
- UP, número de patrollers sin éxito desde el último patroller con éxito.
- SP, número de patrollers con éxito desde el último forager con éxito.
- FR, número de foragers activo.
- PR, número de patrollers activo.

Exito

Un forager con éxito recluta un patroller y resetea a cero el contador SP.

Un patroller con éxito incrementa el valor del contador SP, resetea el contador UP y recluta un patroller o un forager dependiendo del número de foragers activos y del número de patrollers con éxito contabilizados mediante SP. La Figura 15 [11] muestra el pseudo-código del procedimiento Éxito.

```
1: procedure ÉXITO(ant)
2:     if ant es Forager then
3:         FR ← FR − 1
4:         SP ← 0
5:         RP ← 1
6:     else
7:         PR ← PR − 1
8:         SP ← SP + 1
9:         UP ← 0
10:        if SP ≥ FR then
11:            RF ← 1
12:        else
13:            RP ← 1
14:        end if
15:    end if
16: end procedure
```

Figura 15: Descripción en pseudo-código: Procedimiento Éxito

Fracaso

Cuando un forager fracasa, éste pasa a un estado inactivo y disminuye el valor del contador de foragers activos (FR). Por otro lado, en caso de que sea un patroller el que fracasa, primero se incrementa el contador de los patrollers sin éxito (UP) y luego, recluta patrollers según el valor de UP, utilizando la función: *round (log(UP+1)).*

Por último, sin importar a qué tipo de hormiga le sea aplicado, el procedimiento calcula y normaliza la diferencia entre μ y J_{best}. Si esta diferencia fuera mayor que el ratio de foragers en la población de hormigas activas, se recluta un nuevo forager. La Figura 16 [11] muestra el pseudo-código del procedimiento fracaso.

```
1: procedure FRACASO(ant)
2:     if ant es Forager then
3:         FR ← FR − 1
4:         if (1 − J_best/μ) > FR/(FR+PR) then
5:             RF ← 1
6:         else
7:             RP ← 1
8:         end if
9:     else
10:        PR ← PR − 1
11:        UP ← UP + 1
12:        RP ← round(log(UP + 1))
13:        if (1 − J_best/μ) > FR/(FR+PR) then
14:            RF ← 1
15:        end if
16:    end if
17: end procedure
```

Figura 16: Descripción en pseudo-código del procedimiento Fracaso

Reclutamiento

Inicialmente, hasta que se obtenga una primera solución, la dinámica de población recluta un patroller cada ciclo i.e. cada vez que todas las hormigas activas realizan un *paso de búsqueda*. Esta primera solución resulta necesaria para disponer de un valor de umbral y poder clasificar los éxitos y fracasos. Una vez que se logre la primera solución, finaliza el proceso de inicialización y la población es actualizada según los valores de RP y RF. La Figura 17 [11] representa el procedimiento de reclutamiento

```
1: procedure RECLUTAMIENTO
2:     if muestras < 1 then                          ▷ Fase de inicialización
3:         Población ← activarPatroller()
4:         PR ← PR + 1
5:     else
6:         while RP > 0 do
7:             Población ← activarPatroller()
8:             PR ← PR + 1
9:             RP ← RP − 1
10:        end while
11:        while RF > 0 do
12:            Población ← activarForager()
13:            FR ← FR + 1
14:            RF ← RF − 1
15:        end while
16:    end if
17: end procedure
```

Figura 17: Descripción en pseudo-código del procedimiento Reclutamiento

3.2.3.4 Auto-regulación de exploración y explotación

En ACE, la tabla de feromonas contiene una distribución de probabilidad de las acciones a llevar a cabo en cada estado. Los foragers tienden a hacer converger esta distribución hacia una única acción. En contraste, los patrollers tienden a extender la probabilidad sobre diferentes acciones. Por ejemplo, un patroller puede añadir nuevas acciones que previamente no presentaba la tabla de feromonas. Dicho de otra manera, los foragers son útiles para re-direccionar y mantener la búsqueda en áreas prometedoras del espacio de soluciones, mientras que los patrollers, son adecuados para descubrir nuevas soluciones. Sin embargo estos últimos, pueden llegar a dispersar fácilmente la búsqueda, llevándola lejos de esas áreas prometedoras. La dinámica de población intenta coordinar la población de patrollers y foragers con el fin de mantener las capacidades de búsqueda de los patrollers limitadas en las zonas prometedoras.

La tarea de coordinación recae en el reclutamiento realizado por las hormigas con éxito: ellas siempre reclutan una hormiga del otro tipo. Por ejemplo, si los patrollers tienen éxito, ellas tienden a pasar a un estado inactivo, y el número de foragers tiende a incrementarse. Una vez que los foragers han tenido éxito, ellas reclutan nuevos patrollers. Esto ayuda a evitar que los patrollers se dispersen demasiado, porque los foragers ayudan a que la búsqueda converja hacia zonas adecuadas.

En general, un bajo número de foragers es suficiente para realizar este trabajo. Además, un número alto de ellas pueden llegar a estancar la búsqueda. Es por ello que el mecanismo para reclutar nuevos foragers, viene regulado por el número de este tipo de hormigas ya activas. Además su número

también está ajustado mediante la media y la mejor solución encontrada hasta el momento, la cual es una estimación aproximada de la convergencia del algoritmo.

Los patrollers, por el contrario, son reclutados en el momento en el que un forager tiene éxito y también, cuando ellos no son capaces de encontrar una solución lo suficientemente buena. En este último caso, se considera que la exploración está fallando por lo que resulta necesario reforzarla.

El comportamiento de la dinámica de población varía de problema a problema. La Figura 18 muestra una comparación de la evolución de la población activa de hormigas, durante 2000 iteraciones, para el problema MTS discreto tratado en la sección 4.1. La figura muestra como el algoritmo se auto-organiza, variando el tamaño de la población de hormigas y la relación entre el número de patrollers y foragers. Las observaciones indican los sucesivos periodos de éxito y fracaso

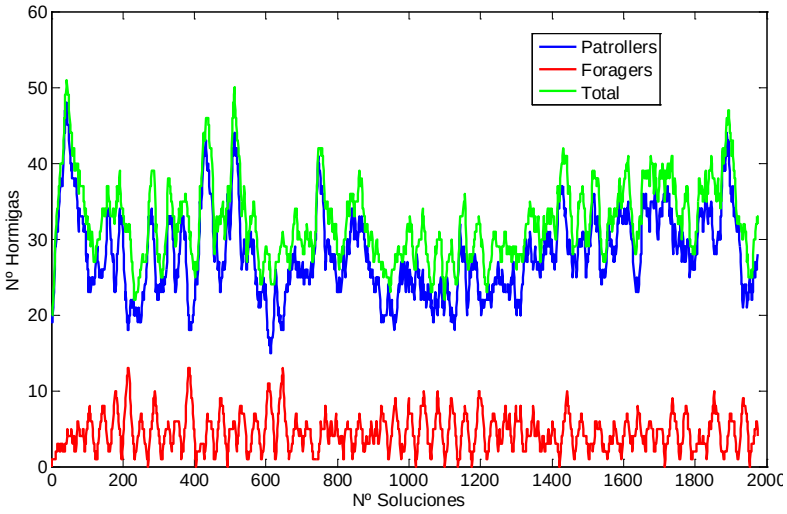


Figura 18: Evolución número de patrollers y foragers

4 Aplicación de ACE en el problema MTS

Este capítulo muestra el proceso seguido durante este TFM para resolver el problema MTS mediante ACE. El punto de partida de dicho proceso ha sido el algoritmo ACO existente para la versión del MTS con 8 acciones cardinales y el final, documentado en la sección 4.3, la obtención de una versión de ACE sobre acciones continuas. Los pasos intermedios, documentados respectivamente en las secciones 4.1 y 4.2, son el uso de ACE sobre un MTS discreto con 8 acciones cardinales y su extensión para un MTS con 360 acciones discretas diferentes.

Para ilustrar el funcionamiento de cada versión de ACE, el capítulo también recoge los resultados obtenidos por cada versión del algoritmo ACE desarrollado sobre algunos escenarios de búsqueda.

4.1 ACE para el problema MTS discreto con 8 acciones cardinales

Para que ACE sea capaz de resolver el problema MTS, donde los UAVs se controlan mediante las 8 acciones cardinales, de manera eficiente debe contar con una heurística razonable. En este caso, se ha utilizado la heurística presentada en [24], donde se realiza la aplicación de ACO en la resolución de problemas MTS con acciones discretas.

Esta heurística otorga al UAV una dirección de las 8 posibles definidas por los puntos cardinales (Figura 5). A partir de la posición actual s_i^k del UAV, se definen una serie de triángulos asociados a las direcciones c correspondientes a los 8 puntos cardinales. La Figura 1 muestra los distintos triángulos representados en varios colores. El número de celdas por triangulo depende de la dirección cardinal y los límites de la zona de búsqueda.

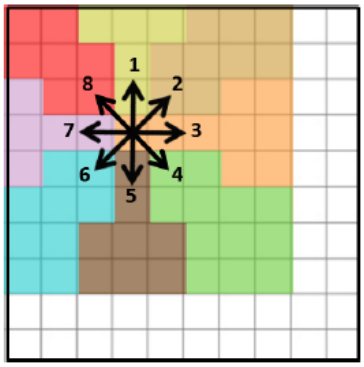


Figura 19: Representación de los distintos nodos considerados para calcular la heurística para cada una de las 8 acciones posibles de una hormiga en la posición central de las flechas [24].

La información de la heurística para cada dirección c se define a partir de siguiente expresión:

$$\eta_{\{s_i^k,c\}} = \sum_{j\in triangle(s_i^k,c,k)} f\left(distance\left(s_i^k,j\right)\right) b\left(\tau^k = j\right) \quad (17)$$

donde f() es una función decreciente con la distancia entre el nodo s_i^k y el nodo j. Por su parte, $b\left(\tau^k = j\right)$ es la probabilidad de encontrar el objetivo en el instante k sobre las posiciones del vértice j. De esta manera, la heurística hace que sea más probable el movimiento de las hormigas hacia las

regiones más prometedoras del mapa. La acción elegida se obtiene utilizando la heurística $\eta_{\{s_i^k,c\}}$ de manera aleatoria entre las 8 direcciones *c* referidas a los puntos cardinales.

4.1.1 Resultados

En lo que respecta al escenario elegido en este caso, éste cuenta con un solo agente (representado mediante una estrella roja) y un objetivo estático situado en alguna de las dos zonas de probabilidad que se representan mediante los círculos de la Figura 20. Una de estas dos zonas presenta una mayor probabilidad que la otra. La posición inicial del agente es central y a la misma distancia de ambas zonas de probabilidad. El espacio es discretizado mediante una malla de 20x20 celdas y el sensor utilizado por el UAV se modela de forma ideal. Sobre este escenario, se plantea la optimización de trayectorias de 20 pasos o elementos.

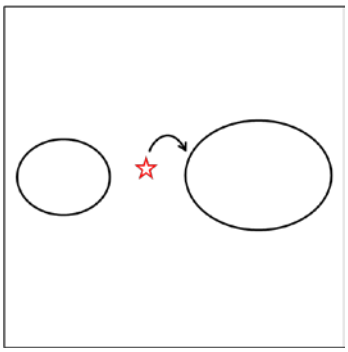


Figura 20: Escenario con 2 zonas de probabilidad y un único UAV

Debido a la naturaleza probabilística del problema MTS y a la naturaleza estocástica del algoritmo ACE, se realizará un estudio estadístico de los mejores resultados obtenidos durante la evolución de ACE al ejecutar 30 optimizaciones. Los parámetros de ACE utilizados para este fin son los mostrados a continuación. Como se puede comprobar se han tomado los valores habituales de esos parámetros [11]:

Tabla 2: Parámetros de ACE

Parámetro	ACE
γ_1	0.9
γ_2	0.5
γ_3	∞

La Figura 21 muestra los resultados obtenidos. En la primera columna se muestra la evolución de los mejores valores de la función objetivo obtenidos por el algoritmo ACE a lo largo del tiempo de cómputo. En la gráfica se representa la evolución del valor medio y la varianza, calculados sobre las 30 optimizaciones realizadas con el algoritmo ACE, de los valores de la función objetivo (tiempo esperado) obtenidos por el algoritmo en diferentes instantes de tiempo. Por otro lado, la segunda y tercera columnas muestran una instantánea del mapa de probabilidad y la trayectoria del UAV devuelta como solución de una de las 30 optimizaciones de esta versión de ACE.

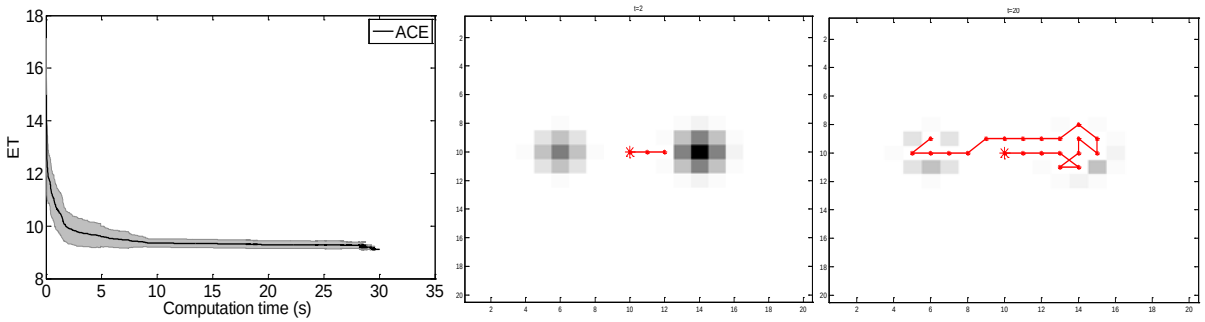


Figura 21: Resultados sobre el escenario a) Evolución tiempo esperado (ET) b) Modelo probabilístico instante intermedio c) Modelo probabilístico instante final

El algoritmo termina una vez que se haya superado el tiempo de computo establecido (30 segundos en este escenario), devolviendo como solución la trayectoria de la hormiga que haya conseguido una mejor evaluación de la función objetivo. Tal y como se puede apreciar, en la gráfica central y de la derecha, la solución propuesta por ACE se desplaza inicialmente hacia la zona de mayor probabilidad para posteriormente, tras recoger la probabilidad de esa región, desplazarse a la zona opuesta. Además, tal y como se muestra en la figura de la izquierda, ACE, con la ayuda de la heurística propuesta, es capaz de obtener soluciones con bajo tiempo esperado de búsqueda (ET) en un bajo tiempo de computo (convergiendo hacia la solución en menos de 10 segundos).

4.2 ACE para el problema MTS discreto con 360 acciones

Tal y como se detallaba en el capítulo 3.2, ACE es una extensión de ACO con un enfoque de búsqueda en espacio de estados. Dicha modificación permite el uso de ACE en aquellos problemas en los que no es posible o conveniente realizar una representación en forma de grafo.

En este apartado se describe el uso de ACE en la resolución de un problema MTS donde las acciones pueden tomar un número elevado (360) de valores diferentes, con el objeto de permitir definir trayectorias realistas de los UAVs. Además, también se considera que el espacio donde se encuentra el objetivo puede tener zonas de vuelo prohibidas (no sobrevolables por los UAVs). El problema de la Búsqueda en Tiempo Mínimo en aplicaciones reales, además de la incertidumbre asociada a la información, puede plantear el problema de la presencia de zonas prohibidas en la zona de búsqueda. En este caso la misión del planificador consiste en obtener un camino (en nuestro caso secuencia de acciones) que evite las zonas prohibidas y que permita a los agentes desplazarse por la región de búsqueda mientras realizan observaciones para la detección del objetivo en un tiempo mínimo.

ACE ya trata la presencia de zonas prohibidas (obstáculos) en [9], donde se utiliza para optimizar las maniobras para barcos autónomos. En este caso, el objetivo consiste en minimizar el tiempo de maniobra. En relación a la heurística, el estudio de dicho trabajo plantea dos tipos de heurística: velocidad y rumbo de los barcos. La heurística de velocidad se define como una distribución normal discretizada centrada en la velocidad deseada en el estado final, mientras que la de rumbo se configura a partir de las propiedades de auto-organización de un autómata celular. El resultado de esta última es un mapa de vectores: cada celda contiene un vector indicando el rumbo a seguir por el barco en ese punto [9].

La metodología propuesta en ese trabajo para la definición de la heurística de rumbo no permite una fácil adaptación a la naturaleza probabilística del problema MTS. Por lo tanto, en este Trabajo de Fin de Máster se propone una nueva heurística basada en el cálculo del campo de potencial. El objetivo es aprovechar las propiedades de ésta sencilla técnica para determinar una heurística de rumbo que sea capaz de tratar la presencia de obstáculos (zonas prohibidas para nuestros UAVs) tanto estáticos como móviles. A continuación se detallan las principales características de los campos de potencial y su utilización para el diseño de la heurística.

4.2.1 *Campo de Potencial Artificial*

El método de campo de potencial es una de las técnicas más populares en la generación de trayectorias para robot móviles. El robot (UAV) es considerado como una partícula sometida a diferentes fuerzas [19]. Por un lado existe una fuerza de atracción hacia el destino predefinido, y por otro una serie de fuerzas de repulsión destinadas a evitar la colisión con los obstáculos. Para definir estas fuerzas se considera un campo de potencial asociado al escenario, cuyo valor depende de su proximidad a los obstáculos y de la distancia al destino (ver Figura 22). El gradiente de dicho campo es utilizado como guía para trazar la trayectoria del UAV. De esta forma, el UAV sigue siempre la dirección que minimiza (máxima pendiente) el valor del campo de potencial, evitando los obstáculos. El mínimo absoluto está situado en el destino.

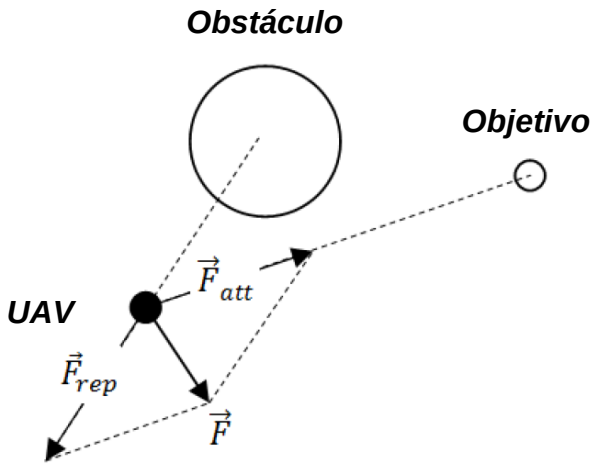


Figura 22: Dirección resultante de la fuerza artificial $\vec{F}_{res}(s_i^k)$, debido a las fuerzas de atracción y repulsiva

El campo de fuerzas $\vec{F}(s_i^k)$ que sufre el robot en Ω viene definido por:

$$\vec{F}(s_i^k) = -\nabla U(s_i^k) \qquad \nabla U(s_i^k) = \begin{pmatrix} \partial U/\partial x \\ \partial U/\partial y \end{pmatrix} \tag{18}$$

siendo $U(s_i^k) = U_{att}(s_i^k) + U_{rep}(s_i^k)$, el campo potencial en un punto $s_i^k \in \Omega$.

U_{att} es el potencial atractivo asociado a la meta s_{goal}, cuyo objetivo es hacer que el robot se aproxime a la meta, mientras que U_{rep} es el potencial repulsivo asociado a los obstáculos, el cual permite evitar que el robot se acerque a los mismos, ya sean estáticos como móviles.

El campo de potenciales responsable de la fuerza de atracción suele definirse como una función parabólica [19]:

$$\mathrm{U}_{\mathrm{att}}(s_i^k) = \frac{1}{2}\xi \cdot \rho_{\mathrm{goal}}^2(s_i^k) \tag{19}$$

donde ξ es un factor de escala y ρ_{goal}es la distancia euclídea $\left\|s_i^k - \mathrm{s}_{\mathrm{goal}}\right\|$

La fuerza de atracción derivada de este campo de potencial se define como:

$$\vec{\mathrm{F}}_{\mathrm{att}}(s_i^k) = -\nabla \mathrm{U}_{\mathrm{att}}(s_i^k) = -\xi\left\|s_i^k - s_{\mathrm{goal}}\right\| \tag{20}$$

Como puede observarse la fuerza de atracción consiste en un vector cuyo módulo es proporcional a la distancia entre el punto s_i^k y el objetivo, y cuya dirección apunta siempre hacia la configuración final.

El campo de repulsión se define como la suma de los campos generados por cada obstáculo individualmente, el cual puede definirse a partir de [19]:

$$\mathrm{U}_{\beta_{\mathrm{w}}}(s_i^k) = \begin{cases} \frac{1}{2}\vartheta \cdot \left(\frac{1}{\rho_{\mathrm{w}}(s_i^k)} - \frac{1}{\rho_0}\right)^2 & \text{si } \rho_{\mathrm{w}}(s_i^k) \le \rho_0 \\ 0 & \text{si } \rho_{\mathrm{w}}(s_i^k) > \rho_0 \end{cases} \tag{21}$$

donde ϑ es un factor de escala positivo, $\rho_{\mathrm{w}}(s_i^k)$ es la distancia del punto s_i^k al obstáculo identificado con el valor w, ρ_0 es una constante positiva denominada *distancia de influencia del obstáculo*. Así el campo de potencial repulsivo tiende a infinito conforme se acerca al obstáculo y se hace nulo más allá de ρ_0. La fuerza de repulsión asociada a cada obstáculo resulta:

$$\vec{\mathrm{F}}_{\beta_{\mathrm{w}}}(s_i^k) = -\nabla \mathrm{U}_{\beta_{\mathrm{w}}}(s_i^k) = \begin{cases} \eta \cdot \left(\frac{1}{\rho_w(s_i^k)} - \frac{1}{\rho_0}\right)\frac{1}{\rho_{\mathrm{w}}^2(s_i^k)} & \text{si } \rho_{\mathrm{w}}(s_i^k) \le \rho_0 \\ 0 & \text{si } \rho_{\mathrm{w}}(s_i^k) > \rho_0 \end{cases} \tag{22}$$

Y la fuerza repulsiva total a la que se ve sometido el robot

$$\vec{\mathrm{F}}_{\mathrm{rep}}(s_i^k) = \sum_{\mathrm{w}=1}^{\mathrm{r}} \vec{\mathrm{F}}_{\beta_w}(s_i^k) \tag{23}$$

Por tanto el vehículo se moverá en función de la siguiente fuerza resultante:

$$\vec{\mathrm{F}}_{\mathrm{res}}(s_i^k) = \vec{\mathrm{F}}_{\mathrm{att}}(s_i^k) + \vec{\mathrm{F}}_{\mathrm{rep}}(s_i^k) \tag{24}$$

En resumen el UAV se ve atraído por el objetivo y repelido por los obstáculos (estáticos o dinámicos). Utilizando esta información, se usa un campo de potencial artificial para modelar el entorno del UAV Figura 23. Por regla general, la posición del objetivo se representa como un mínimo local y los obstáculos como un máximo local. Para generar la trayectoria con estos potenciales solo se requiere calcular los gradientes.

Uno de los principales inconvenientes que presenta el cálculo de una trayectoria mediante los campos de potencial, es que éstos pueden resultar no convexos, presentando mínimos locales dejando estancado al UAV en un punto diferente al objetivo [15]. Además resulta necesaria una correcta estimación de los parámetros del campo de potencial que logre una adecuada evasión de obstáculos. En caso contrario, la trayectoria del UAV podría presentar oscilaciones en presencia de obstáculos dificultando el paso del mismo entre obstáculos cercanos.

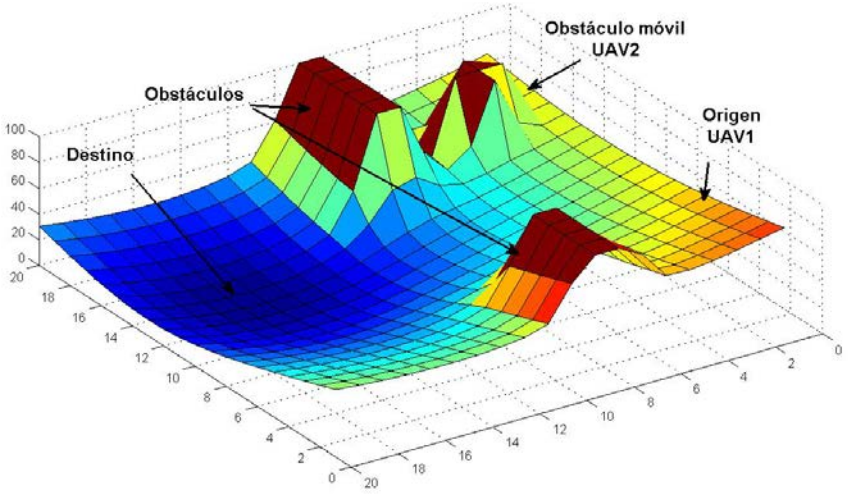


Figura 23: Campo de potencial calculado en un entorno con obstáculos

4.2.2 Implementación de una Heurística para el MTS basada en campos de potencial

El objetivo es por tanto aprovechar las propiedades de los campos de potencial para determinar una heurística de rumbo para el problema MTS. A cada paso de iteración de cada hormiga, la heurística determina la fuerza resultante en el estado actual de los UAVs (su posición s_i^k). La dirección de la fuerza resultante $\vec{F}_{res}(s_i^k)$ del campo de potencial artificial, es considerada la dirección de movimiento más prometedora para el UAV, ya que determina la dirección que debe seguir el vehículo con el fin de alcanzar la meta evitando las zonas de vuelo prohibidas (posición conocida). De esta forma, la planificación de movimiento se realiza de forma iterativa; en cada paso de búsqueda (línea 7 Figura 12), se calcula la fuerza artificial $\vec{F}_{res}(s_i^k)$ resultante en el estado actual del UAV (s_i^k).

Para la resolución del problema MTS tratado en este trabajo, la meta o destino del campo de potencial artificial viene determinada por las zonas de máxima probabilidad definidas a partir del modelo probabilístico $b(\tau^k)$. En caso de que el modelo presente varias zonas, la heurística determinará la región de máxima probabilidad y la designará como punto de destino. Este punto de destino es modificado por la heurística, a medida que los UAVs van realizando observaciones y actualizando el modelo probabilístico; siempre y cuando cambie la zona de máxima probabilidad.

Es decir la heurística modifica de forma iterativa el mapa de vectores en función del resultado de la búsqueda realizada por cada hormiga. La Figura 24 muestra un ejemplo de un mapa de vectores calculado a partir del campo de potencial. Los dos rectángulos en blanco representan obstáculos estáticos a evitar por los UAVs y la zona donde confluyen las flechas (posición 15,13) es donde se encuentra el máximo de $b(\tau^k)$ cuando los UAVs han alcanzado la posición s_i^k después de recorrer la trayectora $s_i^{0:k}$.

Por otra parte los UAVs entre ellos deben mantener una distancia de seguridad en vuelo. Esta distancia es modelada en forma de obstáculos dinámicos, ya que en cada iteración varía la posición de los UAVs. En los problemas con varios UAVs, al calcular la fuerza resultante para uno de ellos, el resto de los vehículos son tratados como obstáculos móviles con una localización conocida. De esta manera se genera un campo de repulsión entre UAVs, como si de un obstáculo se tratase. Este campo de repulsión permite definir la distancia de vuelo segura entre los UAVs. Esta circunstancia

hace que la heurística utilizada también permita considerar la presencia de obstáculos móviles en el entorno.

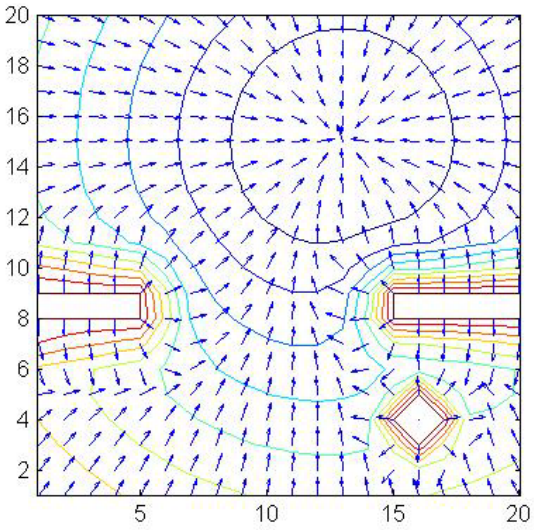


Figura 24: Mapa de vectores obtenido a partir del campo de potencial con obstáculos: cada celda contiene un vector (Fuerza resultante) indicando el rumbo a seguir por el UAV en ese punto. En este esquema se ha asignado el mismo módulo a todos los vectores con el fin de facilitar la compresión de la figura.

De la misma forma, como el punto de destino es actualizado en cada ciclo, y una vez conocido éste se determina la fuerza artificial $\vec{\mathrm{F}}_{\mathrm{res}}(s_i^k)$, la heurística permite también la resolución de problemas con objetivos dinámicos. A pesar de todo, en este trabajo únicamente se han considerado objetivos estáticos.

Una vez determinada la dirección a seguir por el UAV, la heurística construye una distribución de probabilidad centrada en dicha dirección o ángulo. A modo de ejemplo, la siguiente figura muestra una distribución de probabilidad centrada en el ángulo de 90º. La distribución se define de forma triangular para evitar que se tomen rumbos contrarios a la dirección sugerida por el campo de potencial.

4.2.3 Resultados

A continuación se muestran los resultados obtenidos en la variante del problema MTS recogido en esta sección (MTS con 360 acciones discretas) mediante la aplicación del algoritmo ACE y la heurística basada en el campo de potencial. Para ello se han utilizado dos escenarios en los que se cuenta con un espacio Ω cuadrado de 6x6 km^2 discretizado mediante una malla 20x20 celdas (de 300x300 m^2) y donde los UAVs vuelan a la velocidad $v_i^{c,k} = 60\mathrm{m/s}$ y a la altura $h_i^{c,k} = 300m$. En estos escenarios el tiempo de paso de la simulación es de 1 segundo y las observaciones se realizan cada 5 segundos. El sensor utilizado por el UAV se modela de forma ideal.

Los escenarios difieren en el modelo probabilístico del objetivo $\mathrm{b}(\tau^{\mathrm{k}})$, en la posición inicial de los UAVs y en el número N de acciones que componen la solución. Ambos escenarios cuentan con un único UAV (representado mediante una estrella roja).

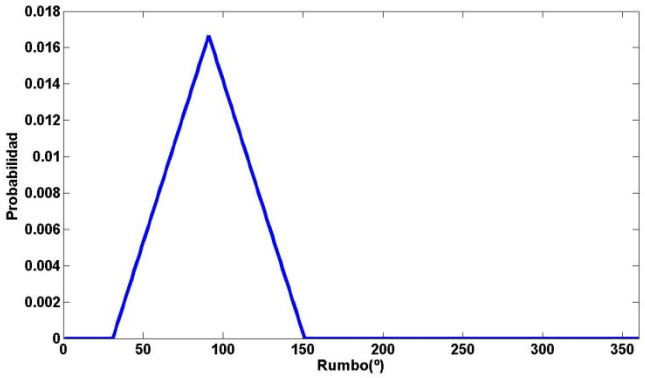


Figura 25: Distribución de probabilidad para la heurística de rumbo centrada en 90º (con una apertura triangular de ±60º).

Los parámetros de ACE utilizados para este fin son los mostrados a continuación. Como se puede comprobar se han tomado los valores habituales de esos parámetros en [11]:

Parámetro	ACE
γ_1	0.9
γ_2	0.5
γ_3	∞

Para cada escenario se realiza un estudio estadístico de los mejores resultados obtenidos durante la evolución de ACE al ejecutar 20 optimizaciones.

Escenario 1

El escenario cuenta con un objetivo estático situado en una zona circular de probabilidad de la Figura 26. Además muestra de forma esquemática la posición inicial del UAV y su orientación. Sobre este escenario, se plantea la optimización de trayectorias de 15 pasos en un único tramo.

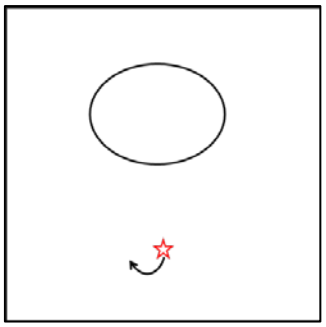


Figura 26: Representación esquemática del escenario 1. b) Modelo probabilístico inicial $b(\tau^0)$

Los resultados obtenidos se muestran en la Figura 27. En la imagen superior muestra la evolución de los mejores valores de la función objetivo, obtenidos por el algoritmo ACE a lo largo del tiempo de cómputo (200 segundos en este caso). En la gráfica se representa la evolución del valor medio y la varianza, calculados sobre las 20 optimizaciones realizadas con el algoritmo ACE, de los valores de la función objetivo (tiempo esperado) obtenidos por el algoritmo en diferentes instantes de tiempo.

Por otro lado, las inferiores muestran el instante inicial sobre el modelo probabilístico $b(\tau^0)$ (izquierda) y la mejor solución (mínimo tiempo esperado) obtenida entre todas las calculadas

(derecha), mostrando su trayectoria desde el punto inicial a la zona de máxima probabilidad. Además, se indica el valor de tiempo esperado obtenido por dicha solución.

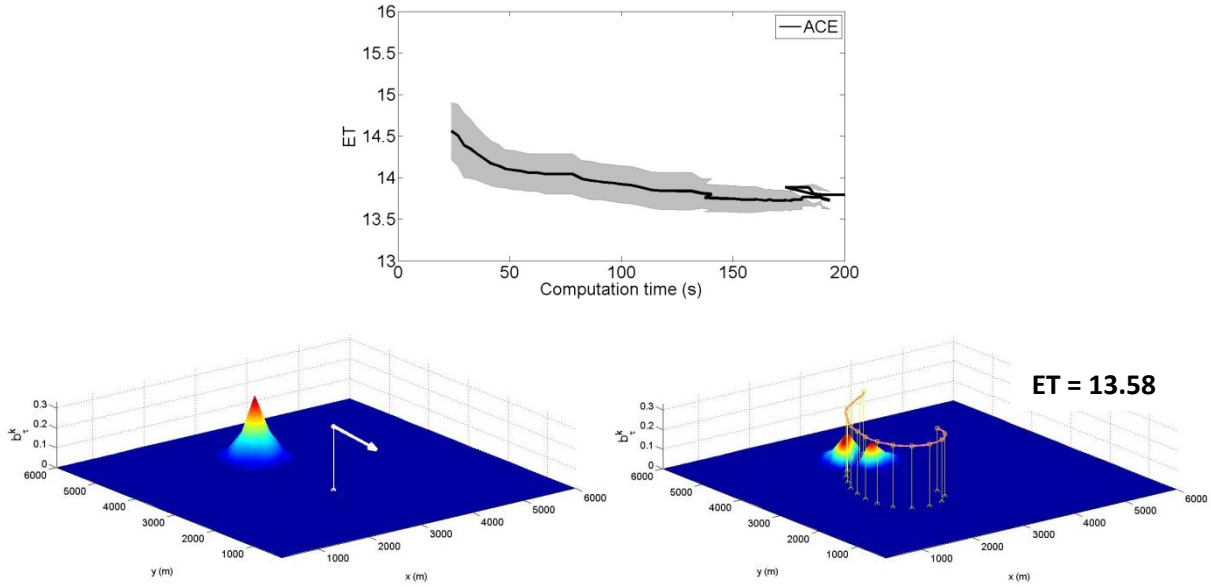


Figura 27: Resultados escenario 1 a) Evolución tiempo esperado (ET) b) Modelo probabilístico inicial $b(\tau^0)$ c) Mejor solución obtenida

Como se puede apreciar en la figura, a pesar de que la orientación inicial del UAV es la opuesta a la dirección en la que se encuentra la zona de máxima probabilidad, el algoritmo ACE, con la ayuda de la heurística basada en el campo de potencial, consigue ir variando la orientación del UAV progresivamente hasta alcanzar la zona de máxima probabilidad. Por su parte, la figura de la izquierda muestra cómo el tiempo esperado de búsqueda (ET) va convergiendo hacia la solución, la cual se alcanza al cabo de 150 segundos.

Escenario 2

El escenario escogido en este caso contiene un objetivo estático situado en alguna de las dos zonas de probabilidad que se representan mediante los círculos de la Figura 28. Una de estas dos zonas presenta una mayor probabilidad que la otra. Como se puede observar es el mismo escenario que el utilizado en el problema MTS con 8 acciones cardinales (Sección 4.1). El objetivo es observar el efecto de la dinámica del UAV en su maniobrabilidad. Sobre este escenario, se plantea la optimización de trayectorias de 20 pasos en un único tramo.

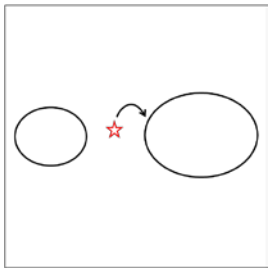


Figura 28: Representación esquemática del escenario 2 b) Modelo probabilístico inicial $b(\tau^0)$

Al igual que en el caso del escenario 1, a continuación se muestran la evolución del valor medio y la varianza, calculados sobre las 20 optimizaciones de los valores del tiempo esperado, el modelo probabilístico inicial y la mejor solución obtenida.

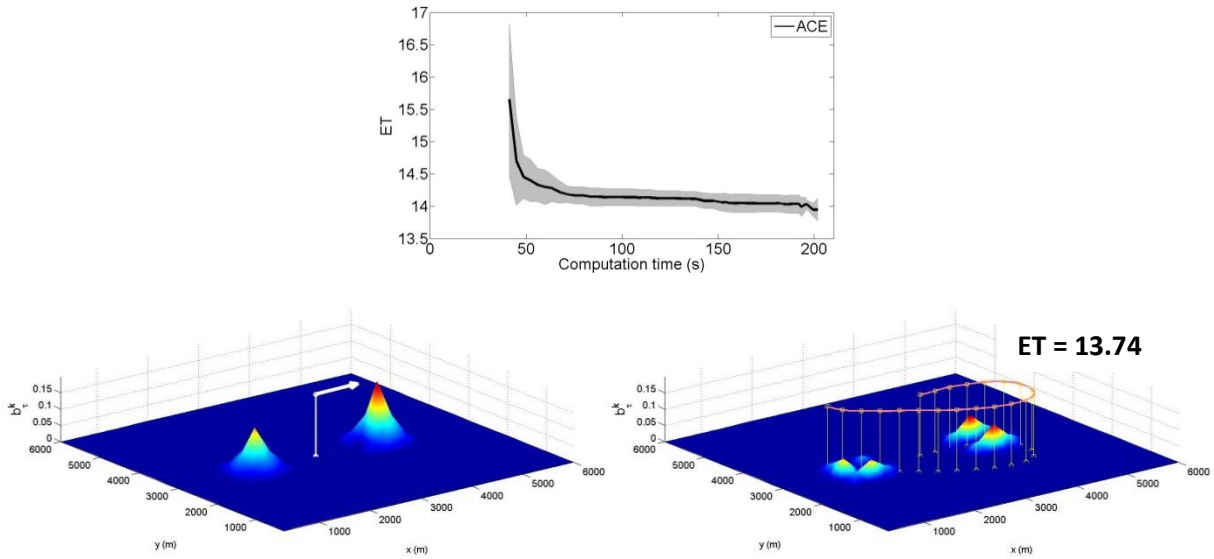


Figura 29: Resultados escenario 2 a) Evolución tiempo esperado (ET) b) Modelo probabilístico inicial $b(\tau^0)$ c) Mejor solución obtenida

La figura superior muestra cómo el algoritmo es capaz de obtener soluciones con bajo tiempo esperado de búsqueda (ET) en un bajo tiempo de computo (convergiendo hacia la solución en menos de 100 segundos).

Al igual que ocurría en el ejemplo del caso discreto con 8 acciones, la imagen inferior derecha muestra cómo inicialmente el UAV se desplaza hacia la zona de mayor probabilidad. Sin embargo, a diferencia de lo que ocurría en la Figura 21, debido a las restricciones impuestas por la dinámica del UAV, éste esta vez no recorre toda la zona. Además, tras recoger la probabilidad central de dicha zona, la heurística determina que existe otra región con un valor de probabilidad superior y modifica el punto de destino del campo de potencial. Como consecuencia de ello, la solución propuesta por ACE se desplaza a la zona opuesta.

El mallado utilizado para el cálculo de la heurística de rumbo en principio es infinito, es decir no se restringe el movimiento del UAV, pero únicamente se calcula para una determinada zona. En las simulaciones mostradas hasta el momento el gradiente del campo de potencial se obtiene para el mallado de la zona de búsqueda (20x20 celdas). Ello se debe a que el cálculo del gradiente requiere de una superficie determinada. Sin embargo, esta superficie podría reducirse con el fin aligerar la carga computacional del algoritmo. A modo de ejemplo, la muestra el resultado obtenido para el escenario 2, con un mallado para la heurística de rumbo de 9x9 celdas: en cada ciclo el campo de potencial es únicamente calculado en una matriz de 9x9.

Las dos imágenes de la primera fila de la Figura 30 muestran la evolución del valor medio y la varianza, calculados sobre las 20 optimizaciones de los valores del tiempo esperado y la mejor solución obtenida. La imagen de la fila inferior muestra una comparativa entre el resultado obtenido para ACE con un mallado de 20x20 y con un mallado 9x9 (ACE_{mr}, ACE mallado reducido). Hemos

eliminados la varianza con el fin de poder analizar la diferencia entre ambas. Como se puede observar el resultado es prácticamente el mismo.

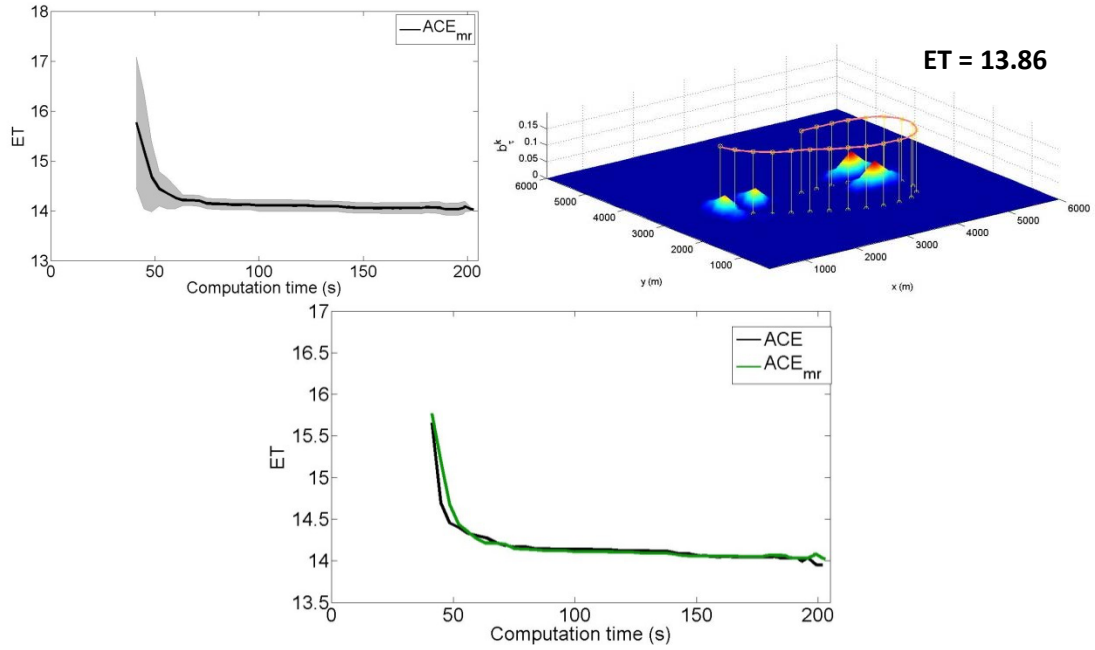


Figura 30: Resultados escenario 2 a) Evolución tiempo esperado (ET) b) Mejor solución obtenida c) Comparativa de ETs obtenidos a partir de un mallado de 20x20 y de 9x9

4.3 ACE con acciones continuas

A pesar de que los algoritmos ACO fueron originalmente diseñados para solucionar problemas discretos, su adaptación para solucionar problemas de optimización continuos está recibiendo cada vez más atención

[2]. Algunos de los primeros desarrollos incluyen el algoritmo CACO (Continuous ACO [1]) o el algoritmo (CIAC) (Continuous Interacting Ant Colony [8]). Sin embargo a estos algoritmos se les considera conceptualmente alejados de los clásicos problemas discretos ACO. Una de las últimas adaptaciones, y más próxima al esquema original de los casos discretos, es el algoritmo ACO_R (ACO extended to continuous domains) propuesto por Socha [26].

La adaptación de ACE a continuo, realizada para permitir la optimización de las trayectorias del UAV codificadas a partir de una secuencia de acciones que toma valores continuos, se ha realizado en base al planteamiento del algoritmo ACO_R. En este capítulo se realiza una breve descripción del mismo, así como la adaptación de ACE a problemas de optimización continuos.

4.3.1 Codificación de las feromonas en espacios discretos y continuos

Tal y como se indicaba en el capítulo 3, el algoritmo ACO almacena la información de la feromona en forma de tabla. Dicha información representa los pesos de todas las aristas que conectan los nodos: la probabilidad $p_{\{s_i^k,j\}}$ se define como la probabilidad de escoger un nodo $j \epsilon Q_{s_i^k}$ desde un nodo s_i^k, donde $Q_{s_i^k}$ es el vecindario del nodo s_i^k. El conjunto de probabilidades asociadas al vecindario $Q_{s_i^k}$

puede ser representado a partir de una distribución de probabilidad discreta, la cual es muestreada de forma aleatoria por la hormiga para determinar el siguiente elemento de la solución. La Figura 31 muestra un ejemplo de una distribución de probabilidad discreta [26].

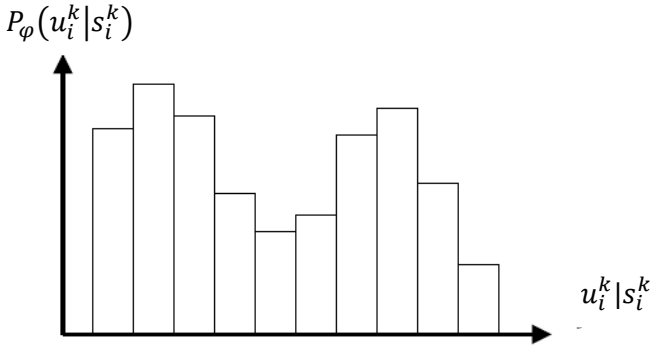


Figura 31: Distribución de probabilidad discreta [26]

En el caso particular de ACE, la información relativa a las feromonas se guarda mediante una tabla asociativa que contiene información sobre los estados visitados y las acciones llevadas a cabo en esos estados (apartado 3.2.3). La clave o entrada viene definida por los estados, discretizado según la malla empleada, mientras que la tabla registra pares de acción-peso. Una vez que una hormiga alcanza uno de los estados de la tabla, ésta selecciona de forma aleatoria una acción de entre la lista de acciones disponibles, donde cada acción tiene una cierta probabilidad de ser escogida. Por lo tanto, igual en ACO, la información almacena en cada estado viene definida por una distribución de probabilidad discreta.

En el caso continuo no es posible representar la información de la feromona de la misma forma. Se debe adoptar un enfoque diferente. En este sentido, el algoritmo ACO_R [26] plantea sustituir una distribución de probabilidad discreta (Figura 31) por otra continua: una función de densidad de probabilidad (Probability Density Function, PDF, en inglés). En consecuencia, en el algoritmo ACO_R una hormiga muestrea una PDF en lugar de elegir un elemento $p_{\{s_i^k,j\}} \epsilon Q_{s_i^k}$ (ec. 5). La siguiente figura muestra un ejemplo de una PDF.

4.3.2 ACO para optimización continua (ACO_R)

Socha plantea el cálculo de dicha PDF a partir del muestreo de un conjunto de soluciones denominada *archivo*. Antes de comenzar el algoritmo, este archivo de soluciones se inicializa con l soluciones generadas de forma aleatoria. Este se corresponde con la inicialización de la tabla de feromonas en ACO. Posteriormente, en cada iteración, se crean e nuevas soluciones, que son añadidas a las anteriores l soluciones. A continuación este conjunto de $l + e$ soluciones es ordenado, de acuerdo a la calidad de las mismas, de mejor a peor calidad. Se seleccionan las l mejores soluciones que vuelven a ser almacenadas en el archivo, el cual tiene un tamaño de $l * N$, siendo N el número de elementos que componen la solución. Cada fila del archivo se corresponde con una solución encontrada.

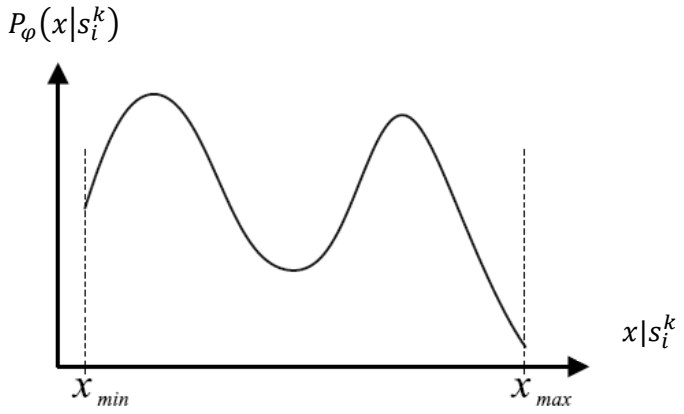


Figura 32: Función de probabilidad de densidad continua [26]

Para construir una solución, una hormiga elige, en cada ciclo, un valor para las variables de decisión. Para realizar este elección Sacha plantean el uso de una función de densidad de probabilidad basada en un kernel gaussiano, la cual se compone de varias funciones Gaussianas g_j^i, donde jes un índice de solución e i es un índice de elementos. La Figura 33 presenta un ejemplo de una PDF con kernel gaussiano construida a partir de 5 funciones Gaussianas.

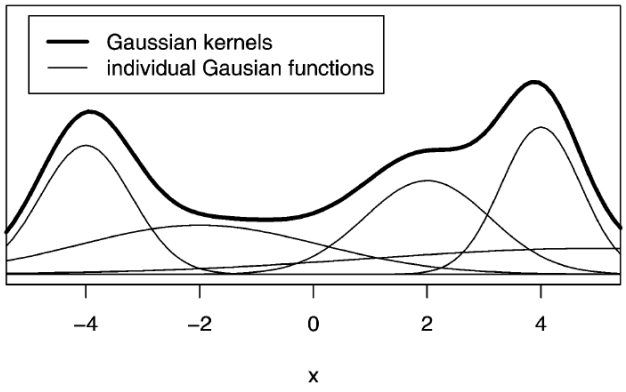


Figura 33: Ejemplo de una PDF kernel gaussiano construida a partir de 5 funciones Gaussianas [26].

El kernel gaussiano para el elemento $i \in \{1, \dots, N\}$ viene definido por la expresión [26]:

$$G^i(x) = \sum_{j=1}^{k} w_j g_j^i(x) = \sum_{j=1}^{k} w_j \frac{1}{\sigma_j^i \sqrt{2\pi}} e^{\frac{(x-\delta_j^i)^2}{2\sigma_j^{i\,2}}} \quad (25)$$

donde δ_j y σ_j representan la media y desviación estándar de la solución j respectivamente. El peso w_j, de la solución j, es calculado a partir de

$$w_j = \frac{1}{qk\sqrt{2\pi}} e^{\frac{(rank(j)-1)^2}{2q^2k^2}} \quad (26)$$

donde q es un parámetro del algoritmo. Un valor de q pequeño otorga preferencia por las soluciones de mejor calidad, mientras un valor de q grande, crea una probabilidad más uniforme.

En el proceso de construcción de una solución, cada elemento es tratado de forma independiente. Cada hormiga elige probabilísticamente una de las soluciones de acuerdo a su correspondiente peso:

$$p_j = \frac{w_j}{\sum_{i=1}^{k} w_i} \tag{27}$$

A continuación, el algoritmo muestrea una PDF gaussiana entorno a la solución seleccionada S_j^i, cuyo valor medio viene definido como $\delta_j^i = S_j^i$, y la desviación estándar como:

$$\sigma_j^i = \xi \sum_{r=1}^{k} \frac{\left|S_r^i - S_j^i\right|}{k-1} \tag{28}$$

La ecuación 25 calcula la desviación estándar como la distancia media entre la solución seleccionada y el resto de soluciones del archivo, multiplicada por el parámetro ξ. Este parámetro tiene un efecto similar al ratio de evaporación de feromona en ACO. Cuanto mayor sea el valor de ξ, menor será la velocidad de convergencia del algoritmo.

Después de la construcción de cada solución, la actualización de la feromona se realiza incorporando *e* soluciones nuevas y descartando las *e* peores soluciones del archivo. De esta manera siempre se mantiene l soluciones en el archivo.

4.3.3 Implementación de ACE Continuo

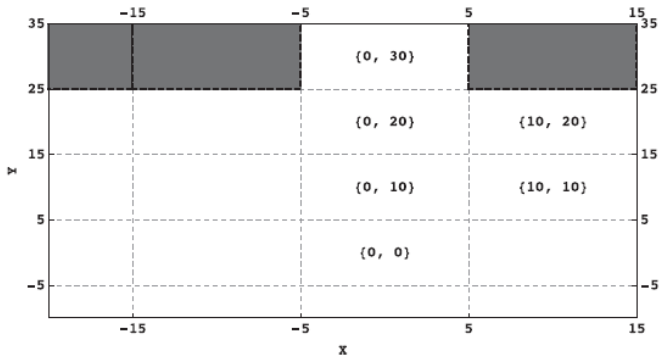
Basándonos en el planteamiento del algoritmo ACO_R, este Trabajo Fin de Máster pretende representar la información de feromona almacenada por el algoritmo ACE, mediante una función de densidad de probabilidad. Todo ello con el objetivo de permitir la optimización de las trayectorias del UAV codificadas a partir de una secuencia de acciones que toma valores continuos.

Sin embargo, la arquitectura del algoritmo ACE difiere de la del algoritmo ACO clásico, por lo que resulta necesario adaptar la metodología del algoritmo ACO_R. A continuación se describen las principales diferencias estructurales entre ambos algoritmos, así como las adaptaciones realizadas para la implementación de ACE con acciones continuas.

ACE no genera una tabla de feromonas de dimensiones fijas

El algoritmo ACE comienza con una tabla de feromonas vacía, la cual va rellenando según va realizando la búsqueda. Además, la tabla solo contiene información en aquellas celdas que han sido visitadas hasta ese momento. La siguiente figura muestra un ejemplo de una posible tabla de feromonas creada por ACE [9]. A la izquierda se representan las celdas visitadas dentro del espacio de búsqueda. Estas celdas representan la clave de la tabla de feromonas mostrada a la derecha. La información almacenada en la tabla contiene la acción tomada por el UAV en esa celda y un valor de probabilidad. Como se puede apreciar, ACE puede generar una tabla asimétrica como la de la figura, es decir, que no todas las celdas tengan el mismo número de pares acción-peso.

Las entradas de la tabla de feromonas dependen únicamente de la celda y son independientes de cómo se ha alcanzado esa celda; no se guardan las soluciones completas que hayan llevado a su registro. Además, ACE ha sido diseñado con una arquitectura asíncrona, lo que hace que no todas las hormigas terminen sus soluciones a la vez; cada hormiga va construyendo su propia solución en cada paso de iteración.



Key (x,y)	Course Value *Degree*	*Prob.*
{0,0}	144	0.338
	76	0.662
{10,10}	78	0.185
	71	0.810
{0,10}	28	0.749
	61	0.152
	42	0.09
{10,20}	337	0.574
	310	0.01
	326	0.411
{0,20}	123	0.574
	128	0.258
	108	0.167
{0,30}	97	0.09
	75	0.597
	89	0.310

Figura 34: Ejemplo de tabla de feromonas generada por ACE [9]

Por lo tanto, el algoritmo ACE para acciones continuas calcula el kernel gaussiano (ecuación 22) a partir del número de pares acción-peso presentes en ese instante en la celda donde se encuentra la hormiga (elemento de la solución que está construyendo) en ese instante. En cambio ACO_R obtiene en cada ciclo nuevas soluciones para todos los elementos de la solución; se generan tantas funciones kernel gaussianas como elementos tiene la solución.

Cabe señalar que hasta que se construya la primera solución no existe información en la tabla de feromonas de ACE y el algoritmo se guía únicamente por la información proveniente de la heurística. Esto implica que en las primeras iteraciones, la tabla de feromonas únicamente pueda presentar un solo par acción-peso en las celdas. En este caso la función de densidad de probabilidades se obtiene a partir de una función Gaussiana centrada en el valor de la acción:

$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\delta)^2}{2\sigma^2}} \qquad (29)$$

donde δ y σ representan el valor medio y la desviación estándar respectivamente. El muestreo de dicha función permite la obtención de acciones continuas.

Mantenimiento de soluciones anteriores en la tabla de feromonas

El algoritmo ACO_R actualiza la tabla de feromonas en base a la obtención soluciones de calidad, al igual que el ACO clásico. Sin embargo, en ACO la solución actual es descartada una vez que la tabla ha sido actualizada. Por el contrario, ACO_R mantiene un número de soluciones anteriores y únicamente va eliminando por ciclo aquellas que considera de menor calidad.

En el caso de ACE, éste mantiene las acciones realizadas en cada celda o estado hasta que encuentra la mejor solución obtenida hasta el momento. El contenido de la tabla de feromonas es reemplazado en mayor medida en función de λ según la ecuación 12. En base a la ecuación 13 la calidad de la mejor solución viene determinada por $\lambda = 1$, lo que lleva a actualizar los pesos del resto de acciones contenidas en esa celda a cero. Es importante remarcar que este reemplazo puede darse porque λ tenga una valor próximo a 1, pero también puede darse si se acumula el efecto de sucesivas

actualizaciones con valores menores de 1. De ahí que ACE tras una mejor solución guarda más pares acción-peso.

Algunos de los pares acción-peso eliminados podrían llevar a lograr soluciones de calidad, por lo que se ha optado por modificar el algoritmo ACE original para que mantenga un conjunto de acciones previas a la obtención de la mejor solución hasta el momento. Esto se logra ponderando la calidad de esta mejor solución a λ = 0.95. De esta manera, se consigue poner en valor la acción realizada, asignándole un peso importante, y mantener anteriores soluciones. Aquellas acciones que presenten un peso inferior a un determinado valor son eliminadas de la tabla.

Tal y como avanzábamos anteriormente, el algoritmo ACO_R, muestrea una PDF gaussiana entorno a la solución seleccionada S_j^i (solución de mayor calidad). En el caso de ACE, la calidad de las acciones viene determinada por el peso asociado; criterio seguido para la selección de la mejor acción. Al igual que en ACO_R, se muestrea la PDF gaussiana creada entono a este valor.

Heurística

Las adaptaciones realizadas para la implementación de ACE con acciones continuas también afectan a la heurística basada en el campo de potencial. En apartado 4.2 se presentaba una heurística basada en 360 acciones discretas. Esta nueva versión de ACE emplea una heurística que construye una función de densidad de probabilidad continua mediante una función gaussiana (ecuación 27), centrada (δ) en la dirección determinada por la fuerza resultante $\vec{F}_{res}(s_i^k)$. La Figura 35 muestra un ejemplo de una PDF centrada en 180º (dirección determinada por el campo de potencial).

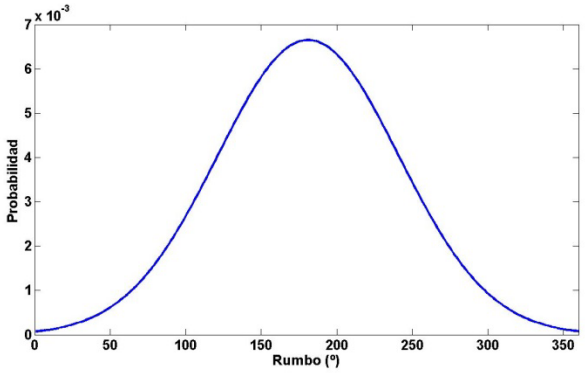


Figura 35: Distribución de probabilidad para la heurística de rumbo centrada en 180º (con una desviación estándar de 60º).

El muestreo de esta distribución gaussiana permite la obtención de acciones continuas.

Para ilustrar el funcionamiento de la versión ACE con acciones continuas, el capítulo 5 recoge los resultados obtenidos por este algoritmo sobre varios escenarios de búsqueda.

5 Estudios Experimentales

En este capítulo se describen una serie de estudios experimentales utilizando el algoritmo ACE de acciones continuas. Inicialmente, se realiza una comparativa sobre diferentes escenarios de búsqueda entre este algoritmo la versión de ACE discreto con 360 acciones diferentes presentado en el apartado 4.2. A continuación, se compara la versión de ACE de acciones continuas con otros dos algoritmos continuos como son un algoritmo genético (GA) y un algoritmo de optimización de estimación de distribuciones basado en una mezcla de gaussianas (MG), también sobre diferentes escenarios.

En ambos estudios se han utilizado escenarios con un espacio Ω cuadrado de 6x6 km^2 discretizado mediante una malla 20x20 celdas (de 300x300 m^2) y donde los UAVs vuelan a la velocidad $v_i^{c,k} = 60\mathrm{m/s}$ y a la altura $h_i^{c,k} = 300m$. En estos escenarios el tiempo de paso de la simulación es de 1 segundo y las observaciones se realizan cada 5 segundos. El sensor utilizado por el UAV se modela de forma ideal.

Los escenarios difieren en el modelo probabilístico del objetivo $\mathrm{b}(\tau^{\mathrm{k}})$, en el número y la posición inicial de los UAVs, en la presencia o no de zonas prohibidas, en el número de tramos que componen la solución y en el número N de acciones que componen cada tramo. Los escenarios con presencia de zonas prohibidas únicamente se han utilizado en la comparativa entre las dos versiones de ACE, ya que los algoritmos GA y MG no tienen una heurística que les permita evitar dichas zonas prohibidas. Los algoritmos ACE discreto con 360 acciones y ACE con acciones continuas cuentan con la heurística basada en los campos de potencial.

Los parámetros de ACE utilizados para este fin son los mostrados a continuación. Como se puede comprobar se han tomado los valores habituales de esos parámetros en [11]:

Tabla 3: Parámetros del algoritmo ACE

Parámetro	ACE
γ_1	0.9
γ_2	0.5
γ_3	∞

Además se definen los siguientes parámetros para ACE con acciones continuas:

Tabla 4: Parámetros adicionales de ACE con acciones continuas

Parámetro	ACE
ξ	0.85
σ	60

Para analizar la eficacia de los distintos algoritmos se han lanzado 20 optimizaciones de cada algoritmo en los distintos escenarios presentados. De cada conjunto de optimizaciones se guarda la mejor de las soluciones. La comparativa se realiza en función de los valores de tiempo esperado obtenidos por el algoritmo en diferentes instantes de tiempo.

En los escenarios donde la solución final presenta varios tramos, el tiempo esperado se calcula para todo el conjunto de elementos que componen la solución, desde el tramo actual a lo posteriores. Es decir, para un escenario con 2 tramos de 20 elementos cada uno, el ET del primer tramo corresponde al cálculo realizado para los 40 elementos que componen la solución completa. Dicho cálculo, sobre el primer tramo, se realiza para el peor escenario posible, que se corresponde con la no detección por parte de los UAVs durante las observaciones realizadas en el segundo conjunto de 20 elementos. Esta es la razón por la que estas gráficas presentan escalones en los valores ET calculados para los tramos consecutivos.

5.1 Comparativa entre diferentes versiones de ACE

En esta sección se realiza una comparativa entre las dos últimas versiones de ACE presentadas en este trabajo: ACE discreto con 360 acciones diferentes y ACE con acciones continuas. El estudio de realiza sobre el siguiente conjunto de escenarios:

- ***Escenario 1:*** El escenario cuenta con un solo UAV (representado mediante una estrella roja) y un objetivo estático situado en alguna de las dos zonas circulares de probabilidad. Una de estas dos zonas presenta una mayor probabilidad que la otra. La Figura 36 a) muestra la posición inicial del UAV y su dirección de movimiento. Sobre este escenario, se plantea la optimización de trayectorias de 20 pasos en dos tramos diferentes.
- ***Escenario 2:*** El escenario cuenta con dos UAVs (representados mediante una estrella roja) y un objetivo estático situado en alguna de las dos zonas de probabilidad (Figura 36 b)). Una de estas dos zonas presenta una mayor probabilidad que la otra. Además este escenario presenta dos zonas de vuelo prohibidas (rectángulos en negro) a medio camino entre los UAVs y las zonas de máxima probabilidad. Sobre este escenario, se plantea la optimización de trayectorias de 15 pasos en tres tramos diferentes.
- ***Escenario 3:*** El escenario cuenta con un solo UAV (representado mediante una estrella roja) y un objetivo estático situado en alguna de la zona rectangular de probabilidad. La Figura 36 c) muestra la posición inicial del UAV y su dirección de movimiento. Sobre este escenario, se plantea la optimización de trayectorias de 25 pasos en un único tramo.

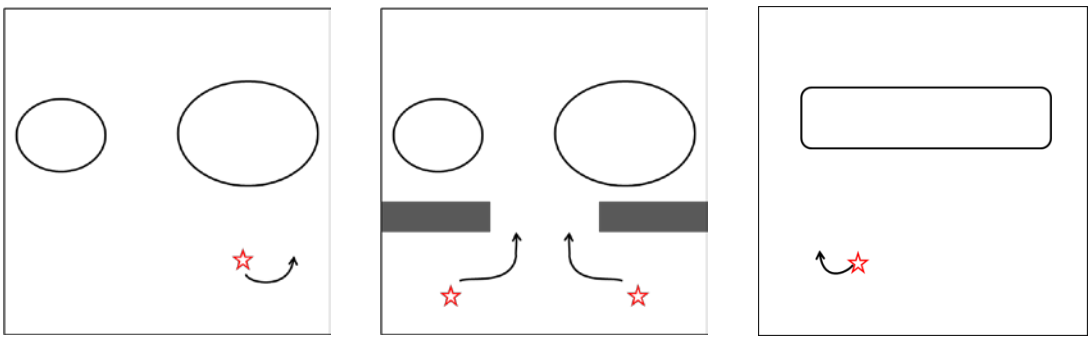


Figura 36: Escenarios utilizados en la comparativa entre ACE con 360 acciones discretas y ACE con acciones continuas a) Escenario 1 b) Escenario 2 c) Escenario 3

Los resultados obtenidos sobre cada uno de los escenarios se muestran a continuación. Los resultados se presentan siguiendo una misma estructura de representación. En la fila superior de las figuras (Figura 37, Figura 38 y Figura 39) se muestran el modelo probabilístico $b(\tau^0)$ del escenario en el instante inicial (imagen de la izquierda) y la evolución de los mejores valores de la función objetivo,

obtenidos por el algoritmo ACE con 360 acciones discretas diferentes y ACE con acciones continuas (imagen derecha). En la gráfica se representa la evolución del valor medio y la varianza, calculados sobre las 20 optimizaciones realizadas con ambos algoritmos, de los valores de la función objetivo (tiempo esperado) obtenidos por los algoritmos en diferentes instantes de tiempo.

Por otro lado, la segunda fila muestra la mejor solución (mínimo tiempo esperado) obtenida entre todas las calculadas por ACE con 360 acciones discretas (izquierda) y ACE con acciones continuas (derecha). Se muestran las trayectorias seguidas por los UAVs desde el punto inicial a las zonas de máxima probabilidad. Además, se indica el valor de tiempo esperado obtenido por dicha solución para cada uno de los algoritmos.

Resultados Escenario 1

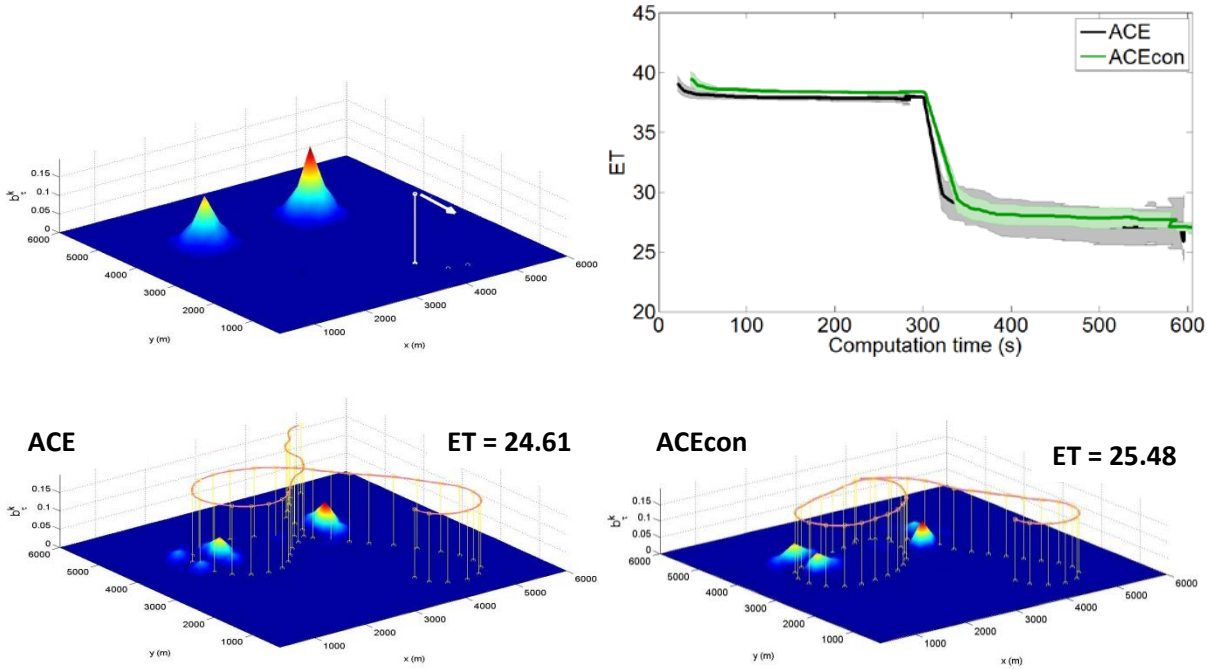


Figura 37: Resultados sobre el escenario 1 de ACE con 360 acciones discretas diferentes y ACE con acciones continuas

Como se puede ver en las figuras de la segunda fila, a pesar de partir con una orientación opuesta a la dirección donde se encuentran las zonas de máxima probabilidad, ambos UAVs realizan un giro para dirigirse a la zona de máxima probabilidad. Esto es debido a que la heurística basada en el campo de potencial guía a los UAVs hacia dicha zona de máxima probabilidad. Una vez recogida parte de la probabilidad de esa zona, se dirigen hacia la segunda zona de probabilidad.

En la fila superior se representa la evolución del valor medio y la varianza del tiempo esperado para ambos algoritmos, calculados sobre las 20 optimizaciones. Ambas curvas de valor medio dibujan una rápida convergencia hacia la solución final.

Resultados Escenario 2

Además de los obstáculos estáticos mostrados en la Figura 38, los UAVs entre ellos deben mantener una distancia de seguridad en vuelo. Dicha restricción se modelada mediante una fuerza de repulsión entre los UAVS. En el momento de calcular la fuerza resultante para uno de los UAV, el resto de la

flota es considerada como obstáculos, ejerciendo un campo de repulsión con el fin de evitar las colisiones entre ellos.

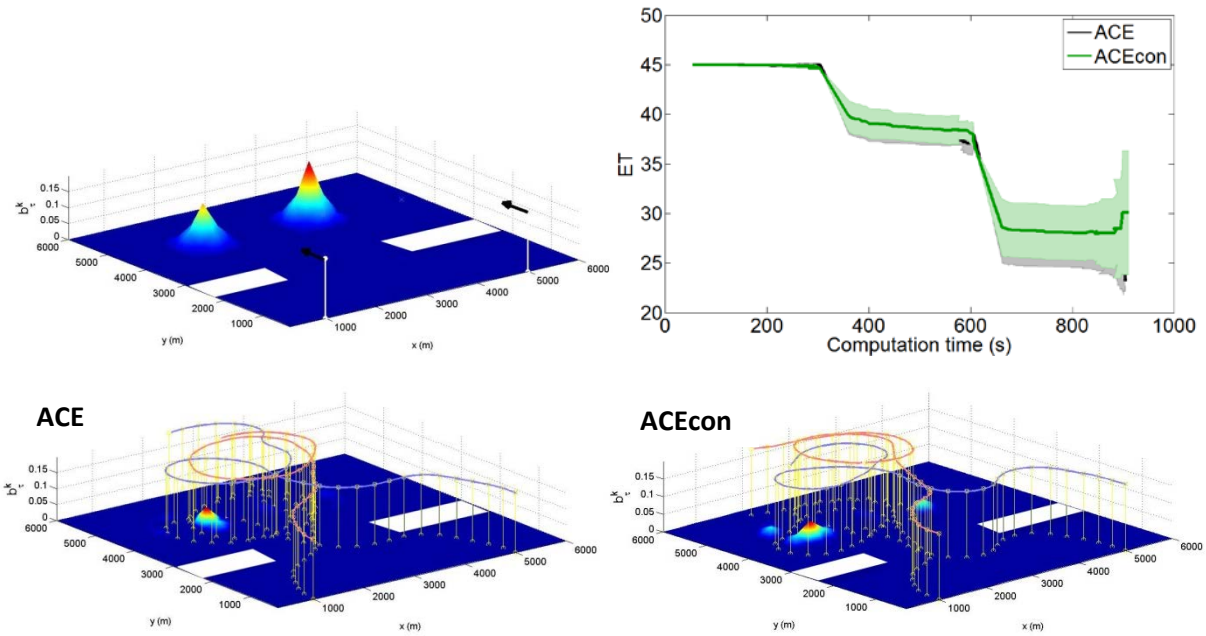


Figura 38 : Resultados sobre el escenario 2 de ACE con 360 acciones discretas diferentes y ACE con acciones continuas

Como se puede observar en la figura, los dos algoritmos, con la ayuda de la heurística basada en el campo de potencial, consiguen evitar las dos zonas prohibidas de vuelo. Todo ello a pesar de que ambos UAVS deben sobrevolar el mismo pasillo central, a la vez que ejercen una fuerza de repulsión mutua. Una vez sobrepasado los obstáculos los UAVs van recogiendo la probabilidad de las zonas de máxima probabilidad, alternado de una zona a la otra.

Resultados Escenario 3

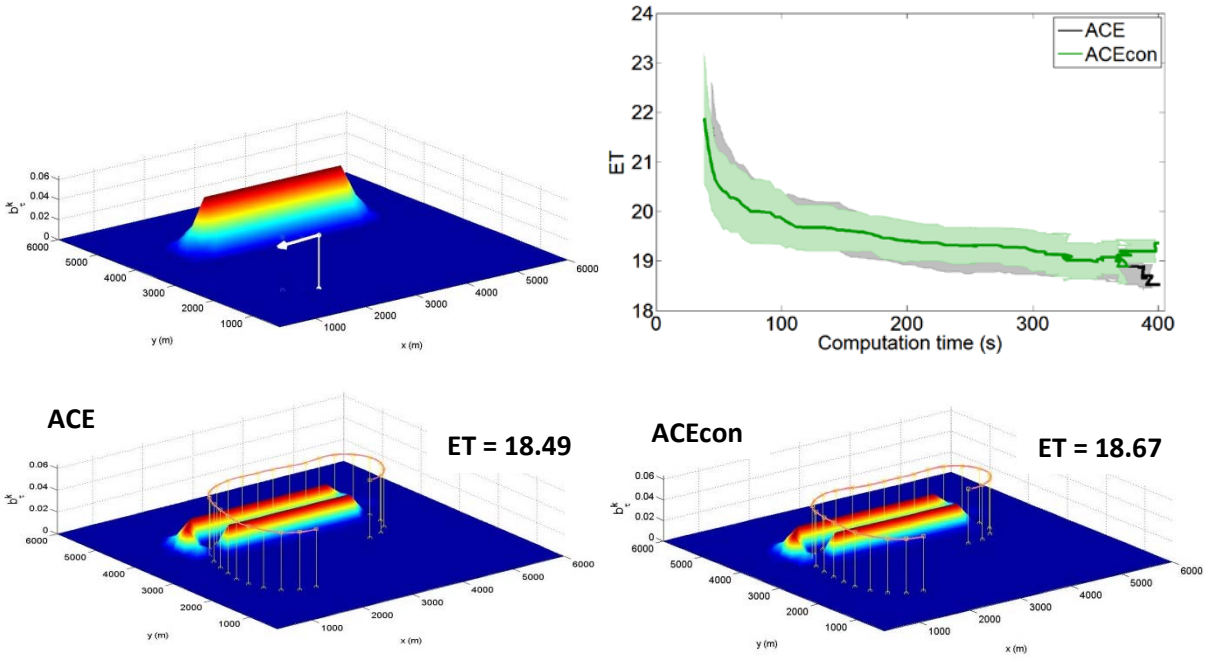


Figura 39: Resultados sobre el escenario 1 de ACE con 360 acciones discretas diferentes y ACE con acciones continuas

Una vez más, los dos algoritmos comparados en este estudio obtienen resultados muy similares. De hecho, tal y como ocurría en los escenarios anteriores, la gráfica que dibuja la evolución del tiempo esperado muestra como como ambas curvas se sobreponen. Ambos convergen rápidamente hacia una solución parecida.

5.2 Comparativa con otros algoritmos

En esta sección se realiza una comparativa entre la versión de ACE con acciones continuas y otros algoritmos continuos como son GA y MG.

5.2.1 Algoritmos utilizados

Los algoritmos de optimización elegidos para realizar la comparativa que se recoge en este TFM son dos: un algoritmo genético (GA) y un algoritmo de estimación de distribución (EDA) cuyo soporte estadístico es una mezcla de gaussianas (MG). Sus características más relevantes se describen a continuación.

- El algoritmo genético genera inicialmente una población de S_{GA} soluciones, en las que se codifica, al igual que en la última variante de ACE, la secuencia de acciones de control (ángulos de guiñada deseados) que se aplicarán a los UAVs. A continuación, selecciona $S_{GA}/2$ parejas de padres utilizando el método de torneo, mezcla los valores de un porcentaje (p_c) de las pareja utilizando el cruce por un punto de la secuencia y modifica los valores de las nuevas soluciones utilizando una mutación incremental gaussiana doble, en el que se modifican todos los valores de la secuencia ligeramente (σ_p) y un porcentaje (p_m) de valores de la secuencia un poco más (σ_g). A continuación se eligen, como punto de partida de una nueva iteración del algoritmo, entre las nuevas soluciones que se han creado y las previas, las S_{GA} mejores (i.e. las que tienes menor ET). El algoritmo termina una vez que ha pasado su tiempo prefijado de evolución, devolviendo como solución la mejor encontrada hasta ese momento.
- El segundo algoritmo, denominado MG por su soporte estadístico, genera una población de S_{MG} soluciones, en las que se codifica otra vez la secuencia de acciones de control (ángulos de guiñada deseados) que se aplicarán a los UAVs. A continuación, selecciona las $S_{MG}/4$ soluciones mejores (con menor ET) y las utiliza para aprender la mezcla de gaussianas que mejor se ajusta a las soluciones elegidas según el criterio de Akaike. A continuación, utiliza la mezcla de gaussianas recién aprendidas para muestrear S_{MG} soluciones nuevas, que serán el punto de partida de la siguiente iteración del algoritmo. Al igual que en el caso anterior, el algoritmo termina una vez que ha pasado su tiempo prefijado de evolución, devolviendo como solución la mejor encontrada hasta ese momento.

Los parámetros de cada algoritmo se recogen en Tabla 5. El tamaño de la población de MG es 8 veces que el de GA, ya que el método utilizado para aprendizaje de la mezcla de gaussianas que se encuentra implementado en Matlab requiere que el número de soluciones usados ($S_{MG}/4 = 2*M*N$) sea superior al número de variables de la mezcla (M*N).

Es importante destacar que ninguno de los dos algoritmos incluye un mecanismo para rechazar/penalizar las soluciones que sobrevuelan las zonas prohibidas, motivo por el que la

comparativa con estos algoritmos únicamente se realizará sobre escenarios en los que estas zonas no existan.

Tabla 5: Parámetros de los algoritmos GA (izquierda) y MG (derecha)

$S_{GA} = M*N$	$S_{MG} = 8*M*N$
$p_c = 80\%$	
$p_m = 1/(M*N)$	
$\sigma_p = 0.1;$	
$\sigma_g=0.5;$	

Finalmente, destacar que los algoritmos elegidos para la comparativa de ACE comparten propiedades similares a los utilizados previamente en [23] para comparar el funcionamiento del ACO descrito en [24]. GA es un algoritmo que busca nuevas soluciones manipulando directamente las ya existentes mientras que MG las genera a partir de la distribución que más se ajusta a las mejores soluciones obtenidas hasta el momento. Además, como MG comparte con el ACE continuo el soporte probabilístico (mezcla de gaussianas vs. kernel de gaussianas), ambos generan soluciones a partir de una función de distribución similar, aunque el procedimiento para adquirirla sea diferente y ACE aproveche las ventajas de utilizar una heurística propia del problema MTS

5.2.2 Resultados

A diferencia de lo que ocurre con el algoritmo ACE, GA y MG no presentan ningún método que permita al algoritmo determinar si se está sobrevolando una zona prohibida, motivo por el que los escenarios seleccionados para este último análisis, no contienen zonas de vuelos prohibidas. Los escenarios escogidos son los siguientes:

- ***Escenario 1:*** El escenario cuenta con un solo UAV (representado mediante una estrella roja) y un objetivo estático situado en alguna de las dos zonas circulares de probabilidad. Una de estas dos zonas presenta una mayor probabilidad que la otra. La Figura 40 a) muestra la posición inicial del UAV y su dirección de movimiento. Sobre este escenario, se plantea la optimización de trayectorias de 20 pasos en dos tramos diferentes.
- ***Escenario 2:*** El escenario cuenta con dos UAVs (representados mediante una estrella roja) y un objetivo estático situado en alguna de las dos zonas de probabilidad (*Figura 40* b)). Una de estas dos zonas presenta una mayor probabilidad que la otra. Sobre este escenario, se plantea la optimización de trayectorias de 25 pasos en dos tramos diferentes.
- ***Escenario 3:*** El escenario cuenta con un solo UAV (representado mediante una estrella roja) y un objetivo estático situado en alguna de las zonas rectangular de probabilidad. La Figura 40 c) muestra la posición inicial del UAV y su dirección de movimiento. Sobre este escenario, se plantea la optimización de trayectorias de 25 pasos en un único tramo.

Los resultados obtenidos sobre cada uno de los escenarios se muestran a continuación. Los resultados se presentan siguiendo una misma estructura de representación. En la fila superior de las figuras se muestran la evolución de los mejores valores de la función objetivo (imagen derecha), obtenidos por los distintos algoritmos. En la gráfica se representa la evolución del valor medio y la varianza, calculados sobre las 20 optimizaciones realizadas con los distintos

algoritmos, de los valores de la función objetivo (tiempo esperado) obtenidos por los algoritmos en diferentes instantes de tiempo.

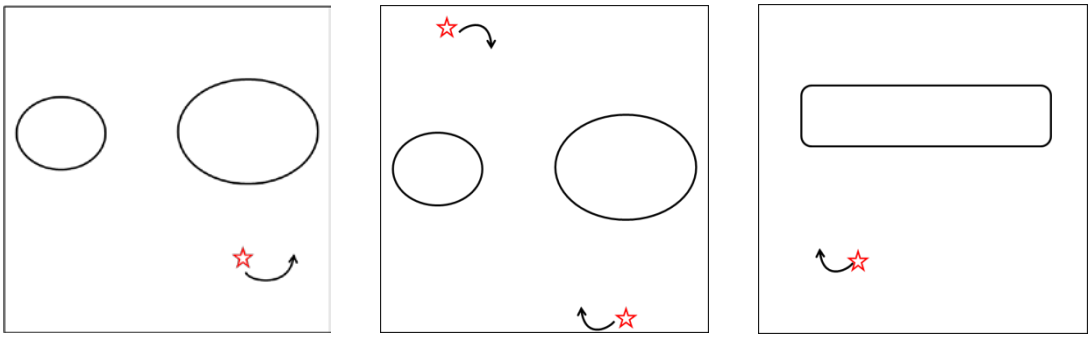


Figura 40: Escenarios utilizados en la comparativa entre ACE, GA y MG a) Escenario 1 b) Escenario 2 c) Escenario 3

Por otro lado, la segunda fila muestra el modelo probabilístico $b(\tau^0)$ del escenario en el instante inicial (imagen de la izquierda) y la mejor solución (mínimo tiempo esperado) obtenida entre todas las calculadas por ACE con acciones continuas (imagen derecha). En la última fila se pueden observar las mejores soluciones obtenidas entre todas las calculadas para los algoritmos GA (izquierda) y MG (derecha). Se muestran las trayectorias seguidas por los UAVs desde el punto inicial a las zonas de máxima probabilidad. Además, se indica el valor de tiempo esperado obtenido por dicha solución para cada uno de los algoritmos.

Resultados Escenario 1

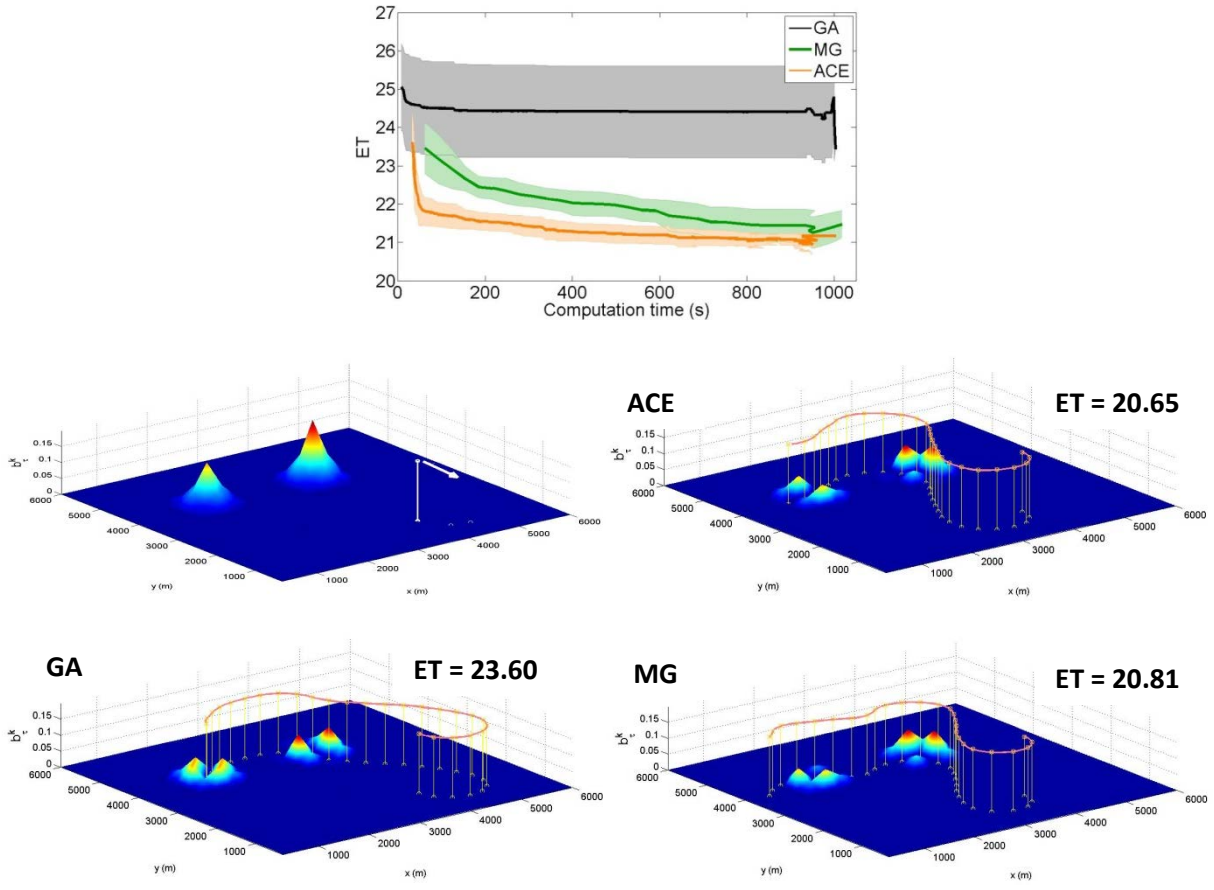


Figura 41: Resultados sobre el escenario 1 de ACE con acciones continuas, GA y MG

Como se puede apreciar en la imagen superior el algoritmo ACE converge rápidamente hacia la solución. De hecho, al cabo de 200s se encuentra próximo a alcanzarla. Por su parte, el algoritmo MG también converge, aunque de forma más lenta, hacia la misma solución que ACE con acciones continuas. A diferencia de este último, MG requiere de cerca de 1000s para alcanzar la solución.

Por su parte, el algoritmo GA converge hacia un valor muy superior al obtenido por ACE y MG. Además la evolución del tiempo esperado presenta mucha dispersión en las 20 optimizaciones realizadas por este algoritmo algoritmos. Ello podría indicar que algunas de las soluciones calculadas por este algoritmo no alcanzaron las zonas de máxima probabilidad

Resultados Escenario 2

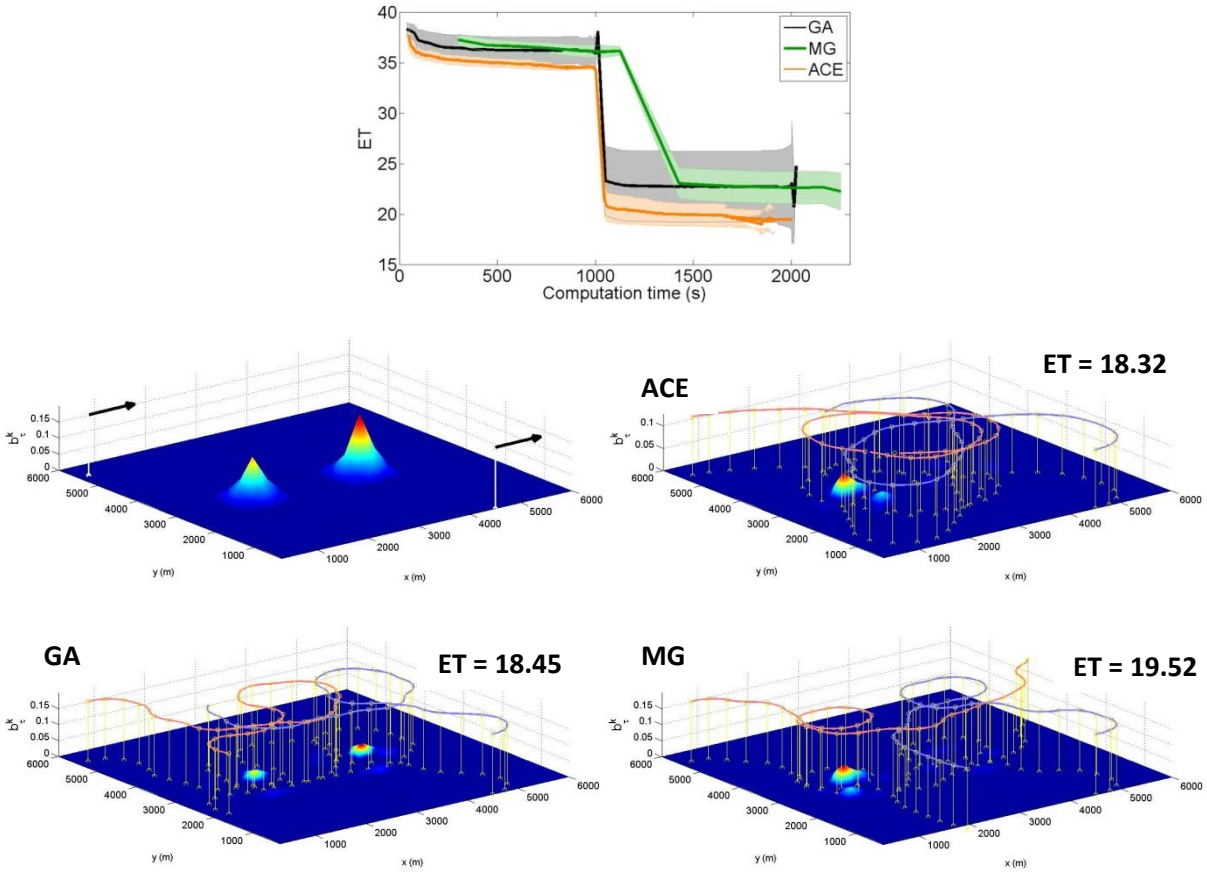


Figura 42: Resultados sobre el escenario 2 de ACE con acciones continuas, GA y MG

Analizando las 3 trayectorias definidas por las mejores soluciones de cada algoritmo, se puede apreciar como en el caso de los algoritmos GA y MG, los UAVs se dirigen a las zonas de probabilidad más cercana a su posición inicial. Inspeccionando de esta manera, durante el primer tramo de la solución, cada agente una zona de probabilidad diferente. A continuación, en el segundo tramo, cada agente pasa a inspección la zona dejada por el otro.

A diferencia de los otros dos algoritmos continuos, ACE empujado por la heurística, desplaza a los dos UAVs hacia la zona de máxima probabilidad. Y solo cuando ambos han recogido parte de la probabilidad acumulada en esta zona, se desplazan juntos a la segunda zona. La comparativa de las tres trayectorias también muestra que, mientras las trayectorias de ACE se componen de círculos

entornos a las zonas de máxima probabilidad, las obtenidas por los otros dos algoritmos resultan más caóticas.

La gráfica comparativa muestra como ACE logra una mejor solución que el resto. Sin embargo, en este caso la diferencia entre los algoritmos, en cuanto a calidad de solución, no es tan grande como en el escenario 1. A pesar de todo, GA sigue mostrando la dispersión más elevada de entre los 3 algoritmos

Resultados Escenario 3

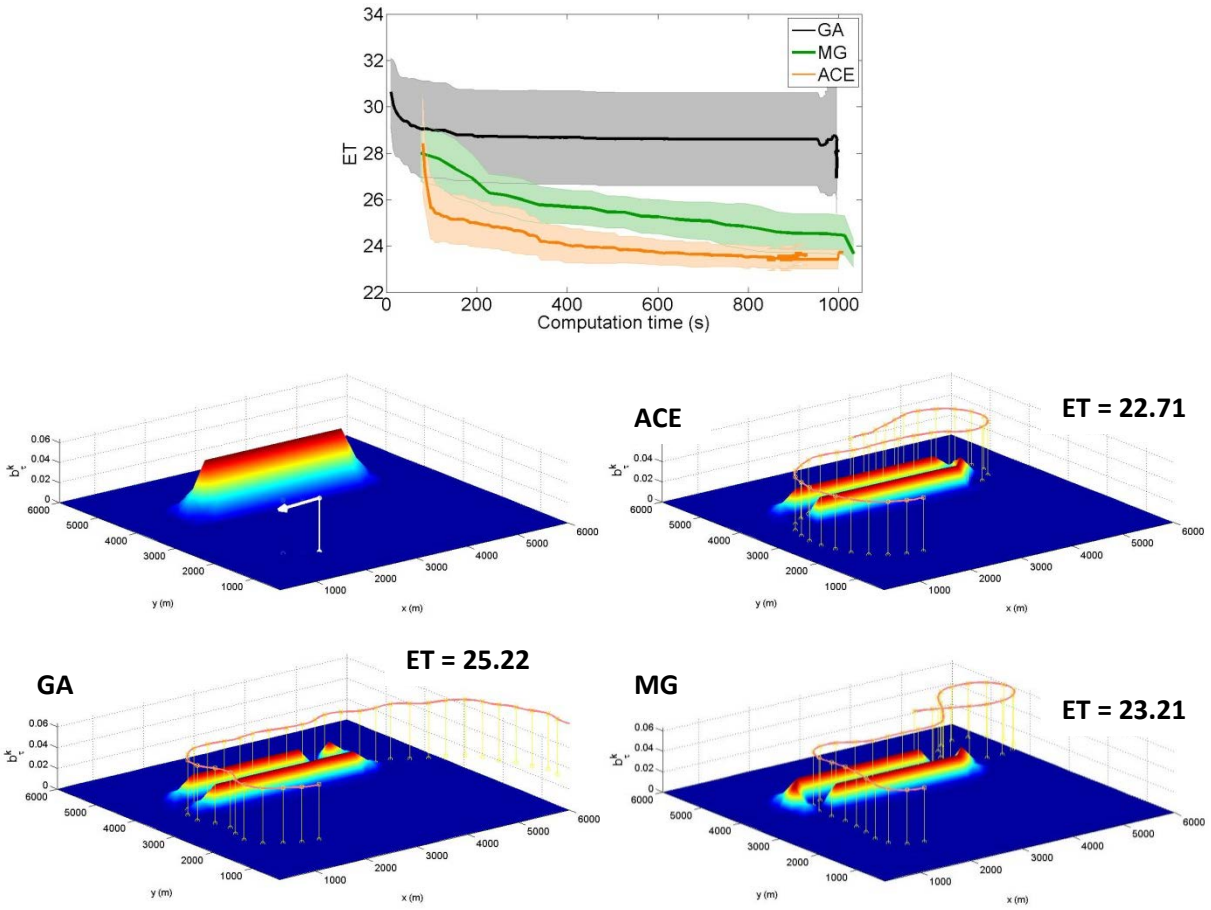


Figura 43: Resultados sobre el escenario 3 de ACE con acciones continuas, GA y MG

La Figura 42 muestra un resultado similar al obtenido sobre el escenario 1. El algoritmo ACE con acciones continuas obtiene soluciones muy buenas en los instantes iniciales. Además la posterior convergencia también resulta más rápida que en el resto de algoritmos. Nuevamente el algoritmo MG alcanza una solución de calidad cercana (tiempo esperado) a la de ACE, al cabo de los 100s que dura la simulación.

Como en los casos anteriores, el algoritmo GA vuelve a mostrar una dispersión mayor entre las 20 optimizaciones realizadas. Además el valor medio de dichas optimizaciones queda muy por encima del obtenido por ACE y MG. Analizando la trayectoria de la mejor solución obtenida por GA, se aprecia cómo tras recorrer a la largo la zona de probabilidad, un tercio de la solución apenas recoge probabilidad, ya que se desplaza en la dirección opuesta a la zona.

6 Conclusiones

Finalmente, en este capítulo, resumimos las conclusiones más relevantes de los algoritmos y análisis desarrollado durante este Trabajo Fin de Master y presentamos algunas de las líneas de trabajo que podrían seguirse en un futuro próximo.

6.1 Conclusiones

En este TFM se presenta una adaptación del algoritmo Ant Colony Extended para la resolución del problema de búsqueda en tiempo mínimo (MTS). En concreto, se diseña un planificador para múltiples UAVs con el fin de encontrar un objetivo de ubicación desconocida en mínimo tiempo. Este planificador debe determinar la secuencia de acciones que permite a los agentes desplazarse por la región de búsqueda mientras realizan observaciones.

En este sentido, a partir del algoritmo ACE se han desarrollado dos tipos de planificadores diferentes, uno para el problema MTS discreto con 360 acciones y otro para el problema MTS con acciones continuas. Esta última versión, realizada en base al planteamiento del algoritmo ACOR, permite la optimización de las trayectorias del UAV codificadas a partir de una secuencia de acciones que toma valores continuos. Se ha conseguido diseñar un algoritmo que trabaja con acciones continuas manteniendo la arquitectura de ACE original.

Además en este trabajo también se ha diseñado una heurística de rumbo basada en los campos de potencial, que permite al UAV trazar rutas hacia las zonas de máxima probabilidad de detección del objetivo capaz de esquivar zonas prohibidas de vuelo y evitar la colisión de unos UAVs con otros.

La eficacia de ambos algoritmos junto con la heurística basada en campo de potencial ha sido comprobada frente a varios escenarios, con objetivos fijos, distintos número de agentes e incluso presencia de zonas de vuelo prohibidas. En primer lugar se ha realizado una comparativa de los resultados obtenidos mediante las dos versiones de ACE: con 360 acciones discreta y con acciones continuas. Los resultados obtenidos para ambos algoritmos son muy similares en todos los escenarios probados.

También se ha comparado la versión de ACE con acciones continuas con otros algoritmos continuos, sobre diferentes escenarios. A diferencia de ACE, estos algoritmos no presentan ningún método que les permita determinar la existencia de zonas prohibidas, por lo que los escenarios seleccionados en este análisis no presentaban dichas zonas. El estudio realizado muestra que ACE tiene una mayor rapidez de convergencia y presenta soluciones de mayor calidad que los otros dos métodos. Aún más, en los primeros instantes de tiempo del algoritmo ya obtiene soluciones de calidad, debido al diseño de la heurística diseñada para el problema de tiempo mínimo. Destaca a su vez la poca dispersión existente entre las 20 optimizaciones realizadas en comparación por ejemplo con el algoritmo GA.

6.2 Trabajos Futuros

En el estudio realizado se pueden distinguir las siguientes vías de investigación futura:

- Estudiar la influencia de los parámetros de ACE con 360 acciones discretas y ACE con acciones continuas en los resultados de los algoritmos. En especial, sería interesante realizar dicho análisis para la versión con acciones continua ya que incorpora nuevos parámetros a los ya utilizados por el algoritmo ACE original.
- Mejora de la heurística basada en el campo de potencial con el fin conseguir una búsqueda más eficiente. La heurística debería dispersas más la búsqueda entre los distintos UAVs de la misión. Por ejemplo incluyendo un factor dependiente de la distancia entre la posición de los UAVs y las zonas de máxima probabilidad. Además resulta necesario estudiar la influencia de los parámetros del campo de potencial con el fin de lograr una adecuada evasión de obstáculos.
- Adaptación al problema MTS con objetivos dinámicos. En este trabajo se ha abordado el cálculo de trayectorias para objetivos estáticos, esto es, para el caso en el que la posición del objetivo no varía respecto al tiempo. En muchas aplicaciones los objetivos son dinámicos por lo que resulta necesario abordar este tipo de escenarios. Además la heurística diseñada en este trabajo permite ya la consideración de objetivos dinámicos, motivo por el que en una primera aproximación se podría analizar el funcionamiento de las técnicas ya diseñadas sobre escenarios con objetos dinámicos.

Bibliografía

[1] Bilchev B, Parmee IC. *The ant colony metaphor for searching continuous design spaces*. In: Fogarty TC, editor. Proceedings of the AISB workshop on evolutionary computation. Lecture Notes in Comput Sci, vol. 993. Berlin: Springer; 1995. p. 25–39.

[2] Blum C., *Ant colony optimization: Introduction and recent trends*, Physics of Life Reviews 2 (2005) 353–373.

[3] Bonabeau E, Dorigo M, Theraulaz G. Inspiration for optimization from social insect behavior. Nature 2000;406:39–42.

[4] Canny J, Reif J, New lower bound techniques for robot motion planning problems. Proceedings of IEEE Symposium on the Foundations of Computer Science, Los Angeles, California, pp. 49-60. (1987)

[5] Dorigo, M., 1992. Optimization, Learning and Natural Algorithms. Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy.

[6] Dorigo, M., Gambardella, L.M., 1997. Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66.

[7] Dorigo M., Stützle T., *Ant Colony Optimization (Bradford Books)*. The MIT Press, June 2004.

[8] Dréo J, Siarry P. A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions. In: Dorigo M, Di Caro G, Sampels M, editors. Ant algorithms—Proceedings of ANTS 2002—Third international workshop. Lecture Notes in Comput Sci, vol. 2463. Berlin: Springer; 2002. p. 216–21.

[9] Escario J. B., Jiménez J. F., Girón-Sierra J. M., *Optimisation of autonomous ship manoeuvres applying ant colony optimisation metaheuristic*, Expert Systems with Applications 39 (11) (2012) 10120 – 10139. doi:10.1016/j.eswa.2012.02.069.

[10] Escario J. B., Jiménez J. F., Girón-Sierra J. M., *Ant colony extended: Experiments on the travelling salesman problem*. Expert Systems with Applications, 42(1):390–410, 2015.

[11] Escario J. B., *Computación emergente y auto-organización aplicada al diseño de algoritmos bio-inspirados de búsqueda heurística*. Tesis doctoral, Universidad Complutense de Madrid, España.

[12] Gambardella L. M., Dorigo M., *Ant Colony System hybridized with a new local search for the sequential ordering problema*. INFORMS J Comput 2000; 12(3):237-55.

[13] Gambardella LM, Taillard ÉD, Agazzi G. *MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows*. In: Corne D, Dorigo M, Glover F, editors. New ideas in optimization. London: McGraw-Hill; 1999. p. 63–76.

[14] Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by simulated annealing. Science, 220, 671.

[15] Koren Y., Borenstein J., *Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation*, Proceedings of the IEEE Conference on Robotics and Automation, Sacramento, California, April 7-12, 1991, pp. 1398-1404.

[16] 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).

[17] Lanillos, P., Javi, Y.-Z., Ruz, J. J., Besada-Portas, E., *A Bayesian approach for constrained multi-agent minimum time search in uncertain dynamic domains*. In Genetic and Evolutionary Computation (GECCO) Conference. (2013).

[18] Lanillos Pradas P., *Búsqueda de objetivos móviles en tiempo mínimo sobre entornos con incertidumbre*. Tesis Doctoral, Universidad Complutense de Madrid, España, 2013.

[19] Latombe, J. C., *Robot Motion Planning*. Kluwer Academic Press. (1990).

[20] La Valle, S. (2006). *Planning algorithms*. Cambridge University Press.

[21] Liao, T., de Oca, M., Aydn, D., Stützle, T., Dorigo, M., *An incremental ant colony algorithm with local search for continuous optimization*. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011). New York (2011).

[22] Maniezzo V, Boschetti M, Jelasity M. An ant approach to membership overlay design. In: Dorigo M, Birattari M, Blum C, Gambardella LM, Mondada F, Stützle T, editors. Proceedings of ANTS 2004—Fourth international workshop on Ant colony optimization and swarm intelligence. Lecture Notes in Comput Sci, vol. 3172. Berlin: Springer; 2004. p. 37–48.

[23] Pérez Carabaza S., Besada Portas E., López Orozco J.A., de la Cruz J.M, *Informe técnico interno para AIRBUS, comparativa de algoritmos discretos para MTS*.

[24] Pérez Carabaza S., Besada Portas E., López Orozco J.A., de la Cruz J.M., *Resolución del problema de Búsqueda en Tiempo Mínimo mediante Colonias de Hormigas*, Actas de las XXXVI Jornadas de Automática, 2 - 4 de septiembre de 2015. Bilbao.

[25] Pérez Carabaza S., Besada Portas E., López Orozco J.A., de la Cruz J.M, *A Real World Multi-UAV Evolutionary Planner for Minimum Time Target Detection.*

[26] Socha K., Dorigo M., *Ant colony optimization for continuous domains*, European Journal of Operational Research 185 (2008) 1155–1173.

[27] Stone, L. D. (1975). *Theory of optimal search*. Academic Press, New York.

[28] Stützle, T., Hoos, H.H., *MAX–MIN Ant System*. Future Generation Computer Systems 16 (8), 889–914, 2000.

[29] Sugihara K., Smith J., *Genetic algorithms for adaptive motion planning of an autonomous mobile robot*. Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, California, pp. 138-143. (1997).

[30] Yáñez Zuluaga F. J., *Técnicas heurísticas para la búsqueda en tiempo mínimo (comparativa)*, Trabajo Fin de Máster. Tutor: Eva Besada Portas y Pablo Lanillos Pradas. Fecha de presentación: 18/9/2014.

Lista de figuras

Lista de tablas

Listado siglas, abreviaturas y acrónimos

ACE: Extensión del algoritmo de colonia de hormigas (Ant Colony Extended)

ACO: Optimización por colonias de hormigas (Ant Colony Optimization)

ACO_R: Extensión de ACO a entornos continuos (ACO extended to continuous domains)

CO: Optimización combinatorial (Combinatorial Optimization)

GA: Algoritmos Genéticos (Genetic Algorithms)

ET: Tiempo esperado (Expected Time)

MG: Mezcla de Gaussianas (Gaussians mixture)

MTS: Búsqueda en Tiempo Mínimo (Minimum Time Search)

PDF: función de densidad de probabilidad (Probability Density Function)

UAV: Vehículo aéreo no tripulado (Unmanned Aerial Vehicle)