Universidad Politécnica de Cartagena

APLICACIÓN DE ALGORITMOS EPSO Y SVM A LA INTELIGENCIA DE ENJAMBRES DE RPAS EN MISIONES DE SATURACIÓN DE DEFENSAS E ISR

Trabajo Final de Máster

Premios Ejército del Aire 2019

Modalidad Investigación Aeroespacial Universitaria

Curso: 2017/2018

El autor certifica que este trabajo de investigación es una versión redactada fiel al original del Trabajo Fin de Máster "Aplicación de algoritmos EPSO y SVM a la inteligencia de enjambres de RPAs en misiones de saturación de defensas E ISR". Este trabajo de investigación es un Trabajo Fin de Máster, de la Universidad Politécnica de Cartagena, cursado en el curso académico 2017/2018, y defendido ante un tribunal académico en el año 2018. La presente versión ha sido adaptada ligeramente en su formato y extensión, con la única finalidad de ajustarse a las condiciones indicadas en la convocatoria (letra arial tipo 12, interlineado 1 ½), habiendo eliminado cualquier referencia al nombre del autor, agradecimientos, felicitaciones y dedicatoria, garantizando así su anonimato, con el objeto de poder ser presentada a los premios del Ejército del Aire 2019, en su modalidad Investigación Aeroespacial Universitaria. Es en todo lo demás, una reproducción fiel del original.

Asimismo, este trabajo de investigación tiene una extensión máxima de ciento veinte (120) páginas tamaño DIN A4, teniendo en cuenta que el contenido comprendido entre la primera página y la vii, páginas 7, 16, 23, 27, 39, 45, 63, 70, 76, 86, 99, 105 y 125, y páginas 133 a 244, se corresponden con diverso contenido no relevante, como páginas dejadas intencionadamente en blanco, apéndices, anexos y bibliografía.

APLICACIÓN DE ALGORITMOS EPSO Y SVM A LA INTELIGENCIA DE ENJAMBRES DE RPAS EN MISIONES DE SATURACIÓN DE DEFENSAS E ISR

RESUMEN: Mediante la aplicación de una serie de algoritmos que comprenden un EPSO de optimización evolutiva y búsqueda de estados en combinación a una SVM de aprendizaje automático, se define una máquina de inteligencia artificial que explota el concepto de coordinación, cooperación y colaboración, con alto carácter de reciprocidad y ayuda mutua, entre RPAs de un enjambre, capaz de tomar la mejor decisión entre varias posibilidades, en misiones operativas de saturación de defensas e ISR, basándose en nuevos conceptos desarrollados específicamente para este trabajo como la entropía del enjambre y tecnologías avanzadas de elusión de colisiones.

ABSTRACT: Applying a set of algorithms related to an EPSO of evolutionary optimization and search of states in combination with a machine learning algorithm SVM, it is defined an artificial intelligence tool which makes use of the concept of coordination, cooperation and collaboration, with a high character of reciprocity and mutual aid, among RPAs of a swarm, able to make the best decision among several possibilities, in operational missions of air defense saturation and ISR, based on new concepts developed specifically for this work as the entropy of a swarm and advanced technologies of collision avoidance.

Agradecimientos

Contenido

Lista de Acrónimos

ACM	Computer Chess Championships
AgI	Agentes inteligentes
AHLoS	Ad-Hoc Localization System
AoA	Angle of Arrival
APIT	Approximate Point In Triangulation
APS	Ad-hoc Positioning System
ASW	Anti-Submarine Warfare
CC	Campos computacionales
CODE	Collaborative Operations in Denied Environment
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
DIC	Digital Image Correlation
DoD	Department of Defense
EMOE	Electromagnetic Operational Environment
EO	Electroóptica
EPSO	Evolutionary Particle Swarm Optimization
ERM	Empirical Risk Minimization
EW	Electronic Warfare

GA	Genetic Algorithm
GCS	Ground Control Station
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GSM	Global System for Mobile
I2R	Imaging Infrared
IR	Infrarroja
ISR	Intelligence, Surveillance and Reconnaisance
ISTAR	Intelligence, Surveillance Target, Adquisition and Reconnaisance
KAIST	Korea Advanced Institute of Science and Technology
KKT	Karush-Kuhn-Tucker
KS	Kolmogorov-Smirnov
LCCM	Low Cost Cruise Missile
LIDAR	Laser/Light Imaging Detection And Ranging
LOCUST	Low-Cost UAV Swarming Technology
LOS	Line-Of-Sight
MANET	Mobile and Ad Hoc Wireless Networks
MDS	MultiDimenSional
ML	Machine Learning
MMW	Millimeter Wave
MPA	Maritime Patrol Aircraft
MS	Maestro y subordinados
NAVAIR	Naval Air Systems Command
ONR	Office of Naval Research

PSO	Particle Swarm Optimization
RBF	Gaussian Radial Basis Function
RCS	Radar Cross-Section
RPAs	Remotely Piloted Aircraft System
RSSI	Received Signal Strenght Indicator
S&A	Sense & Avoid
SAASM	Selective Availability Anti-Spoofing Module
SAM	Surface to Air Missile
SCO	Strategic Capabilities Office
SEAD	Suppression of Enemy Air Defenses
SRM	Structural Risk Minimization
SVM	Support Vector Machine
TDoA	Time Difference of Arrival
TE	Trabajo en equipo
ToA	Time of Arrival
USAF	United States Air Force
USN	United States Navy
UV	Ultravioleta
VC	Variables de consenso

Capítulo 1. Introducción

Los enjambres de RPAs constituyen un proyecto prometedor e innovador, que requiere del desarrollo de diversas tecnologías punteras, con el fin anticipar y predecir el comportamiento de un grupo de RPAs, comunicándose de forma segura entre ellos, detectando y evitando obstáculos en tiempo real, reconfigurando la trayectoria de estos RPAs de forma automática, todo ello con el objetivo de cumplir misiones operativas de alto nivel.

Estas aeronaves remotas actuarán volando en escuadrillas, donde el piloto del enjambre coordinará a todo el grupo como un conjunto, descargándose de la ardua tarea de pilotar cada uno de los vehículos remotos de forma individual; por otra parte, misión prácticamente imposible de ejecutar por una persona si el número de RPAs resultará muy numeroso. Para ello, una tecnología clave que apoyaría el desarrollo de estos enjambres será la inteligencia artificial, entendida como la capacidad de una máquina para tomar la mejor decisión entre varias posibilidades.

El primer problema que surge en los enjambres de RPAs es el de las colisiones, el cual se acentúa en la medida que aumenta el número de vehículos remotos que componen el enjambre. Qué duda cabe que las líneas de investigación se centran en tecnologías de inteligencia artificial que profundicen en el concepto de cooperación y colaboración entre individuos, con alto carácter de reciprocidad y ayuda mutua, donde el enjambre actúa como un ente colectivo; sin embargo, primeramente resulta esencial solventar la cuestión de las colisiones, antes de integrar estos RPAs autónomos en una estructura organizada, inteligente y sincronizada.

Aunque varios investigadores, partiendo de los postulados de Reynolds, han avanzado en el campo de la elusión de colisiones, se considera crucial el avance experimentado por el KAIST, que efectúo múltiples simulaciones del comportamiento de un enjambre,

llegando a la conclusión de que cada RPA debe guardar una distancia de seguridad con los otros RPAs del enjambre, con una dimensión comprendida entre 5 y 15 veces el tamaño promedio del RPA, lo cual evita en su mayor parte las colisiones, cuestión que ha sido confirmada en este trabajo.

Otro avance realizado por el KAIST ha sido la aplicación de la navegación proporcional (NP) como herramienta para evitar que los RPAs colisionen entre sí en un enjambre. Los investigadores analizaron la ley NP empleada con éxito en el guiado de misiles, para encontrar un algoritmo de colisión evasión aplicable a RPAs. Mediante ecuaciones matemáticas, estos investigadores definieron una ley de control de elusión de colisiones, si bien en el presente trabajo se ha determinado que la ley NP resulta más apropiada para el vuelo en formación de un enjambre de RPAs que para misiones operativas.

Partiendo de los postulados de Reynolds y el KAIST, en este trabajo se ha desarrollado un algoritmo de evitación de colisiones con RPAs del enjambre, definido por tres variables de decisión (factdeca, factdecv y factMmmm). Básicamente, para cada RPA se calculan las distancias al resto de RPAs y se asocia al primer RPA la distancia más corta (dis), de tal modo que cuando dis es inferior a factdecv, el RPA gira de acuerdo a la velocidad angular de viraje proporcionada por la Mecánica del Vuelo. Si aun así, existe riesgo de colisión y dis es inferior a factdeca (factdecv>=factdeca), el RPA se decelera pero sin llegar a entrar en pérdida. No obstante, también se ha visto en este trabajo que es necesario introducir un algoritmo que evite que el enjambre se disgregue y disperse, mediante la adopción de un factor llamado factMmmm, que provoca que al sobrepasar este valor, el RPA vire y se dirija al centroide del enjambre. Obviamente un asunto más peliagudo es el impacto con infraestructuras o elementos ajenos al enjambre, lo cual requiere algo más que el dato de posición de los otros RPAs, como información exterior proporcionada por sensores. A estos efectos, al final del presente trabajo se exploran tecnologías de tratamiento estadístico de imágenes mediante las cuales se podría identificar un obstáculo y evitar su colisión.

En cuanto a las misiones operativas, de entre todas a la que los RPAs podrían aplicarse, para este trabajo se ha seleccionado la de saturación de defensas como ejemplo donde la tecnología aquí desarrollada podría mostrar sus elevadas capacidades potenciales. Un enjambre de RPAs sería un formidable candidato para degradar y/o neutralizar los sistemas de defensa aérea enemigos, lo cual permitiría que

otras operaciones aéreas se desarrollen sin pérdidas innecesarias. Estos enjambres se encargarían de saturar las defensas enemigas generando numerosos objetivos económicos a los que derribar, ya sea con caros misiles o simplemente provocando a los radares, que al conectarse revelarían su posición y serían destruidos. De este modo, este sistema de defensa aérea, diseñado para repeler una agresión de aviones y misiles de mayor tamaño, sería fácilmente colapsado, confundido y abrumado por un grupo de pequeños RPAs, lo cual sería aprovechado por las fuerzas propias. Asimismo, estos RPAs incrementarían artificialmente su firma radar con paneles radar para confundirse con aviones más grandes.

Por otra parte, otra misión operativa aplicable a los enjambres sería la ISR. Estos enjambres son elementos idóneos para desempeñar misiones ISR, dado su pequeño tamaño y elevado número que les permite integrarse colaborativamente para la adquisición, procesamiento y recopilación de información de inteligencia. Indudablemente, este tipo de misiones se desarrollan en zonas de operaciones congestionadas de elevado riesgo, por lo cual estos aparatos deben adoptar una configuración de maniobras evasivas utilizando ardides de desconcierto y decepción contra las defensas aéreas enemigas, todo ello dentro de una estructura cooperativa, inteligente, organizada y sincronizada que incremente la probabilidad de supervivencia del enjambre. De ahí el paralelismo físico matemático entre ambas misiones.

Aunque actualmente los RPAs han alcanzado un grado suficiente de consolidación, el principal problema es dotar a un enjambre de estas aeronaves de un comportamiento colaborativo y de coordinación, donde estos individuos combinen sus habilidades de forma que un comportamiento colectivo, organizado, sinérgico e inteligente surja de la suma del comportamiento de los RPAs individuales; sin embargo, adquirir esta capacidad requiere del desarrollo de algoritmos complejos de inteligencia artificial que permitan construir una red cognitiva entre estas plataformas remotas.

Fundamentalmente, una inteligencia artificial puede considerarse como un conjunto de algoritmos estructurados en programas computacionales, que según el investigador Nils John Nilsson, uno de los fundadores de la inteligencia artificial, debe apoyarse en los siguientes cuatro pilares: búsqueda de estados, métodos de optimización evolutivos, técnicas de aprendizaje automático y razonamiento lógico formal.

En primer lugar, para que el enjambre incremente el efecto saturador, degradativo y neutralizante en el sistema defensivo enemigo y potencie el desconcierto, la confusión,

la decepción y el colapso, cada RPA del enjambre debe evolucionar de forma distinta respecto al resto de RPAs, donde la velocidad de cada vehículo remoto sea diferente. Obviamente, un gran número de objetos volando, cada uno en una dirección distinta, dará “mucho trabajo” al sistema enemigo de defensa aérea. Para lo cual, en este trabajo ha sido necesario definir una función de entropía del enjambre como el número de estados posible, que refleja para el caso de enjambres, el número de RPAs con velocidades diferentes.

Una vez definida la entropía, el mayor efecto neutralizante, de degradación y saturador sobre el sistema de defensa enemigo se conseguirá maximizando la entropía en función de factdeca, factdecv y factMmmm. Como se verá posteriormente, esta función es tremendamente discontinua, ruidosa, variable con el tiempo y con muchos óptimos locales, para lo cual, en este trabajo, se ha desarrollado y programado un algoritmo EPSO (Evolutionary Particle Swarm Optimization) donde las potentes características del algoritmo PSO se ven potenciadas adicionando elementos evolutivos (selección, cruce y mutación). Así, el EPSO teje una estructura que partiendo de un trío factdeca, factdecv y factMmmm, con su correspondiente valor de entropía, es capaz de ir obteniendo sucesivos tríos con entropías mayores, rechazando aquellos con entropías menores, de tal suerte de razonamiento que partiendo de un conjunto de tríos se obtiene un trio distinto de los primeros pero con entropía incrementada.

Aunque el enjambre de RPAs puede ser teóricamente concebido para cualquier tamaño de vehículo remoto, lo habitual es una concepción de enjambre de RPAs como un conjunto de vehículos de tamaño más bien pequeño por diversas razones, ya sea la dificultad de interceptación de RPAs de reducido tamaño, su maniobrabilidad, se camuflan con el entorno,… a lo que se une lo antieconómico que resultan las aeronaves de gran tamaño. Así, ya sea desplegable desde aviones de combate o bien desde lanzadores terrestres, un tamaño pequeño de RPA no permite albergar grandes microprocesadores, lo que a la postre se traduce en una baja capacidad de computación y por consiguiente en una lentitud en los cálculos. Como en las misiones operativas resulta primordial la agilidad en la toma de decisiones, en este trabajo se plantea una metodología basada en efectuar un conjunto de simulaciones, previamente al lanzamiento del enjambre, y aplicando un algoritmo de aprendizaje automático SVM (Support Vector Machine) emplear estas simulaciones como datos de aprendizaje para entrenar al SVM y construir un modelo que prediga el resultado y proporcione una

respuesta para una nueva muestra, de tal modo que el enjambre pueda tomar decisiones por sí mismo. Para ello, se ha hecho uso de la función tune del paquete e1071 de r-project que permite localizar los parámetros óptimos mediantes técnicas de validación cruzada con búsqueda aleatoria.

De esta manera, en este trabajo se ha construido un conjunto de algoritmos que comprenden un EPSO de optimización evolutiva y búsqueda de estados junto a una SVM de aprendizaje automático, donde partiendo de valores de entropía más baja es capaz de encontrar un valor de entropía más alta, lo cual configura un artefacto de inteligencia artificial capaz de decidir entre varias posibilidades que valores proporcionan la entropía más alta en un enjambre de RPAs, lo que finalmente conduce a un algoritmo concebido en su conjunto como un sistema de inteligencia de enjambres.

1.1. Estado del arte

La librería del Congreso de los Estados Unidos, en su documento Mini, Micro, and Swarming Unmanned Aerial Vehicles: a baseline study, hace una revisión exhaustiva de las tecnologías necesarias para impulsar y desarrollar los enjambres de RPAs, así como las líneas de acción a acometer en el futuro. En este documento se recalcan dos desafíos esenciales en este tipo de tecnologías de enjambres: la primera es que los RPAs que componen el enjambre sean capaces de evitar las colisiones entre sí (prioritariamente) y con otros objetos alternativamente, así como la capacidad del enjambre de dirigirse hacia un objetivo y cumplir una determinada misión operativa.

A este respecto, el Congreso destaca los avances alcanzados por el KAIST, ya mencionados previamente, donde partiendo del enjambre como un grupo descentralizado, cada individuo maniobra para evitar colisionar con otros miembros del enjambre, llegando a la conclusión, tras múltiples simulaciones, que una burbuja de seguridad de 5 a 15 veces la longitud media del RPA, permite evitar las colisiones en la mayor parte.

Asimismo, este informe también menciona los estudios realizados por la Universidad de Padua, donde los investigadores montaron cámaras en RPAs al objeto de evitar las colisiones, análogamente a como hacen las aves, encontrando un modelo geométrico de correlación entre el ángulo de visión de la cámara, la distancia de visualización y la

prevención de colisiones. Otro estudio a destacar es el realizado por el KAIST, relativo al empleo de la navegación proporcional de guiado de misiles, para definir una ley de control de elusión de colisiones que consigue evitar que los RPAs colisionen entre sí en un enjambre.

Tras evitar las colisiones, el enjambre debe seleccionar rutas más apropiadas para desplazarse hacia el teatro de operaciones. Un estudio destacable a este respecto es el encabezado por el Shaanxi Engine Design Institute de Xian, el Hebei Electric Power Reconnaissance Design Academy, y el Aircraft Engineering Department of Northwestern Polytechnical University, que aplicaron técnicas heurísticas ACO (Ant Colony Optimization) de coordinación de colonias de hormigas para el desarrollo de algoritmos de caminos óptimos hacia blancos, basándose en la prioridad de búsqueda de alimentos.

Otro enfoque destacable es el de la Universidad de Delft que aplicaron técnicas heurísticas para la simulación de rutas autónomas de supresión de defensas, dependiente de parámetros como la distancia, el tiempo y el esfuerzo, destacando la necesidad de minimizar el tiempo en zona de operaciones. Esta idea de minimizar el tiempo del RPA en territorio enemigo es también objeto de preocupación en el Israel Institute of Technology Technion, que ha desarrollado un algoritmo heurístico capaz de tener en cuenta cambios en el grupo completo, destacando el problema de que incrementar la complejidad computacional proporciona mejores resultados, con el inconveniente de que reacciona con lentitud a los cambios.

El Technion también mostraba un algoritmo de patrones de vuelo que exploraba un área rectangular, de tal forma que diferenciaba entre áreas escaneadas y no escaneadas, donde la conclusión principal es la necesidad de un alto nivel de comunicación entre RPAs. Asimismo, en el Indian Institute of Science en Bangalore, los investigadores analizaron las limitaciones de tiempo en las decisiones óptimas de búsqueda de rutas, proponiendo un algoritmo de ruta óptima basado en la teoría de juegos, comparando tanto estrategias cooperativas como no cooperativas.

Por último, el concepto ultraswarm define un grupo de RPAs que no solo se comporta como un enjambre, sino que también puede combinar su potencia computacional dentro de una red para realizar cálculos complejos, que se prevé podría realizar ciertas funciones de alto nivel, como la de análisis de datos en AWACs, por ejemplo.

Capítulo 2. Concepto de enjambre de RPAs

La era de los RPAs (Remotely Piloted Aircraft System) ha surgido de improviso y ya forma parte de nuestras vidas habiéndose convertido en una tecnología esencial en el mundo actual. Su desarrollo ha sido espectacular para la aviación, su consolidación parece vislumbrarse próxima, el conocimiento y los fundamentos se han apuntalado, y la madurez de estos sistemas se ve cada vez más próxima. Pero aun así, científicos e ingenieros trabajan en nuevos desafíos, incorporando mejoras sucesivas en comunicaciones e inteligencia artificial, y la aplicación de estas tecnologías a estos aviones ha llevado a la aparición de un nuevo elemento estratégico, el enjambre, cuyas capacidades se entrevén impresionantes.

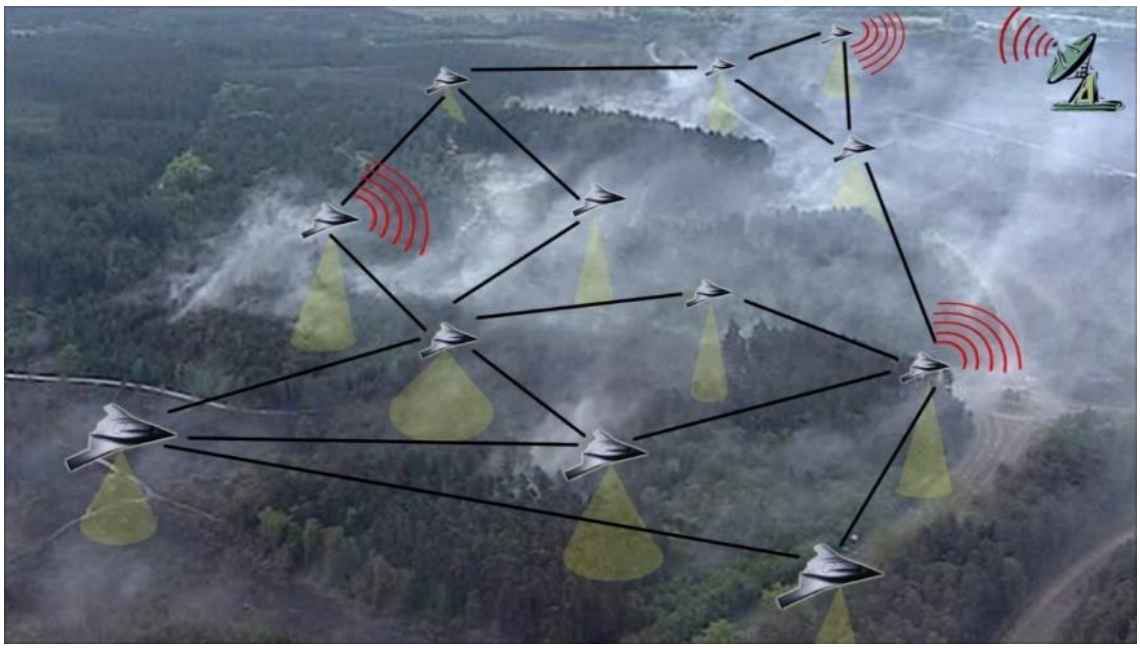

Figura 2.1. Ejemplo de una recreación artística de un enjambre.

Swarm es una palabra proveniente de la naturaleza que describe cómo un grupo de hormigas, abejas u otros insectos combinan sus habilidades de forma cooperativa para lograr un objetivo común. Empero, más que un grupo, se debe hablar de comportamiento cooperativo, donde un conjunto de RPAs actúa de forma coordinada, estableciendo una relación entre los comportamientos individuales simples de cada aparato remoto y un comportamiento inteligente colectivo del enjambre.

Aunque tradicionalmente el enjambre ha sido concebido como aquel formado por RPAs de tipo nano o bien micro (menos de 2 Kg), del tamaño de pequeñas aves o insectos, actualmente este concepto se está expandiendo a otras categorías, planeándose ya proyectos de tamaño mini (2-20 Kg). Este es un reto transcendental porque estos vehículos deben operar sincronizadamente a través de una red de información, tomando decisiones a la vez que analizan el entorno, lo que permite realizar misiones arriesgadas de forma rápida y económica sin poner en peligro la vida de los pilotos.

Descendiendo a un nivel puramente técnico, la definición de un enjambre puede concretarse como un conjunto de RPAs autónomos que generalmente tienen su propia capacidad sensorial y comportamiento reactivo contra los elementos del entorno, y el resto de los RPAs, de modo que un comportamiento colectivo brota de la adición de la conducta de los RPAs individuales.

ACTIVIDADES TIPO “ENJAMBRE”	ANIMALES
Transporte	Hormigas
Búsqueda de alimento	Hormigas, abejas
Construcción de nidos	Hormigas, abejas
División del trabajo	Hormigas, abejas
Generación de rutas	Hormigas, abejas
Construcción de redes	Arañas
Coordinación en el vuelo	Aves
Coordinación natatoria	Peces
Caza cooperativa	Lobos, leones

Tabla 2.1. Actividades tipo “enjambre”.

La idea de emplear un grupo de individuos que puedan operar cooperativamente para lograr un objetivo no es nueva, constituye la esencia del comportamiento de muchos animales en la naturaleza. De esta forma, las abejas y las hormigas dividen el trabajo y se organizan para la búsqueda de comida, los leones y los lobos cazan en manada, los pájaros han aprendido que la coordinación del vuelo disminuye la resistencia al avance y ahorra energía, etc.

Pero este tipo de conducta también es propia de la actividad humana, el hombre prehistórico se vio en la tesitura de aprender a cazar para sobrevivir y adoptó estas técnicas de combate en sus luchas con otras tribus. De este modo, mediante un proceso manifiestamente evolutivo, este hombre aprendió a través del aprendizaje que necesitaba la ayuda de más humanos para sobrevivir aplicando estas estrategias iniciales de sorpresas y emboscadas. Esta sinergia representa la base del nacimiento de los ejércitos, donde la suma de los soldados que pertenecen a una unidad militar aumenta la acción de estos soldados por separado.

Figura 2.2. Acción cooperativa de zapadores paracaidistas en un ejercicio de instrucción.

Aunque actualmente hay RPAs suficientemente consolidados, el problema fundamental es integrar un conjunto de estas aeronaves en una estructura organizada inteligente. Cada uno de estos vehículos remotos debe ser autónomo o semiautónomo al tomar ciertas decisiones, pero también debe proceder como una entidad sincronizada de tal modo que las misiones se

puedan ordenar al sistema completo, como un todo, sin tener que descender a los RPAs individuales.

Los humanos han aprendido comportamientos cooperativos mediante la educación recibida y los militares cumplen su misión en base a la capacitación militar recibida en las Academias Militares, pero alcanzar esta capacidad para todo el enjambre de RPAs requerirá el desarrollo de algoritmos complejos en inteligencia artificial que permitirá construir una red cognoscitiva entre estas aeronaves remotas, de tal modo que se cree un comportamiento colaborativo de apoyo mutuo.

Un enjambre reducirá plausiblemente el tiempo necesario para llevar a cabo una misión ISTAR (Intelligence, Surveillance Target, Adquisition and Reconnaisance) previéndose obtener información de mayor calidad y precisión. Estas plataformas serán extraordinariamente versátiles y adaptables, combinarán sus habilidades de forma inteligente, evitarán colisiones a través de dispositivos complejos, como el “Sense & Avoid” (S&A) por ejemplo, y podrán actuar en caso de pérdida de algunos de ellos en el teatro de operaciones.

2.1. Comunicaciones en enjambres

Una red de comunicaciones de un enjambre consta de un conjunto de características mínimas como implementar tareas críticas, recuperarse de la pérdida de un RPA y tener elevada resistencia a interferencias, fallos o jamming. Actualmente se trabaja con cuatro topologías básicas: enlaces directos, satelitales, celulares y redes de mallas.

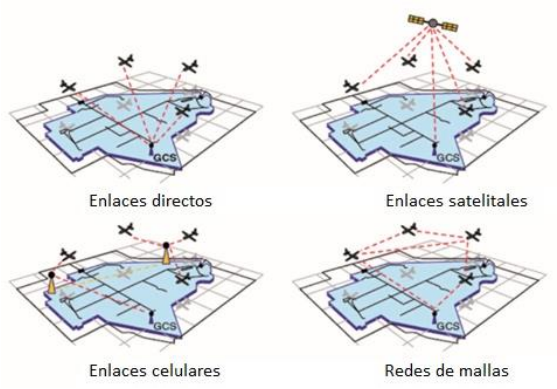

Figura 2.3. Topologías esenciales de comunicación de enjambres.

Los enlaces directos entre la GCS (Ground Control Station) y los RPAs forman una topología centralizada, aunque esta estructura no es apropiada para hacer un amplio uso de las tecnologías de cooperación. Un inconveniente importante es que no puede haber obstáculo alguno entre los enlaces, necesitan transmisores de alta potencia y un gran ancho de banda.

Los enlaces satelitales también forman una topología centralizada, presentan una mayor cobertura que el directo mostrado en el párrafo anterior, no muestran problemas de LOS (Line-Of-Sight) pero requieren medios complejos y caros.

Los enlaces celulares son similares a los de la telefonía móvil con un coste equivalente de radios y repetidores. No obstante, esta topología tiene ventajas destacables como una amplia cobertura, repetidores redundantes, extraordinaria fiabilidad, otros RPAs podrían hacer uso de ella, etc.

Las redes de mallas son topologías en que cada nodo, terrestre o bien instalado en un RPA, pueden funcionar como repetidor. Ventajas destacables son que la comunicación aeronave-aeronave puede ser directa, pueden adoptarse protocolos de enrutamiento, combinarse con topologías previas al ser compatibles aportando redundancia adicional, son robustos y flexibles, y el alcance podría ampliarse utilizando un par de RPAs a modo de repetidores.

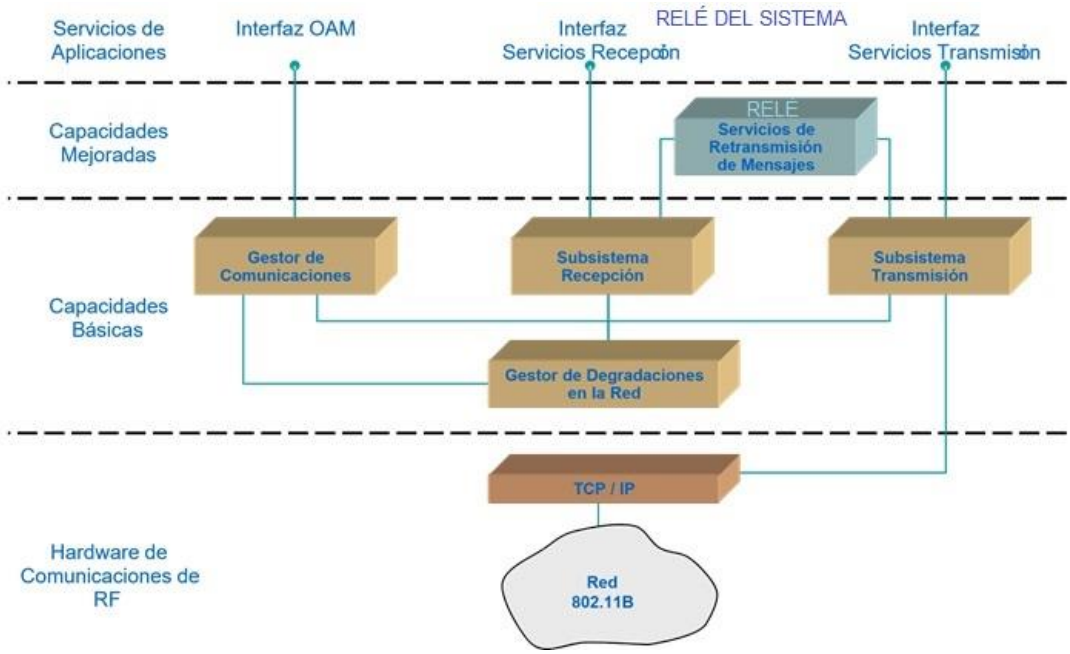

Figura 2.4. Estructura básica de comunicaciones de enjambres.

Redes de malla típicas son las MANET (Mobile and Ad Hoc Wireless Networks), fundamentadas en clusters de nodos, con un nodo central y otros periféricos, nodos pasarelas (gateway) y racimos. Muchas ventajas aporta este sistema como modularidad, resistente a degradaciones, configurable y habitualmente no necesita un administrador de red.

2.2. Gestión y control de enjambres

Disponer de un enjambre en estado operativo requerirá dotar a los RPAs de cierta conducta autónoma, de tal forma que partiendo de acciones individuales simples de aviones remotos individuales, acudiendo a protocolos de apoyo mutuo con un alto carácter de reciprocidad, el enjambre progresa y se adapta en conjunto como si fuera una única entidad.

Para que un enjambre sea autónomo en su vuelo se necesita que cada RPA disponga al menos tres elementos: un piloto automático avanzado, un controlador encargado de evitar colisiones, seguimiento del terreno y mantenimiento de vuelo en formación, y finalmente un elemento de discriminación entre el controlador y el piloto automático.

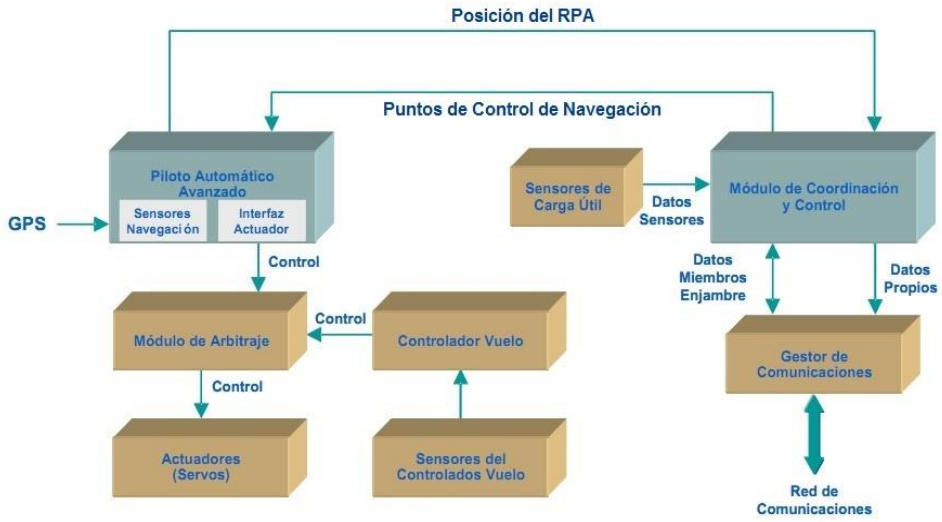

Figura 2.5. Control de una aeronave autónoma.

Actualmente se barajan dos líneas para alcanzar el nivel de cooperación exigido. O bien RPAs, especializados de diferentes tipos que trabajan en equipo para resolver tareas complicadas empleando sensores de altas

características, o bien RPAs iguales y económicos menos versátiles que los primeros.

Los modelos de comportamiento de enjambre se pueden catalogar, de menor a mayor nivel de complejidad, en modelo maestro y subordinados (MS), trabajo en equipo (TE), variables de consenso (VC), agentes inteligentes (AgI) y modelo de campos computacionales (CC) co-fields.

El modelo MS consiste básicamente en un RPA master más grande, controlado desde una GCS, que a la vez está interconectado con un enjambre de RPAs más pequeños dotados de alguna capacidad autónoma, y que mantienen la comunicación con la GCS y el master. La pérdida de algún RPA subordinado no es importante ya que su rol lo asumirá cualquier otro. El vehículo maestro si es significativo por lo que permanecerá a salvo en zona segura libre de conflicto.

TE es un modelo cotidiano que no muestra un comportamiento cooperativo como tal. Estos RPAs podrán ser distintos o iguales, serán configurables mediante un simple cambio de la carga de pago, autónomos, colaborativos, aunque sea de una forma básica donde la función de cada RPA puede ser asumida por otro vehículo remoto.

En el modelo VC, cada RPA posee en su interior la mínima información para coordinarse con el enjambre. Esta información, conocida como variable de consenso, se actualiza aperiódicamente con las cadenas de datos que envían el resto de RPAs, si bien presenta la ventaja de que no es necesario conectarse a tiempos fijos.

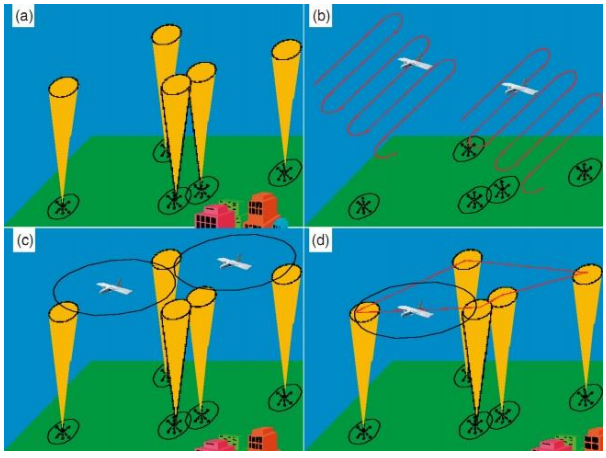

Figura 2.6. Variables de consenso.

En el modelo AgI, cada RPA está dotado de algún tipo de razonamiento y puede tomar ciertas decisiones autónomas, siendo posible incorporar diferentes tipos de RPAs en el grupo. Por otra parte, el modelo considerado por los expertos el más avanzado concebido hasta hoy es el de campos computacionales. Imaginado como una imitación al comportamiento de enjambres de insectos, estos combinan sus destrezas dependiendo del entorno, exponiendo la primera conducta realmente efectiva y colectiva a partir de la adición de las conductas de los simples individuos. Empero, aún no se ha desarrollado un patrón completo aunque se han conjeturado acerca de soluciones parciales, siendo la más notable la CC co-fields, designación debida a que el entorno se sintetiza mediante este tipo de campos.

Figura 2.7. Agentes inteligentes.

Capítulo 3. Saturación de defensas

La supresión de las defensas aéreas enemigas (SEAD, Suppression of Enemy Air Defenses) se define por el Departamento de Defensa (DoD, Department of Defense) como “aquella actividad que neutraliza, destruye o degrada temporalmente las defensas aéreas de superficie del enemigo por medio de rotura y/o disrupción." También son conocidas como operaciones Wild Weasel y Iron Hand en los Estados Unidos con el objetivo de destruir los misiles superficie-aire (SAM, Surface to Air Missile) y la artillería antiaérea (AAA) del enemigo en las primeras horas del ataque como misión principal.

Durante la guerra de Vietnam se organizaron unidades especializadas con aviones diseñados específicamente para ello y se diseñaron las primeras tácticas especiales. Consisten en la destrucción, neutralización o degradación del sistema de defensa superficie-aire del enemigo. Las operaciones SEAD dan la oportunidad a que otras operaciones aéreas se realicen sin pérdidas innecesarias. Aunque no son misiones por sí mismas, han recibido mucha atención debido a que son capaces de allanar el camino a otras operaciones aliadas creando condiciones favorables. Pueden ser destructivas o degradativas.

Aunque las misiones SEAD pudieran parecer fáciles, son de las más complicadas y arriesgadas de cuántas pueda desempeñar una aeronave. Destinadas a destruir o bien neutralizar las defensas aéreas enemigas, las cuales pueden producir un daño importante incrementando el número de bajas, como ya ocurrió en la Guerra de Yom Kippur en 1973 cuando los SAM egipcios diezmaron a los israelíes.

Una de las técnicas que se pueden emplear es el engaño, mediante RPAs, misiles crucero, o una mezcla de ambos, que simulen aviones, de tal forma que saturen las defensas enemigas, con numerosos blancos baratos a los que derribar, gastando caros misiles, o simplemente provocando a los sistemas radar que se conectan y así podrán ser localizados y destruidos. Estos RPAs o bien misiles de crucero, pueden hacerse pasar por aviones mayores, incrementando su firma radar RCS (Radar Cross-Section) mediante paneles que reflejen las ondas del radar de modo que puedan confundirse con algún tipo de avión grande.

Uno de los más avezados innovadores en este campo han sido los israelíes que llevan experimentando desde los años ochenta. Así, la misión SEAD en el valle de la Bekaa fue iniciada por RPAs, que localizaron las instalaciones de las defensas sirias. Un centro de mando y control fue destruido por Unidades de Operaciones Especiales, guillotinando la red. A continuación, los RPAs entraron en acción a modo de señuelos activando la red de radares sirios, que fueron localizados por RPAs de reconocimiento, para a continuación ser atacados por los sistemas de guerra electrónica israelíes. Como colofón final, entraron los misiles antirradiación que terminaron con la red de defensa aérea siria.

Posteriormente, durante la Guerra de Irak de 1991, los Estados Unidos utilizaron RPAs BQM-74C modificados para que tuvieran una firma radar similar a la de un avión. Fueron desplegados en Arabia Saudí y lanzados en oleadas en la primera fase de la campaña aérea, llegaron a la zona de conflicto en Bagdad, donde estuvieron sobrevolando los cielos durante 20 minutos para que las defensas antiaéreas estuvieran ocupadas, y en paralelo fueran destruidos por los misiles AGM-88 HARM.

También se utilizaron los señuelos ADM-141 TALD con la misma misión aunque difería el método de lanzamiento ya que se desplegaban directamente desde aviones. Actualmente existen modelos más avanzados como el ADM-160 capaz de simular la firma radar de aviones concretos como el B-52, con mayores alcances y alturas.

En este contexto han hecho su aparición los enjambres de RPAs con innumerables ventajas, aunque aún se encuentran en fase de desarrollo e

innovación. La principal ventaja de estos enjambres frente a los sistemas descritos anteriormente será la integración de estos vehículos remotos autónomos en una estructura organizada, inteligente y sincronizada, donde estos individuos combinan sus habilidades de forma cooperativa, de manera que un comportamiento colectivo sinérgico surge de la suma del comportamiento de los RPAs individuales.

Y para muestra de un enjambre avanzado, se puede destacar el LOCUST (Low-Cost UAV Swarming Technology), un programa de la ONR/USN (Office of Naval Research/United States Navy), actualmente en fase de desarrollo, considerado uno de los enjambres con la capacidad autónoma más avanzada. Ideado para dispersar un enjambre de RPAs, tipo Coyote, desde un lanzador multitubo; el sistema ha sido proyectado para alcanzar el teatro de operaciones y generar un efecto de decepción y desconcierto en el espacio aéreo enemigo.

Figura 3.1. Programa LOCUST. Lanzador multitubo de Coyotes.

El Coyote es un RPA no reutilizable de bajo coste con un peso de 5.9 Kg, 1.47 m de envergadura, carga de pago de 0.9 Kg y 90 minutos de autonomía. Ideado inicialmente como componente ISR (Intelligence, Surveillance and Reconnaisance) de los aviones/helicópteros antisubmarinos MPA/ASW (Maritime Patrol Aircraft/Anti-Submarine Warfare), cada Coyote se diseñó para ser lanzado desde tubos sonoboyas, frenado con un paracaídas mientras se abren las superficies de vuelo y la hélice, y entra en funcionamiento su motor

eléctrico. Para realizar la función ISR transporta una cámara electróptica digital Sony FCB-IX10A (EO) o una cámara infrarroja (IR) BAE SCC500.

Aunque LOCUST utiliza un RPA probado como el Coyote, la interconexión y sincronización del enjambre requiere una red de inteligencia artificial aún en fase de evaluación y desarrollo, pero cuyas posibilidades se prevén impresionantes y aún por explotar. Cada uno de estos vehículos tiene una firma radar muy pequeña, reduciendo por tanto el tiempo de alerta, lo cual unido a su alto número produciría tal confusión, que podría ser aprovechado por un misil para penetrar un sistema de defensa aérea abrumado y sorprendido. De este modo, un enjambre de este tipo podría colapsar un sistema de defensa aéreo proyectado para repeler una agresión proveniente de aviones y misiles de mayor tamaño.

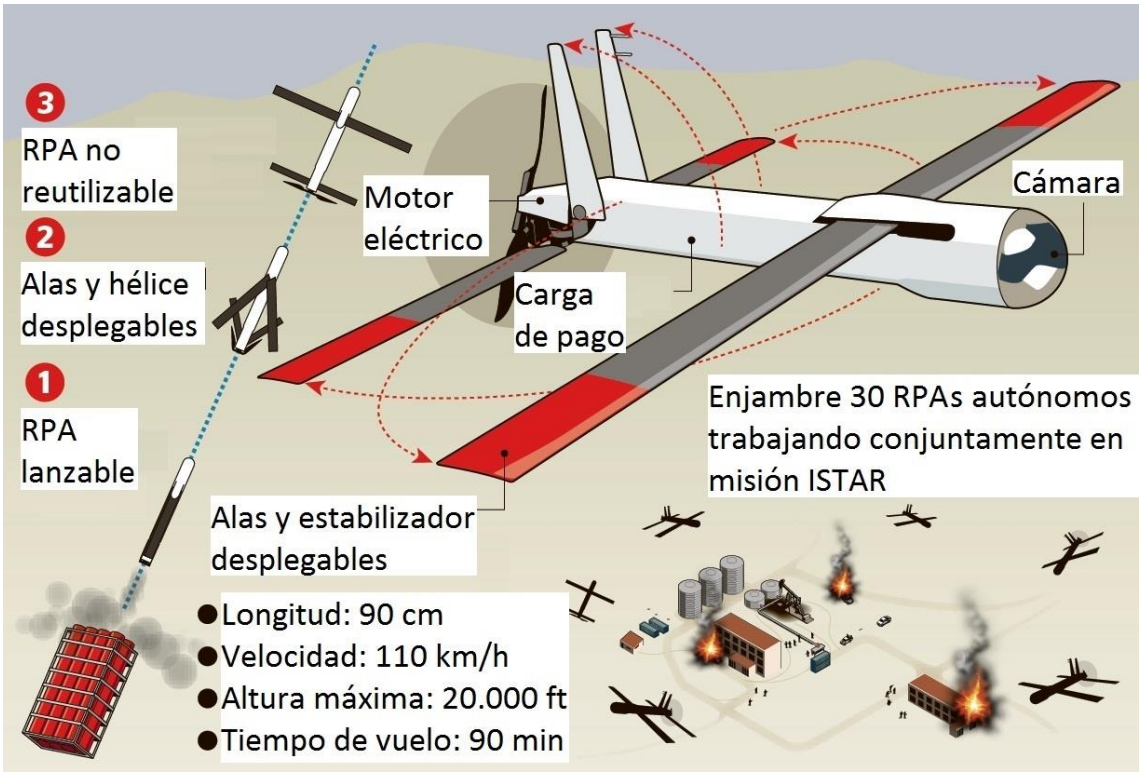

Figura 3.2. RPA Coyote.

La USN se encuentra realizando múltiples simulaciones acerca de cómo afectaría uno de estos enjambres a un Sistema de Defensa de Misiles Aegis, uno de los mejores del mundo, habiendo admitido la propia USN que el Aegis tendría serias dificultades para contrarrestar un número superior a ocho de

estos vehículos. Incluso en el futuro, la USN ya concibe un enjambre defensivo que sería dispersado para contrarrestar una acción armada.

En otro orden de cosas, la Agencia DARPA (Defense Advanced Research Projects Agency) del Departamento de Defensa de los Estados Unidos también se encuentra trabajando en teorías aún más revolucionarias, como es el caso del Gremlin. Con esta designación se designa un enjambre de RPAs reutilizable, barato y lanzable desde varios tipos de aviones, para ser recuperados desde un C-130 Hércules. Esta nueva plataforma no es totalmente reutilizable, pero puede cumplir hasta veinte misiones arriesgadas, con un mantenimiento muy básico, antes de ser dado de baja.

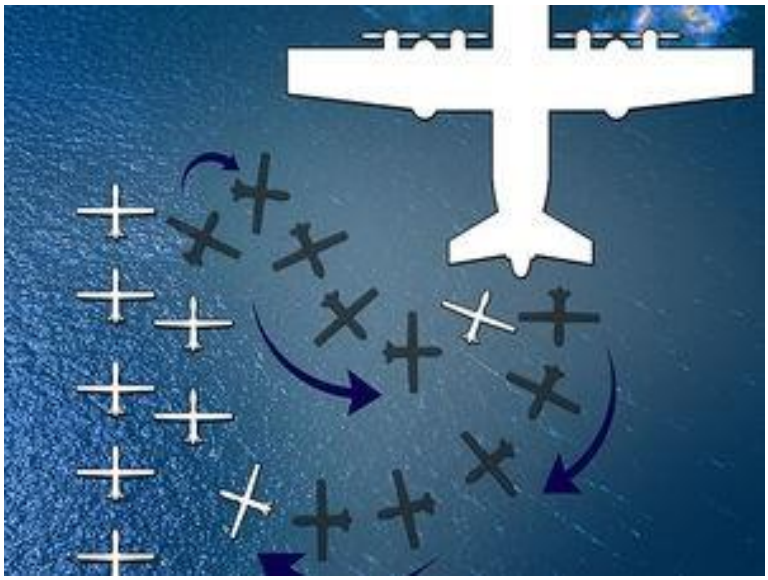

Figura 3.3. Programa Gremlin con un C-130 Hércules recuperando un enjambre.

El Gremlin podrá configurarse para misiones ISR y EW (Electronic Warfare) con el objetivo de saturar las defensas enemigas e inutilizar las redes de comunicaciones. Constituirán una red colaborativa de carácter cognoscitivo donde el éxito de la misión no se verá en absoluto afectado por la pérdida de algún Gremlin.

“System-of-Systems” es otro programa a mencionar de la Agencia DARPA donde misiles, RPAs, y aviones operan coordinadamente. En una simulación efectuada por la DARPA, un caza tripulado no remotamente en misión SEAD analiza el medio y detecta una batería SAM alertando en ese instante a un avión nodriza en espera, a salvo, que lanza un enjambre de RPAs en misión ISR, se acerca a la zona de conflicto recopilando información del enemigo,

estando diseñado para interferir y perturbar a la vez sus radares. Procesada la información y tras confirmar que se trata de un componente hostil, el nodriza despliega un enjambre de misiles LCCM (Low Cost Cruise Missile) en dirección al blanco a batir mientras los RPAs regresan y son recuperados.

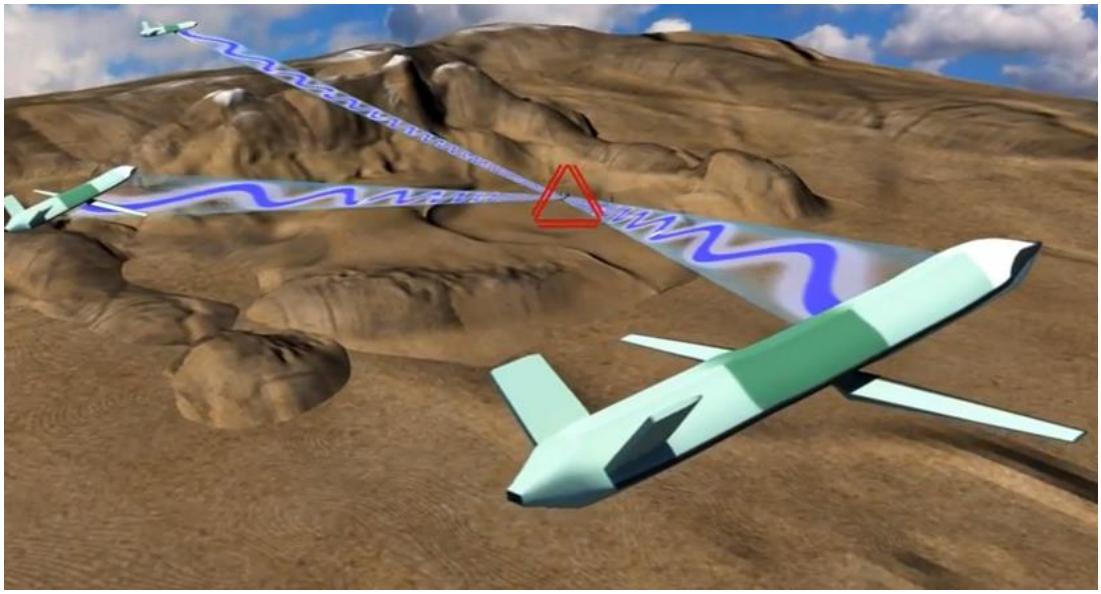

Figura 3.4. "System-of-systems". Enjambre de RPAs recolectando información.

El CODE (Collaborative Operations in Denied Environment), un programa muy reciente de la Agencia DARPA, proyectado para ampliar la autonomía colaborativa de varios RPAs en un área EMOE (Electromagnetic Operational Environment), incluso aunque se pierda la conexión con la GCS. Se prevén capacidades para responder y reconocer obstáculos y situaciones imprevistas, e incluso funcionar como misiles en la fase final.

Figura 3.5. CODE.

Capítulo 4. Misiones ISR

ISR (Intelligence, Surveillance and Reconnaisance) es la adquisición, procesamiento y recopilación de información de inteligencia de manera integrada, coordinada, precisa, coherente y segura, de tal modo que se apoye la conducción de actividades en el teatro de operaciones. ISR abarca múltiples actividades relacionadas con la planificación y operación de sistemas que recopilan, procesan y difunden datos en apoyo de operaciones militares tanto actuales como futuras.

Ejemplos típicos de sistemas ISR abarcan sistemas de vigilancia y reconocimiento que van desde satélites hasta aeronaves tripuladas no remotamente, como el U-2, o RPAs diseñados para ello, como el Global Hawk o el Predator, o incluso equipos marítimos, espaciales, o unidades de inteligencia humana. Los datos de inteligencia suministrados por estos sistemas ISR pueden tomar muchas formas, incluyendo imágenes ópticas, radar, infrarrojas o señales electrónicas. Los datos ISR permiten formar un sistema de alerta temprana para contrarrestar las amenazas del enemigo y permiten que las fuerzas militares aumenten su efectividad, coordinación y letalidad.

Los enjambres de RPAs son elementos idóneos para desempeñar misiones ISR. Estos enjambres se integran de forma cooperativa para la adquisición de información de inteligencia de forma colaborativa, sincronizada, automática e inteligente, donde estos RPAs se combinan y organizan en misiones de vigilancia y reconocimiento. Generalmente, este tipo de misiones ocurren en zonas “calientes” de alto riesgo, donde los RPAs se encuentran en serio peligro de ser derribados por lo que estos aparatos deben adoptar una configuración

de maniobras evasivas durante la misión ISR, generando decepción y desconcierto en las defensas aéreas enemigas, a fin de incrementar su probabilidad de supervivencia. De ahí el paralelismo con las misiones de saturación de defensas y la inclusión de ambas misiones en este trabajo.

Figura 4.1. Despliegue de enjambre de micro-RPAs Perdix desde aviones F/A-18 Super Hornet.

Un nuevo ejemplo de enjambre de RPAs en misión ISR está siendo desarrollado por el DoD. Este organismo anunció el despliegue con éxito de un enjambre de micro-RPAs en misión de vigilancia, desde tres cazas F/A-18 Super Hornet del Naval Air Systems Command (NAVAIR). Estos vehículos remotos son del tipo Perdix, que fueron desarrollados por el Instituto Tecnológico de Massachusetts siendo su principal primicia que cada RPA (agente) es autónomo en su proceder, colaborando y compartiendo información a través de sus data links. Responsables de la SCO (Strategic Capabilities Office) del DoD consideraron el comportamiento de los Perdix como un organismo colectivo que comparte un “cerebro distribuido”, lo cual le permite tomar decisiones y volar en formación. El enjambre adapta su comportamiento de forma automática dependiendo del número de RPAs que constituyan el enjambre en cada momento. En opinión de la SCO, son bastante más que simples individuos sincronizados y preprogramados.

Con una envergadura de 30 centímetros y 290 gramos, el Perdix muestra una disposición clásica de ala fija con cámara incorporada. Ha sido manufacturado

en gran parte con impresoras 3D para reducir costes, se propulsa con una hélice de 66 mm y motor eléctrico con una batería de polímero de litio, lo cual le permite alcanzar 113 km/h durante unos 20 minutos. No es la primera vez que se lanzan Perdix ya que anteriormente habían sido desplegados desde dispensadores de bengalas de aviones F-16 de la USAF (United States Air Force) en Edwards y Alaska. NAVAIR demostró que los Perdix pueden ser desplegados con seguridad a 0.6 Mach y -10°C.

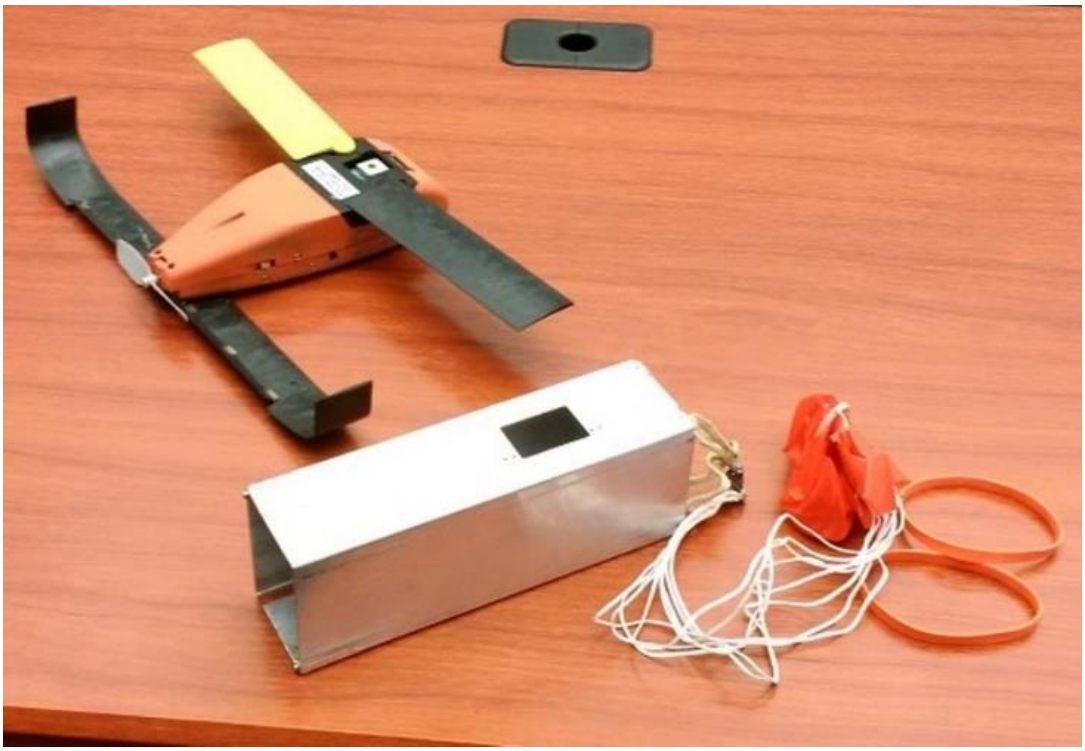

Figura 4.2. Perdix junto a su contenedor de dispensación.

Capítulo 5. El problema de las colisiones

5.1. Principios de Reynolds

En un enjambre no hay posiciones fijas para cada agente sino que las posiciones temporales que ocupan los RPAs del enjambre cambian en cada momento. En el caso particular del vuelo en formación, aunque el aspecto general del enjambre no cambie, las posiciones de cada RPA concreto si lo hace. Aunque no hay razón alguna para ello y de hecho no tiene por qué ocurrir, generalmente el vuelo en formación adopta figuras geométricas.

En lo que respecta a lo que se entiende por vuelo en formación, se deben dar una serie de pautas. Un RPA será el líder de la formación. La formación opera como un único RPA en lo referente a la navegación y la notificación de posición. La separación entre los RPAs que participan en el vuelo será tal que bajo ningún concepto se pueda producir la colisión entre los RPAs que participan en la formación. Tampoco podrá producirse colisión con entes ajenos al propio enjambre.

Se dice que dos o más RPAs vuelan en formación cuando de manera planificada realizan maniobras como si fueran uno solo, siguiendo las directrices del líder de la formación. Es por lo que la labor del líder es esencial al ser el responsable del proceso de toma de decisiones. Por otra parte, la sucesión en el liderazgo es primordial, lo que conlleva que siempre debe existir un líder pero solo uno. La existencia de dos líderes es inadmisible y automáticamente supone la existencia de dos formaciones. Indudablemente deben existir líderes sustitutos que puedan reemplazar al líder en caso de pérdida de este.

Se reconoce a C.W. Reynolds el primero en establecer las reglas básicas a cumplir para simulaciones de enjambres. Reynolds comenzó estudiando las aves aunque él mismo reconoce que estas reglas deben aplicarse también a los peces, las manadas de animales con patas, patrones de tráfico, flujo de automóviles, etc.

En este estudio, Reynolds consideró que los pájaros eran de formas rígidas, ya que el interés primordial de Reynolds era analizar principalmente la naturaleza abstracta subyacente del enjambre como movimiento agregado polarizado, no colisionante, que es connaturalmente independiente del movimiento interno, cambios de forma y articulación de los agentes. Obviamente, el interés de Reynolds es identificar el movimiento de los objetos abstractos del enjambre, sin tener en cuenta cualquier animación interna de los propios agentes.

Así, Reynolds se percata que los enjambres naturales parecen ser el resultado de dos comportamientos opuestos: por un lado, el deseo de permanecer cerca del enjambre en la idea de que la unión hace la fuerza y por otro, el deseo de evitar colisiones con los otros elementos del enjambre. En la naturaleza, la motivación para la primera de estas tendencias podría estar basada en la protección contra los depredadores, obteniendo un cierto beneficio que se materializaría en una mayor cobertura en la búsqueda de alimentos con la consiguiente ventaja para la vida en sociedad y el apareamiento. Reynolds también señala que no hay límite de tamaño para los enjambre, por ejemplo, las migraciones de arenques hacia sus lugares de desove, se extienden hasta en diecisiete millas conteniendo millones de peces. Es evidente que los agentes individuales no prestan atención a todos los otros agentes del enjambre.

Reynolds establece las siguientes reglas, que deben aplicarse en este orden de precedencia, para crear el enjambre:

1. Evitar colisionar con compañeros cercanos del enjambre.
2. Igualar la velocidad con compañeros cercanos del enjambre.
3. Intentar permanecer cerca de compañeros de enjambre cercanos moviéndose hacia su posición promedio.

Adicionalmente, Reynolds hace los siguientes comentarios importantes.

- Los enjambre reales pueden separarse para sortear obstáculos, por lo que los enjambres simulados también deberían disponer de esta capacidad.
- Las fuerzas involucradas en el enjambre deben ser priorizadas en lugar de ser adicionadas. Por ejemplo, volar en un laberinto de calles de una ciudad entre rascacielos, donde en una intersección cualquiera se planteen dos opciones "volar hacia el norte" o "volar hacia el este". Ambas podrían ser aceptables, pero el promedio "volar hacia el noreste" no lo sería. Se puede medir la magnitud de cada fuerza que debería aplicarse, cuando la fuerza máxima aplicable es alcanzada, la última fuerza se recorta.
- Las leyes de control en el modelo de enjambre se deben establecer en términos de cercanía con otros elementos del enjambre. Reynolds considera la vecindad de un agente como una zona esférica centrada en la posición del agente con una cierta sensibilidad que variaría como un exponencial inverso de la distancia. Por tanto, la vecindad se define con dos parámetros: un radio y un exponente. Así, el campo de la sensibilidad debería ser mayor en la dirección de avance y podría depender de la velocidad del RPA.
- Los términos de control de atracción y repulsión usuales que se establecen proporcionales a las distancias, y se corresponden con un modelo de fuerza elástica, con efecto de globo hinchable, no es realista. Un modelo de enjambre más natural y mejor amortiguado se obtuvo cuando se empleó el cuadrado inverso de las distancias, como la fuerza de gravedad o de Coulomb entre cargas eléctricas.
- El camino para el enjambre se especifica en términos de un objetivo global, ya sea como una dirección global, o como un punto de destino hacia el cual todas las aves deberían volar. Esta información podría ser conocida por todos los agentes o únicamente por el líder (o los lideres sustitutos que actuarían únicamente en caso de pérdida del líder principal).
- Reynolds usa dos mecanismos diferentes para evitar colisiones. Uno se basa en el concepto de campo de fuerza y el otro en la de variar el

rumbo para evitar la colisión. De acuerdo con el campo de fuerza, una fuerza de repulsión virtual se genera por los obstáculos que harán desviar a los agentes que se acercan. Este método funciona bien en la mayoría de las situaciones, pero falla en ocasiones. Por ejemplo, si un agente se acerca a un obstáculo en una dirección perpendicular a él, la fuerza reducirá su velocidad a cero y la acelerará hacia atrás generando un movimiento falso dado que el agente debería desplazarse hacia los lados y evitar el obstáculo Una maniobra controlada para evitar el paso con variación de la dirección es un método mucho mejor de evitación de la colisión.

Olfati-Saber amplió muchas de las ideas de Reynolds, destacando también los trabajos de Vicsek, Fax y Murray, Leonard y Fiorelli, Olfati-Saber y Murray con funciones potenciales para el control de la formación, y por otra parte, Khatig, Rimon y Koditschek con potenciales artificiales para evitar obstáculos.

Olfati-Saber consideró que los agentes se comportan como partículas puntuales y describe los enjambres en términos de figuras geométricas con vértices y bordes. Para evitar las colisiones, Olfati-Saber rodea cada RPA con un campo de fuerza que deriva de un potencial.

5.2. Tratamiento de las colisiones en enjambres

En la medida que la capacidad de procesamiento de los computadores aumenta, se incrementa el número de RPAs que componen el enjambre, con el inconveniente de que un número excesivo acentúa el riesgo de colisión. Esto comporta una topología física del enjambre más y más compleja al igual que la red de comunicaciones que coordina y controla el propio enjambre. Sin embargo, las líneas de investigación se concentran más en el concepto de cooperación y colaboración entre individuos que en aumentar el número de agentes en sí, investigando en nuevos protocolos de ayuda mutua y reciprocidad, donde las acciones individuales de cada RPA se optimizan y evolucionan hacia un comportamiento de enjambre colectivo a la par que inteligente. Esto involucra la puesta a punto de técnicas de simulación y algoritmos que establezcan, entre diversas cuestiones, la cantidad mínima de

RPAs que permiten cumplir cada misión concreta, teniendo en cuenta el riesgo de colisiones y la pérdida de agentes del enjambre.

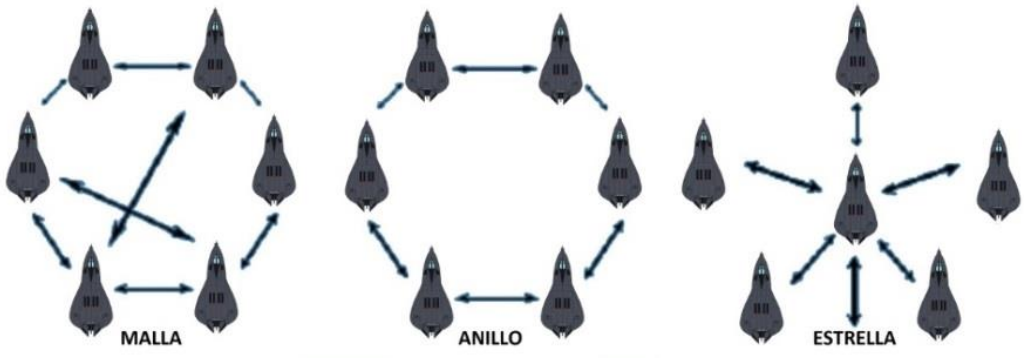

Figura 5.1. Topologías elementales de enjambre (malla, anillo, estrella) de vuelo en formación.

Aunque, el concepto de enjambre técnicamente puede aplicarse a cualquier clase de RPA, comúnmente se ha pensado para categorías nano, micro o mini (peso inferior a 20 kg) por diversas causas. Silenciosos y maniobrables, sus pequeñas dimensiones, una baja firma radar/infrarroja y una reducida velocidad les dan características furtivas, camuflándose con el medio. Baratos y fáciles de fabricar en serie, se disponen en pequeños contenedores y pueden volar no solo en campo abierto sino también dentro de edificios o zonas urbanas, sorteando obstáculos para cumplir su misión.

Para mantener la separación con infraestructuras u objetos es prioritario disponer de alguna tecnología S&A (Sense & Avoid). La función Sense extrae la máxima información del ente entrometido, para posteriormente, la función Avoid analizar y procesar esta información, y decidir si el tráfico localizado constituye algún tipo de riesgo o peligro de choque. Si ese riesgo fuera significativo, el Avoid propondrá al piloto, situado en la GCS, una acción elusiva o bien se pasará a modo autónomo si no hubiera piloto al mando en ese momento, en el caso de que se hubiera producido una caída temporal del enlace por ejemplo. Esto exige disponer de sensores que alerten de la presencia de obstáculos así como recursos de cálculo computacionales, ya sea a bordo de la aeronave, bien a través de los microprocesadores del resto de agentes del enjambre (ultraswarm) que se combinan para realizar cálculos complejos, o bien en un centro de computación en tierra si así fuera posible.

Los S&A cooperativos son aquellos en los que las aeronaves se reconocen mutuamente intercambiando mensajería de forma voluntaria (Sense), incluso mediante una estación en tierra intermediaria (transpondedor). De cualquier forma, cada avión remoto debe componérselas para detectar por sus propios medios edificios, intrusos y aeronaves con las que no estén en comunicación, y averiguar de alguna forma sus parámetros (de vuelo) para que el Avoid facilite una respuesta. Para ello se trabaja en diversas tecnologías aplicables en zona de conflicto, como LIDAR (Laser/Light Imaging Detection And Ranging), radar, sistemas acústicos, IR, EO, DIC (Digital Image Correlation), etc., pudiendo emplearse varias de ellas a la vez.

Estas tecnologías sufren de varias desventajas como gran volumen, peso y consumos excesivos, antenas voluminosas, lo cual solo los hace solo aptos para RPAs grandes. Para los RPAs más pequeños se trabaja en otro tipo de soluciones tendentes a que cada agente conozca la posición del resto de vehículos remotos del enjambre en el espacio y maniobre autónomamente para evitar colisionar con los más cercanos, dado que se asume que el enjambre actúa como un grupo descentralizado. A este respecto, es obligado mencionar la investigación realizada por el KAIST (Korea Advanced Institute of Science and Technology) que efectúo múltiples simulaciones del comportamiento de un enjambre en el cual cada RPA debía guardar una distancia de seguridad con los otros RPAs, llegando a la conclusión, en este estudio, que una zona de seguridad, para cada RPA, con una dimensión comprendida entre 5 y 15 veces el tamaño promedio del RPA en cuestión, es suficiente para impedir las colisiones y por otra parte mantener la cohesión del enjambre (“Multiple Aerial Vehicle Formation Using Swarm Intelligence”). Esta cuestión ha sido corroborada en este trabajo mediante los algoritmos de cálculo desarrollados.

Posteriormente, otro grupo de investigadores del KAIST propuso la navegación proporcional (NP) como herramienta para evitar que los RPA colisionen entre sí en un enjambre. Los investigadores profundizaron en la ley NP, que se ha utilizado con éxito en el guiado de misiles, para encontrar un algoritmo de colisión evasión aplicable a RPAs. Para aplicar la ley PN a la prevención de colisiones, los investigadores definieron una condición de evitación de colisiones y, mediante el uso de ecuaciones matemáticas, especificaron un

vector de colisión-anulación. En este experimento, los investigadores descubrieron que, al encontrar un obstáculo, el RPA usaba una ecuación que definía una maniobra para evitar colisiones, concluyendo que la ley de elusión de colisiones NP utilizada en el guiado de misiles se puede aplicar con éxito a los RPAs. Un análisis más detallado acerca de la ley NP se puede encontrar en los Apéndices al final de este trabajo, si bien dicha tecnología resulta más apropiada para el vuelo en formación que para misiones operativas.

Figura 5.2. Separaciones entre RPAs de 5-15 veces su dimensión media impide las colisiones.

Obviamente un asunto más complicado es el impacto con infraestructuras o elementos ajenos al enjambre, lo cual requiere algo más que el escueto dato de la posición de los otros RPAs, como información exterior proporcionada por sensores. Esta cuestión constituye un problema aparte, como se verá posteriormente, el cual no es trivial y aunque no constituye el objetivo principal de este trabajo, se realizará un análisis posterior más detallado en los Apéndices empleando tecnologías de tratamiento estadístico de imágenes mediante las cuales se podría identificar un obstáculo y evitar la colisión.

Un primer modelo de enjambre MS/VC estaría constituido por un RPA máster, controlado desde la GCS (esto resolvería el problema de impacto con edificaciones o entes que no forman parte del enjambre), liderando al resto de RPAs, de acuerdo a la topología de enjambre escogida y a la separación entre RPAs, de tal modo que no colisionen. Esta topología se podría alterar en vuelo para acometer diferentes misiones. Referente al máster, típicamente se tienen dos corrientes, un máster de mayor tamaño y con más prestaciones que

permanece a salvo en lugar seguro exponiéndose a un riesgo limitado, combinándose con más másteres suplentes que se encuentran en reserva por si el máster principal sufriera algún percance o accidente. Una segunda tendencia consistiría en que el máster es un RPA más del enjambre cuyo rol puede ser asumido por cualquier otro a través de algún protocolo de transferencia de la función máster.

Figura 5.3. RPA Black Hornet.

Volviendo a los RPAs más pequeños, la coordinación y control de estos enjambres demanda que todos los RPAs se comuniquen entre ellos enviando y recibiendo información VC, siendo capital aquella información que evite los choques. Básicamente, cada agente recibirá datos que contendrían la posición espacial del resto de RPAs, con el objetivo de que la distancia de seguridad se conserve dentro de los límites indicados en el párrafo anterior. Esto podría materializarse si cada agente envía una señal discreta de control/coordinación al resto del enjambre; no obstante, el número de señales transmitidas es N*(N-1), lo cual supone un considerable trabajo de procesado/CPU (Central Processing Unit) para cada vehículo remoto cuando el número N de RPAs es muy elevado. Como generalmente estos RPAs tienen un tamaño más bien pequeño, este tamaño no permite albergar grandes microprocesadores lo que

se traduce en una baja capacidad de computación y por ende en una lentitud en los cálculos.

En estadios más avanzados de desarrollo será necesario disponer de una mayor capacidad de computación. De esta forma, en el modelo AgI, cada RPA está dotado de un cierto razonamiento y puede tomar algunas decisiones autónomas al habérsele dotado de una cierta capacidad de inteligencia artificial. El modelo más avanzado en estos momentos es el CC, basado en el comportamiento de un enjambre de insectos, en el que estos insectos suman sus habilidades en función del medio que les rodea, mostrando una conducta ciertamente colectiva a partir de la mezcla de acciones de los elementos individuales del enjambre. Empero, aún no se ha alcanzado un esquema completo, si bien se han perfeccionado soluciones parciales, siendo la más notable la CC.

Estos modelos cobran singular interés en caso de caída de la conexión con la GCS, hackeo de la señal, spoofing del GPS (Global Positioning System), etc. inconvenientes asociados ineludiblemente a estas aeronaves, lo que exige a disponer de capacidad autónoma en estas circunstancias.

Evidentemente las comunicaciones entre RPAs del enjambre son imprescindibles (mucho más que con la GCS por razones obvias), siendo deseable un ancho de banda más bien sobrado y holgados recursos numéricos de computación, para coordinarse sin colisionar, enunciar respuestas contra amenazas en el teatro de operaciones y, en resumen, ser capaz de tomar decisiones en conjunto con el soporte de la inteligencia artificial (IA). En este sentido, es de destacar los avances por parte del gigante chino, ya que esta potencia ha señalado su intención de convertirse en líder mundial de IA para el 2030 incentivando varios proyectos con la designación “Artificial Intelligence 2.0”. Estos proyectos abarcarían sensores inteligentes, big data, cognitive computing, (machine learning) aprendizaje automático e inteligencia de enjambre.

Para el vuelo en formación es necesario especificar una topología de enjambre (anillo, estrella, malla, etc.) y guardar las distancias entre los RPAs dentro de los valores definidos anteriormente para eludir las colisiones. Mantener estas distancias es una de las razones de que los enjambres de RPAs estén

generalmente formados por multirrotores, debido a la mayor maniobrabilidad y control de estas aeronaves frente a la estabilidad del ala fija.

Para que una plataforma aérea sea autónoma en su vuelo se requieren al menos tres componentes principales: un piloto automático, un controlador encargado de evitar colisiones, capacidad de seguimiento del terreno y vuelo en formación, y finalmente un dispositivo de discriminación controlador/piloto automático. Si el espacio a bordo de los RPAs fuera muy limitado, el sistema de seguimiento del terreno y el piloto automático se integrarían en la GCS que actuaría sobre la aeronave máster.

Técnicas de estimación de distancias	ToA (Time of Arrival)	TDoA (Time Difference of Arrival)	AoA (Angle of Arrival)	RSSI (Received Signal Strenght Indicator)				
Técnicas de estimación de posición	Trilateración	Triangulación	Multilateración	Proximidad				
Técnicas de localización basadas en distancias	GNSS (Global Navigation Satellite System)	Cricket	AHLoS (Ad-Hoc Localization System)	RADAR (IEEE 802.11 WaveLAN)	APS (Ad-hoc Positioning System)	N-Hop Multila-teration	MDS-MAP (MultiDi-menSional Scaling)	
Técnicas de localización libres de distancias	Intersección rectangular	Localización amorfa	Fingerprinting (Huella Digital)	APIT (Approximate Point In Triangulation)	Intersección hexagonal			

Tabla 5.1. Técnicas de localización.

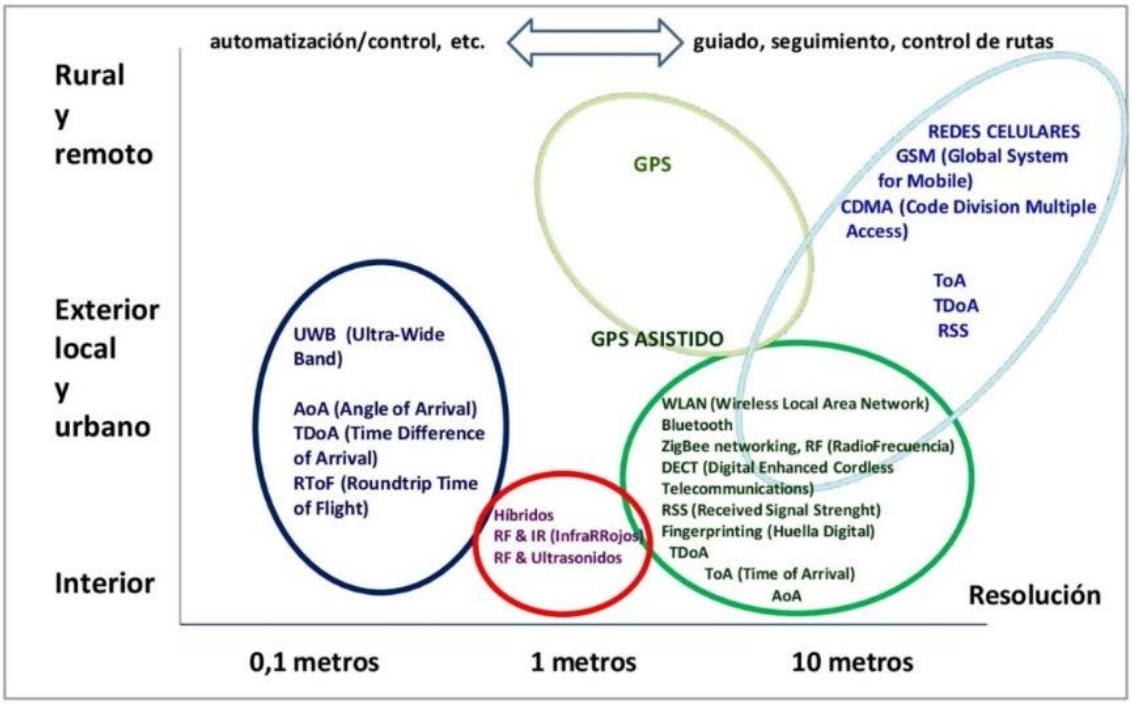

Figura 5.4. Precisión de las técnicas de localización.

Así, cada agente debe conocer la posición del resto de RPAs para no chocar con ellos, lo que se traduce en que cada RPA tiene que conocer su posición exacta en el espacio en todo momento (técnicas de localización) y esta información se transmita al resto del enjambre. Indubitablemente, los dispositivos satelitales (GNSS) son los más utilizados en los sistemas de localización actuales pero necesitan módulos SAASM (Selective Availability Anti-Spoofing Module) que impidan el spoofing, no siendo muy eficaces en espacios cerrados debido a la atenuación de la señal en el interior de edificios. Diversas tecnologías intentan mitigar este contratiempo, como el GSM (Global System for Mobile), WiFi, Bluetooth, etc. u otras menos populares tipo redes de sensores inalámbricos. Tampoco debe perderse de vista la precisión que estos sistemas alcanzan lo cual es un factor a tener en cuenta en cualquier desarrollo.

Capítulo 6. Mecánica de vuelo de RPAs en enjambres

En cuanto a la modelización de cada RPA de forma individual, en este trabajo, se va a considerar el movimiento del centro de masas de cada RPA, como un cuerpo puntual de masa cuasi-constante con tres grados de libertad bajo la acción de diversas fuerzas, aunque una de ellas está atenuada como se verá a continuación. Este movimiento se encuentra definido en cada instante por la posición, la velocidad y la masa del vehículo remoto (considerado como una masa puntual). En cada instante el vehículo está sujeto a una fuerza total compuesta por la fuerza gravitatoria $\mathrm{m}\vec{\mathrm{g}}$, la fuerza aerodinámica $\vec{\mathrm{F}}_{\mathrm{A}}$ y la fuerza propulsiva $\vec{\mathrm{F}}_{\mathrm{T}}$.

Las ecuaciones del movimiento respecto de un sistema inercial de referencia son:

$$\frac{\mathrm{d}\vec{\mathrm{r}}}{\mathrm{dt}} = \vec{\mathrm{V}}_{\mathrm{RPA}} \tag{6.1}$$

$$\mathrm{m}\frac{\mathrm{d}\vec{\mathrm{V}}_{\mathrm{RPA}}}{\mathrm{dt}} = \vec{\mathrm{F}}_{\mathrm{A}} + \vec{\mathrm{F}}_{\mathrm{T}} + \mathrm{m}\vec{\mathrm{g}} \tag{6.2}$$

Donde $\vec{\mathrm{r}}$ es el vector de posición, t el tiempo, $\vec{\mathrm{V}}_{\mathrm{RPA}}$ la velocidad absoluta del vehículo (velocidad respecto del sistema inercial) y m la masa del vehículo. Las derivadas de los vectores $\vec{\mathrm{r}}$ y $\vec{\mathrm{V}}_{\mathrm{RPA}}$ se efectúan respecto al sistema inercial de referencia. Así, se tiene un sistema de seis ecuaciones diferenciales ordinarias para las tres componentes del vector de posición y las tres del vector velocidad.

Un análisis aerodinámico y propulsivo del vehículo proporcionara las fuerzas $\vec{F}_A$ y $\vec{F}_T$.

Si se considera un sistema de ejes fijos respecto a Tierra, se emplearía la velocidad relativa respecto a la superficie terrestre en lugar de la velocidad absoluta. En este caso, las ecuaciones del movimiento en estos ejes (ejes móviles referentes al sistema inercial con velocidad angular constante) quedarían, aplicando las transformaciones correspondientes:

$$\frac{d\vec{r}}{dt} = \vec{V}_{RPA} \tag{6.1}$$

$$m\frac{d\vec{V}_{RPA}}{dt} = \vec{F}_A + \vec{F}_T + m\vec{g} - 2m\vec{\omega} \times \vec{V}_{RPA} - m\vec{\omega} \times (\vec{\omega} \times \vec{r}) \tag{6.3}$$

Los términos $2m\vec{\omega} \times \vec{V}_{RPA}$ y $m\vec{\omega} \times (\vec{\omega} \times \vec{r})$ constituyen las aceleraciones de Coriolis y centrífuga, respectivamente, correspondientes a la rotación de la Tierra. Con carácter general, estas fuerzas de inercia son despreciables comparadas con la fuerza gravitatoria en los casos de baja velocidad y vuelo a baja altitud (que es el caso de los RPAs analizados en este trabajo) siendo el orden de magnitud de estos términos, comparados con los otros, de aproximadamente 10^{-3}, por lo que se desprecian como hipótesis de trabajo. Análogamente, las variaciones de gravedad para el vuelo de RPAs a baja altitud son despreciables también, por lo que puede suponerse la gravedad constante e igual a su valor al nivel del mar.

Por otra parte, en vuelos de corto recorrido, la hipótesis de Tierra plana está plenamente justificada con lo que la superficie terrestre se puede suponer plana y el sistema de referencia ligado a esta superficie se puede asumir como inercial con el plano xy (plano horizontal) y el eje z formando un triedro a derechas. La atmósfera se considera en calma (sin viento) lo que elimina un factor distorsionador que no haría otra cosa que enmascarar los resultados y conclusiones importantes sin aportar valor añadido alguno.

El movimiento de primera magnitud del enjambre se va a desarrollar en planos horizontales paralelos al suelo, siendo el movimiento en el eje z un orden de magnitud inferior comparado con los ejes x e y. Esta hipótesis es plausible

debido a que las misiones de saturación de defensas e ISR se suelen desarrollar a una cierta altura en la zona de conflicto con variaciones suaves en altura. Así $V_{RPAz} \ll V_{RPAx}, V_{RPAy}$, con lo que $V_{RPAz} \cong \epsilon$, muy pequeño, y $V_{RPAx} \sim V_{RPAy}$. Desarrollando las ecuaciones dinámicas

$$\begin{aligned} \frac{dz}{dt} &= V_{RPAz} \\ \frac{dx}{dt} &= V_{RPAx} \\ \frac{dy}{dt} &= V_{RPAy} \end{aligned} \tag{6.1}$$

$$\begin{aligned} m\frac{d\vec{V}_{RPAz}}{dt} &= \vec{F}_{Az} + \vec{F}_{Tz} + m\vec{g}_z \\ m\frac{d\vec{V}_{RPAx}}{dt} &= \vec{F}_{Ax} + \vec{F}_{Tx} + m\vec{g}_x \\ m\frac{d\vec{V}_{RPAy}}{dt} &= \vec{F}_{Ay} + \vec{F}_{Ty} + m\vec{g}_y \end{aligned} \tag{6.4}$$

Con lo que

$$\begin{aligned} 0 \approx m\epsilon &= \vec{F}_{Az} + \vec{F}_{Tz} + m\vec{g}_z \\ m\frac{d\vec{V}_{RPAx}}{dt} &= \vec{F}_{Ax} + \vec{F}_{Tx} + m\vec{g}_x \\ m\frac{d\vec{V}_{RPAy}}{dt} &= \vec{F}_{Ay} + \vec{F}_{Ty} + m\vec{g}_y \end{aligned} \tag{6.5}$$

Generalmente, para el caso de RPAs de ala fija $0 \approx \vec{F}_{Tz} \ll \vec{F}_{Az} + m\vec{g}_z$ aunque en el caso de helicópteros multirrotores, la fuerza de propulsión puede ser comparable al resto de términos de la ecuación. Para las otras dos ecuaciones dinámicas se asume que la propulsión se consigue fundamentalmente con el motor, es decir $\vec{F}_{Ax} + \vec{F}_{Tx} \gg m\vec{g}_x$ y $\vec{F}_{Ay} + \vec{F}_{Ty} \gg m\vec{g}_y$. La fuerza $\vec{F}_{Az}$ suele asociarse a la sustentación y las fuerzas $\vec{F}_{Ax}$ y $\vec{F}_{Ay}$ a la resistencia aerodinámica. En el caso de que no haya propulsión, el RPA se decelera debido a la resistencia aerodinámica que tiende a frenar el vehículo. Los

helicópteros multirrotores necesitan de la propulsión para mantenerse en el aire debido a que la sustentación es muy pequeña, motivada principalmente porque la velocidad de desplazamiento es muy baja e insuficiente para producir sustentación. Así,

$$\frac{dz}{dt} \approx 0$$

$$\frac{dx}{dt} = V_{RPAx} \tag{6.7}$$

$$\frac{dy}{dt} = V_{RPAy}$$

$$m\frac{d\vec{V}_{RPAx}}{dt} = \vec{F}_{Ax} + \vec{F}_{Tx}$$

$$m\frac{d\vec{V}_{RPAy}}{dt} = \vec{F}_{Ay} + \vec{F}_{Ty} \tag{6.8}$$

Como hipótesis de cálculo adicional se va a considerar que el sistema de control de cada RPA tiende a mantener una velocidad de crucero (velocidad estable, constante y uniforme que puede llevar una aeronave en condiciones normales sin sufrir perturbación o variación), de tal modo que en el caso en el que el RPA se viera obligado a decelerarse (como se verá posteriormente a lo largo de este trabajo) por alguna circunstancia sobrevenida, como evitar colisiones por ejemplo, esta deceleración es constante, la cual se infiere de la Mecánica del Vuelo del RPA. En el caso de que el RPA tenga una velocidad inferior a la de crucero, y ninguna circunstancia lo impida, se acelerará con una aceleración constante obtenida de la Mecánica del Vuelo del RPA hasta alcanzar su velocidad de crucero. De cualquier modo, el RPA no podrá descender de la velocidad de entrada en pérdida o velocidad mínima a la que el RPA es capaz de mantenerse en vuelo en el aire con sustentación capaz de igualar al peso y no perder así altura (ala fija). Para helicópteros multirrotores, la velocidad de entrada en pérdida es cero. Estas hipótesis permitirán desarrollar un modelo para cada RPA, como elemento individual, conforme con la realidad, sin comprometer excesivos recursos computacionales, lo cual

permitirá emplear mayores recursos de cálculo en el desarrollo de un algoritmo tendente a alcanzar una cierta inteligencia de enjambre.

Para analizar el efecto del peso en los RPAs, se debe estudiar en primer lugar el grupo motopropulsor. Comúnmente y debido al factor coste, las iniciativas para la innovación de motores específicos para RPAs han quedado en la mesa de dibujo, acudiendo a la aviación convencional para la obtención de los propulsores o por otra parte también de aeromodelos. De esta forma, dependiendo de la categoría de los RPAs, se disponen de varios tipos de motores como son los de combustión interna (gasolina, diésel,…), turbinas de gas y eléctricos (paneles solares, baterías, celdas de combustible,…). Para los de clase I (menores de 20 kg), la tendencia actual es la propulsión eléctrica debido a ventajas evidentes como baja detectabilidad, alta eficiencia, bajo nivel de mantenimiento, contaminación mínima, etc. Aunque, técnicamente, el concepto enjambre se podría aplicar a cualquier clase de RPA, típicamente esta tecnología se ha pensado para categorías nano, micro o mini (peso inferior a 20 kg) por múltiples razones. Maniobrables, silenciosos, de pequeñas dimensiones, baja señal de radar/infrarrojo unido a su baja velocidad les dan una cierta capacidad furtiva.

Entre las baterías recargables más comunes utilizadas en los motores eléctricos se pueden citar las de NiMH (Hidruro metálico de níquel), que presentan una gran capacidad de carga por unidad de volumen. Más avanzadas son las de ion litio pero son mucho más caras, no tienen efecto memoria y son de gran capacidad. El litio polímero (LIPO) son las más evolucionadas aunque más frágiles. Por último, las de aire Zinc brindan indiscutibles ventajas con respecto a las anteriores pero aún deben de ser mejoradas.

Así, un RPA con grupo motopropulsor eléctrico y baterías, pierde masa, en condiciones normales de funcionamiento de la batería, debido a la conversión de la energía química almacenada en la propia batería en corriente eléctrica. Esta pérdida de masa de la batería se puede cifrar, dependiendo del peso y tipo de batería, en un valor inferior a un 0.5% en el proceso completo de descarga, lo cual a la postre se traduce en $\mathrm{dm}/\mathrm{dt} \approx 0$ confirmando de esta forma la hipótesis indicada anteriormente de masa constante.

Capítulo 7. Algoritmo de elusión de colisiones

Continuando con lo indicado en el capítulo anterior, se asume la hipótesis de Tierra plana, atmósfera en calma y movimiento del enjambre de RPAs en planos horizontales paralelos al suelo, con el movimiento en el eje z un orden de magnitud inferior comparado con los ejes x e y. Esta hipótesis es perfectamente plausible debido a que las misiones de saturación de defensas e ISR tienen a ocurrir a una cierta altura sobre el teatro de operaciones con variaciones suaves en altura, dado que los RPAs al acercarse a saturar los radares enemigos no suelen realizar cambios bruscos de altura por la proximidad del suelo. La masa de cada RPA se considera constante como se ha justificado anteriormente.

Para mantener la separación con construcciones u objetos se necesita disponer de algún tipo de tecnología. La tecnología Sense & Avoid podría ser una posible solución aunque necesita hacer uso de sensores de alto nivel del tipo LIDAR, radar, acústica, IR, EO, DIC,… y amplios recursos de cálculo computacionales. En definitiva, que necesita albergar equipos de gran volumen, peso y consumos excesivos, antenas voluminosas, con lo que esta tecnología solo es apta para RPAs grandes.

En el caso de RPAs más pequeños (hipótesis de cálculo en este trabajo), se trabaja en otro tipo de soluciones tendentes a que cada agente conozca la posición del resto de vehículos remotos del enjambre en el espacio y maniobre autónomamente para evitar colisionar con los más cercanos, asumiendo que el

enjambre actúa como un grupo descentralizado. Obviamente, esto requiere que todos los RPAs del enjambre se comuniquen entre ellos enviando y recibiendo información VC, siendo vital aquella información que evite los choques. Esencialmente, cada agente recibirá datos diversos, siendo los más importantes a efectos de evitación de colisiones aquellos que contienen la posición espacial del resto de RPAs, con el objetivo de que la distancia de seguridad se conserve dentro de los límites indicados anteriormente. Esto podría realizarse si cada agente envía una señal discreta de control/coordinación al resto del enjambre.

Retrotrayendo a la investigación del KAIST mencionada previamente, cada RPA del enjambre debe guardar una distancia de seguridad con los otros RPAs, con una dimensión comprendida entre 5 y 15 veces el tamaño promedio del RPA, lo cual es suficiente para impedir las colisiones y por otra parte mantener la cohesión del enjambre.

Así, se asume que de alguna forma y tal como se ha explicado anteriormente, todo RPA está dotado de algún dispositivo que le permite conocer la posición del resto de RPAs del enjambre y maniobrar autónomamente para evitar colisionar con los más cercanos. En cuanto a los choques de RPAs del enjambre con construcciones u otros elementos ajenos al propio enjambre, esta cuestión es algo más peliaguda y obviamente requiere algo más que el dato de la posición de los otros RPAs, como información exterior proporcionada por sensores. Este problema en cuestión no resulta trivial y aunque no constituye el objetivo principal de este trabajo, en los Apéndices se ha realizado un interesante análisis de investigación en detalle empleando tecnologías de tratamiento estadístico de imágenes mediante las cuales se podría identificar un obstáculo y evitar la colisión.

En definitiva, se establece como hipótesis de cálculo adicional en este trabajo que los RPAs del enjambre, objeto de este estudio, solo pueden colisionar con otros RPAs del propio enjambre, no existiendo ningún otro objeto con el que puedan chocar.

Una vez que cada RPA conoce la posición del resto de RPAs del enjambre, hay que diseñar un algoritmo que permita al RPA maniobrar autónomamente para evitar colisionar con los más cercanos. Para ello, se monta un bucle para todos los RPAs y se calculan las distancias al resto de RPAs del enjambre, asociando a cada RPA la distancia mínima al RPA más cercano así como el identificativo de dicho RPA. Cuando esta distancia disminuye de un valor, que se ha denominado factdecv*tam (factor de decisión por tamaño promedio del RPA), el RPA gira de acuerdo a la velocidad angular de viraje ($\dot{\xi}$), extraída de la Mecánica del Vuelo del RPA, para no colisionar. Así, para cada paso de tiempo (Δt), el RPA gira una velocidad $\xi_{max} = \dot{\xi} \cdot \Delta t$.

Se toma una referencia angular y se calculan dos ángulos para cada RPA. El que forma el vector velocidad (V) y el que forma el vector (A) con respecto a la referencia angular. El vector A es aquel que parte del RPA en cuestión (origen) y apunta al RPA más cercano. Para que el RPA en cuestión se aleje del RPA más cercano, el ángulo entre el vector V y el vector A tiene que aumentar sin sobrepasar los 180º (ángulos sexagesimales). De una forma simbólica:

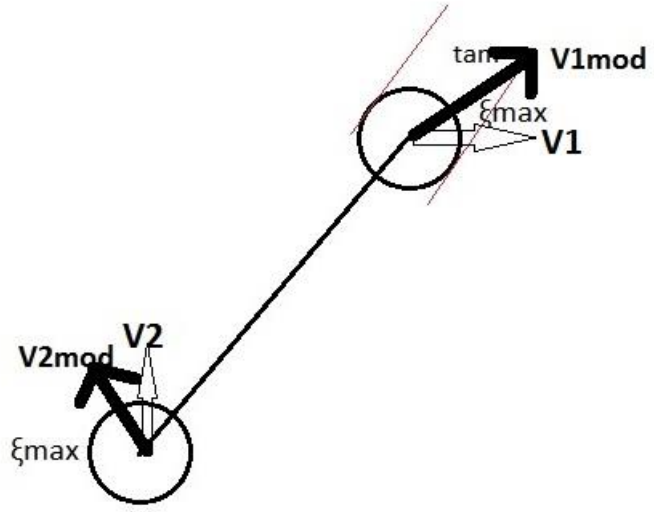

Figura 7.1. Esquema de ángulos para dos RPAs que se acercan.

if abs(angulo VECTOR V–angulo VECTOR A)<180º

{

nuevo ángulo VECTOR V=ángulo VECTOR V+signo(VECTOR V–VECTOR A)*ξ_{max}

}

else

{ ## abs(ángulo VECTOR V–ángulo VECTOR A)≥180º

nuevo ángulo VECTOR V=ángulo VECTOR V-signo(VECTOR V–VECTOR A)*ξ_{max}

}

Por otra parte, independientemente de lo anterior, si aun girando con la velocidad angular de viraje, ambos RPAs siguen acercándose con riesgo de colisión, actúa el siguiente algoritmo que decelera el RPA con una deceleración constante extraída de la Mecánica del Vuelo del RPA. Esto se implementa cuando la distancia disminuye de factdeca*tam.

Con los dos algoritmos expuestos anteriormente, se han efectuado múltiples simulaciones con éxito evitando las colisiones. Pero ahora aparece otro problema para evitar que el enjambre se disperse y disgregue por el espacio aéreo. Esto se consigue introduciendo otro factor que se ha venido en denominar factMmmm*tam que lo que hace es que el RPA en cuestión, al alejarse por encima de este factor gire de acuerdo a la velocidad angular de viraje ($\dot{\xi}$) regresando hacia el enjambre, hacia un punto que se ha llamado centroide del enjambre.

Para ello, se monta un bucle para todos los RPAs y se calculan las distancias de todos los RPAs al centroide del enjambre. Cuando esta distancia aumenta por encima de factMmmm*tam, el RPA gira de acuerdo a la velocidad angular de viraje ($\dot{\xi}$), extraída de la Mecánica del Vuelo del RPA, dirigiéndose al centroide del enjambre. A este respecto, el centroide del enjambre es un punto representativo del enjambre alrededor del cual se desplazan todos los RPAs. Aunque podría considerarse un punto análogo al centro de masas de un sistema, el centroide no es tan restrictivo sino que más bien es un punto que identifica el enjambre y que conocido este permite conocer la posición de todos

los RPAs del enjambre. Cuando el piloto tome el control del enjambre, el elevado número de RPAs hace imposible que el piloto pueda pilotar cada una de los vehículos remotos de forma individual siendo especialmente útil este centroide ya que el piloto si podrá pilotar este centroide del enjambre, especialmente en misiones ISR. Evidentemente se tiene que cumplir por la propia definición que factMmmm≥factdecv≥factdeca.

Se toma una referencia angular y se calculan dos ángulos para cada RPA. El que forma el vector velocidad (V) y el que forma el vector (B) con respecto a la referencia angular. El vector B es aquel que parte del RPA en cuestión (origen) y apunta al centroide. Para que el RPA en cuestión se acerque al centroide, el ángulo entre el vector V y el vector B tiene que disminuir. De una forma simbólica:

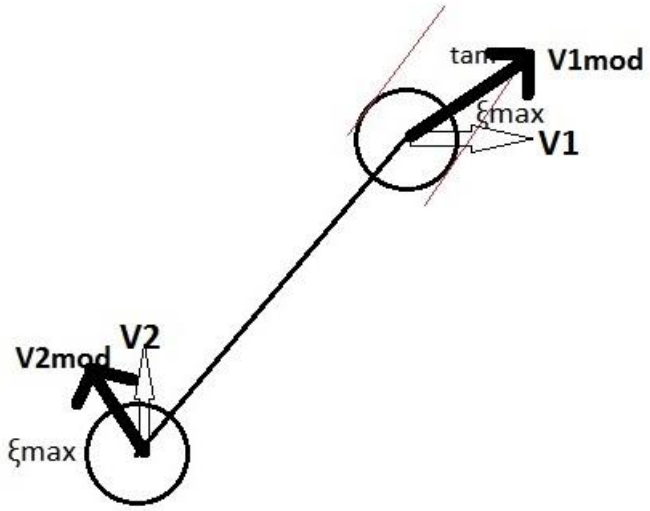

Figura 7.2. Esquema de ángulos para un RPA que se aleja del centroide.

if abs(ángulo VECTOR V–ángulo VECTOR B)<180º

{

nuevo ángulo VECTOR V=ángulo VECTOR V-signo(VECTOR V–VECTOR B)*ξ_{max}

}

else

{ ## abs(ángulo VECTOR V–ángulo VECTOR B)≥180º

nuevo ángulo VECTOR V=ángulo VECTOR V+signo(VECTOR V–VECTOR B)*ξ_{max}

}

Además, según se ha comentado anteriormente, como hipótesis de cálculo extra se va a considerar que el sistema de control de cada RPA tiende a mantener una velocidad de crucero constante (velocidad estable, constante y uniforme que puede llevar una aeronave en condiciones normales sin sufrir perturbación o variación). Así, en el caso de que no haya riesgo de colisión, el algoritmo evalúa la velocidad en cada paso de tiempo y si es inferior a su velocidad de crucero, se acelera con una aceleración constante extraída de la Mecánica del Vuelo del RPA, pero sin sobrepasar la velocidad de crucero. De cualquier forma, el RPA no podrá descender de la velocidad de entrada en pérdida.

Este conjunto de hipótesis va a permitir desarrollar un modelo, acorde con la realidad, sin comprometer gran cantidad de recursos computacionales, con el objetivo de dedicar mayores recursos de cálculo al desarrollo de un algoritmo tendente a conseguir una cierta inteligencia de enjambre.

7.1. r-project

Para resolver el problema que se aborda en este trabajo se han implementando los códigos necesarios en lenguaje de programación r-project.

r-project es un entorno y lenguaje de programación enfocado principalmente al análisis estadístico. Está basado en el software libre del lenguaje S pero con soporte de alcance estático. Así, se trata de uno de los lenguajes de programación más utilizados en investigación estadística, siendo además muy popular en el campo de la minería de datos, la bioinformática, la investigación biomédica y las matemáticas financieras. También contribuye la posibilidad de

cargar una gran cantidad de bibliotecas o paquetes con funcionalidades de cálculo o graficación.

r-project es parte del sistema GNU y se distribuye bajo licencia GNU GPL. Está disponible para los sistemas operativos Windows, Macintosh, Unix y GNU/Linux proporcionando un extenso abanico de herramientas estadísticas (modelos lineales y no lineales, algoritmos de clasificación, test estadísticos, series temporales, algoritmos de clasificación y agrupamiento, etc.) y gráficas.

Al igual que S, se trata de un lenguaje de programación, lo que permite al usuario definir sus propias funciones. De hecho, una gran parte de las funciones de R están escritas en el mismo R, aunque para algoritmos más complejos es posible desarrollar bibliotecas en C, C++ o Fortran que se cargarían dinámicamente. Los usuarios más avezados pueden también manipular los objetos de R directamente desde código desarrollado en C. R también puede extenderse a través de paquetes desarrollados por su propia comunidad de usuarios.

R hereda de S su orientación a objetos. Además, extender R es más fácil por su permisiva política de lexical scoping. R también puede integrarse con distintas bases de datos y existen bibliotecas que facilitan su utilización desde lenguajes de programación interpretados como Perl y Python. Otra de las características esenciales de R es su capacidad gráfica, que permite generar gráficos con alta calidad. Asimismo, R posee su propio formato para la documentación basado en LaTeX.

R también puede usarse como herramienta de cálculo numérico, campo en el que puede ser tan eficaz como otras herramientas especializadas como GNU Octave y su equivalente comercial MATLAB. Se ha desarrollado una interfaz, RWeka7 para interactuar con Weka que permite leer y escribir ficheros en el formato arff y enriquecer R con los algoritmos de minería de datos en dicha plataforma.

R nació como parte de un proyecto colaborativo y abierto. Sus usuarios pueden publicar paquetes por el mundo que extienden su configuración básica. Existe

un repositorio oficial de paquetes cuyo número superó en otoño de 2009 la cifra de los 2000. Dado el enorme número de nuevos paquetes, éstos se han clasificado en vistas (o temas), que permiten agruparlos según su naturaleza y función.

7.2. Implementación del algoritmo elusivo

A la hora de elegir el número de RPAs que definen el enjambre, se ha asumido una postura conservativa con únicamente ocho vehículos remotos por enjambre. Ello es debido a que cuanto mayor sea el número de RPAs que formen parte del enjambre, mayores recursos computacionales de cálculo se necesitarán, los cuales únicamente se emplearan en el cálculo dinámico de las aeronaves remotas que no aportará relevancia a la investigación. Por el contrario, es preferible reservar recursos computacionales al desarrollo del algoritmo tendente a estudiar la inteligencia del enjambre, lo cual enriquecerá la investigación. Ocho RPAs por enjambre es una solución conservativa que, como se verá posteriormente en el conjunto de simulaciones efectuadas, no limita el concepto de enjambre, manteniendo la propia definición del enjambre.

En cuanto a la configuración inicial del enjambre se ha supuesto que el enjambre se desplaza en formación hasta la zona de conflicto donde en el tiempo t=0 acomete la maniobra bien de saturación de defensas, bien ISR. La geometría adoptada para el enjambre en vuelo en formación ha sido un cuadrado por simplicidad como puede observarse en la figura.

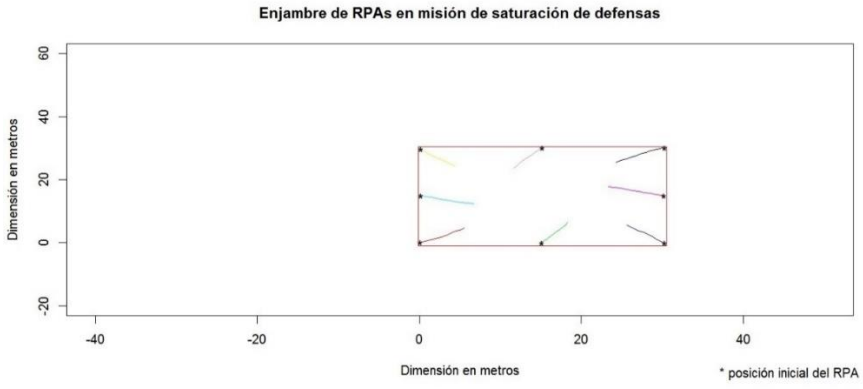

Figura 7.3. Posición inicial del enjambre.

El centroide se localiza en el centro geométrico del enjambre. Respecto a las velocidades iniciales de los RPAs del enjambre, se debe tener en cuenta que el objetivo posterior es definir la entropía y que esta sea máxima (como se verá posteriormente); sin embargo, dependiendo de las direcciones de los vectores velocidad iniciales de los RPAs, la entropía puede tardar un cierto tiempo en llegar al máximo y a efectos de análisis e investigación se ha considerado más interesante que la entropía tarde un tiempo corto en alcanzar este óptimo, como se estudiará más adelante. Con el elevado número de simulaciones efectuadas, se ha determinado una configuración inicial de vectores de velocidad unitarios, a los que se multiplica por un escalar para alcanzar la velocidad definitiva, con los que se ha estimado es posible alcanzar la entropía de trabajo en un corto periodo de tiempo.

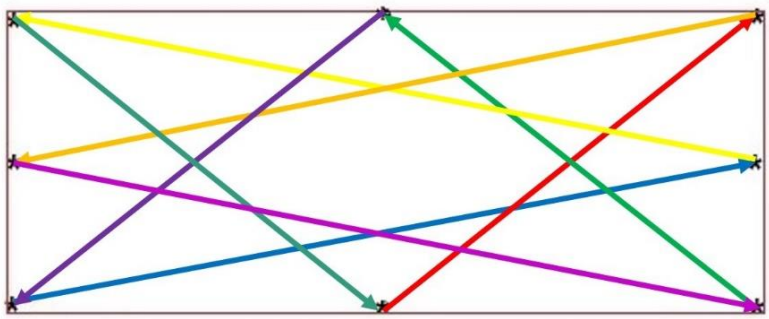

Figura 7.4. Esquema de velocidades iniciales del enjambre.

A continuación se muestran algunas de las simulaciones efectuadas aplicando el entorno r-project, donde los códigos de cálculo más representativos se encuentran al final del documento en los Apéndices. Los parámetros de cálculo se indican a continuación:

- tamaño promedio de los RPAs=1 metro
- aceleración=5 m/s^2
- deceleración=1.5 m/s^2
- tiempo de la trayectoria completa=60 segundos
- paso de tiempo=0.2 segundos
- velocidad de crucero=velcru (m/s)

- velocidad de pérdida=velper (m/s)
- velocidad angular de viraje=velang (grados sexagesimales/segundo)
- factdeca
- factdecv
- factMmmm

Simulación para 4m/s:

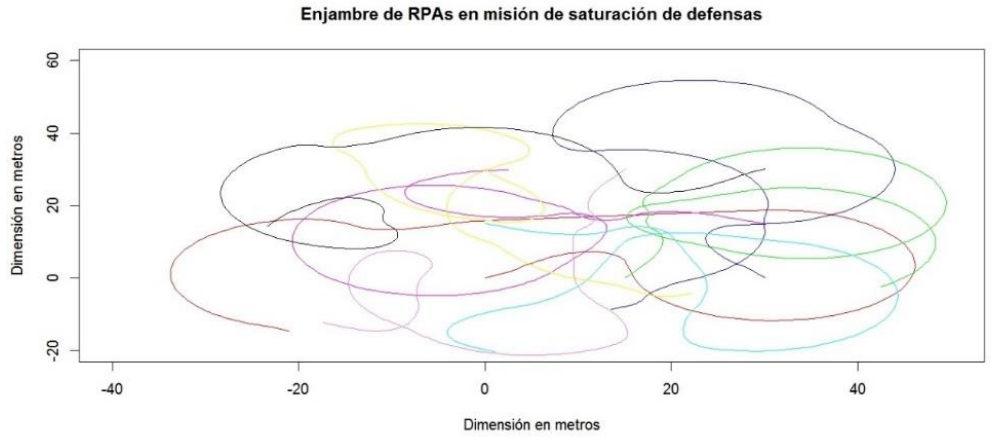

Figura 7.5. velcru=4, velper=1, velang=15, factdeca=7, factdecv=10, factMmmm=15.

Con respecto a esta simulación, se ha calculado la secuencia de tiempos:

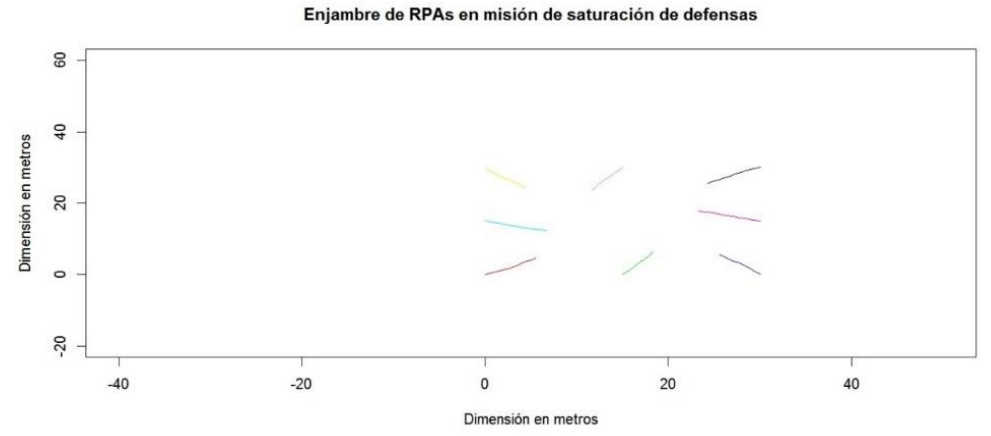

Figura 7.6. velcru=4, velper=1, velang=15, factdeca=7, factdecv=10, factMmmm=15, tiempo=2 seg.

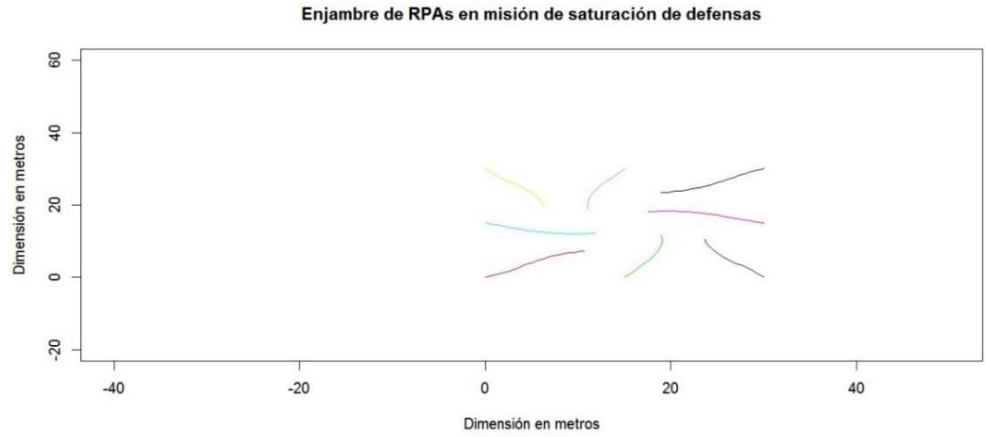

Figura 7.7. velcru=4, velper=1, velang=15, factdeca=7, factdecv=10, factMmmm=15, tiempo=4 seg.

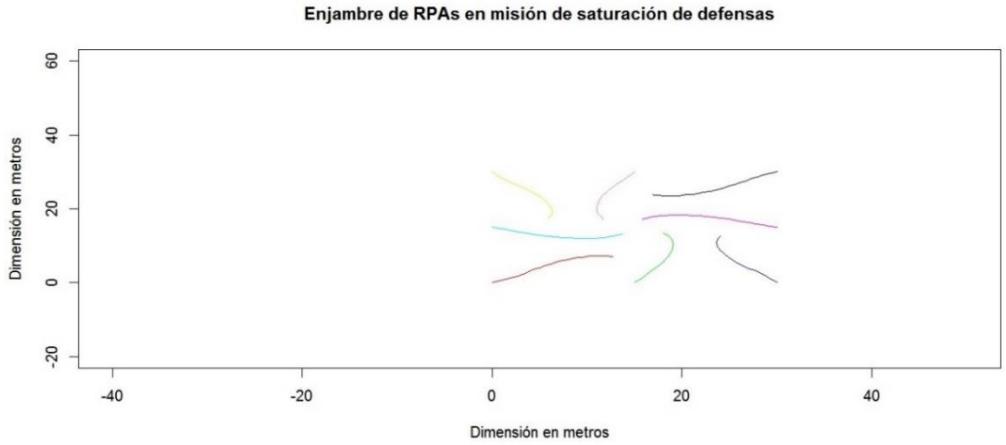

Figura 7.8. velcru=4, velper=1, velang=15, factdeca=7, factdecv=10, factMmmm=15, tiempo=6 seg.

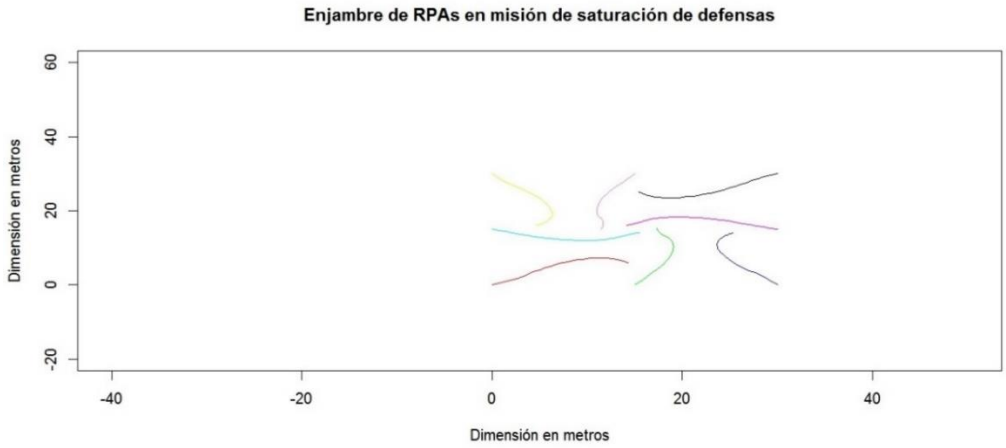

Figura 7.9. velcru=4, velper=1, velang=15, factdeca=7, factdecv=10, factMmmm=15, tiempo=8 seg.

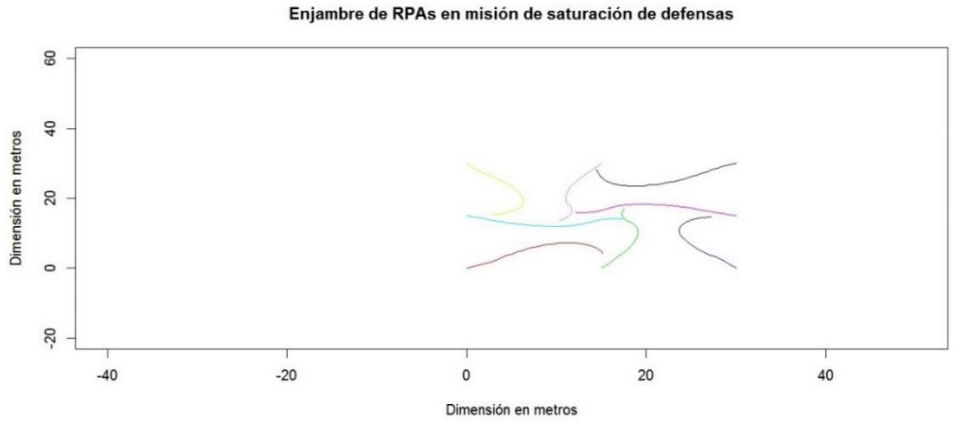

Figura 7.10. velcru=4, velper=1, velang=15, factdeca=7, factdecv=10, factMmmm=15, tiempo=10 seg.

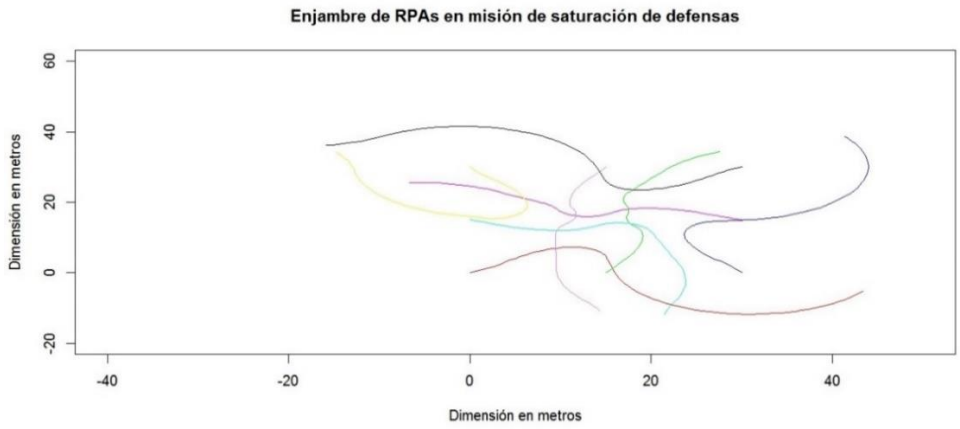

Figura 7.11. velcru=4, velper=1, velang=15, factdeca=7, factdecv=10, factMmmm=15, tiempo=20 seg.

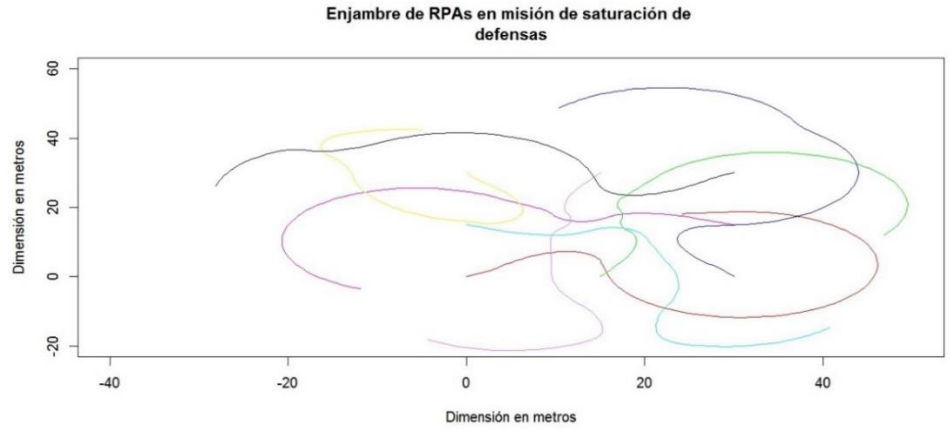

Figura 7.12. velcru=4, velper=1, velang=15, factdeca=7, factdecv=10, factMmmm=15, tiempo=30 seg.

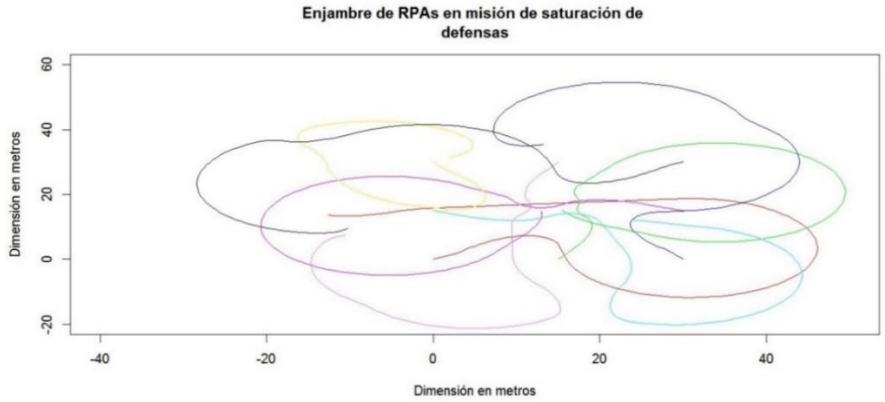

Figura 7.13. velcru=4, velper=1, velang=15, factdeca=7, factdecv=10, factMmmm=15, tiempo=40 seg.

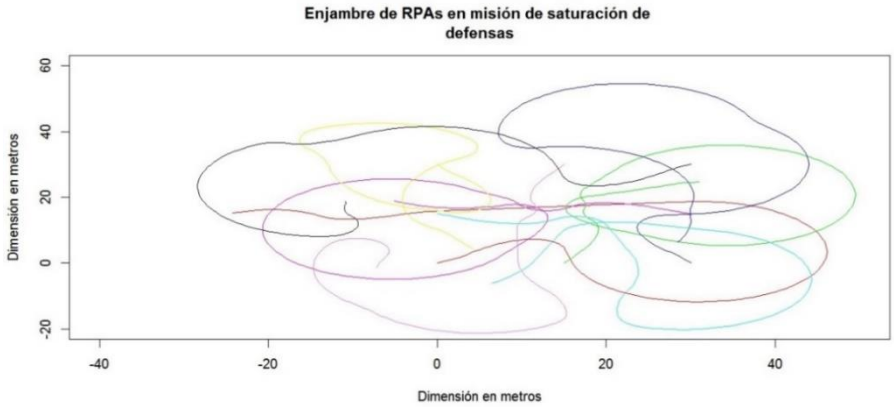

Figura 7.14. velcru=4, velper=1, velang=15, factdeca=7, factdecv=10, factMmmm=15, tiempo=50 seg.

Incrementando la velocidad a 10 m/s:

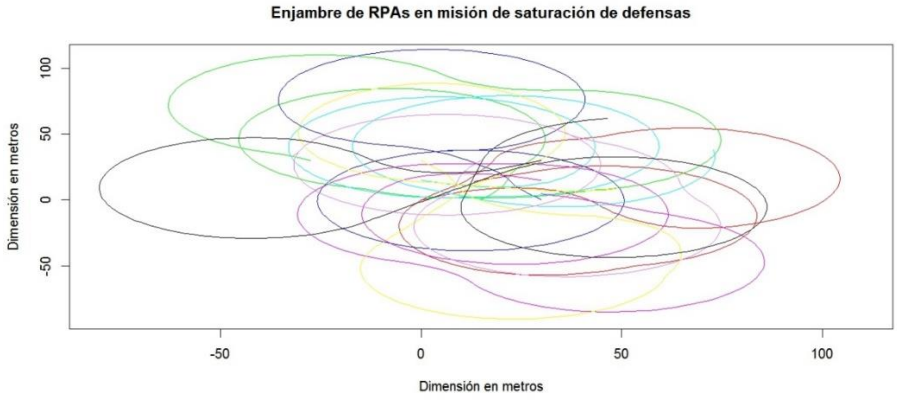

Figura 7.15. velcru=10, velper=3, velang=15, factdeca=7, factdecv=10, factMmmm=15.

Con velocidad a 15 m/s, variando la velocidad angular de viraje y factMmmm:

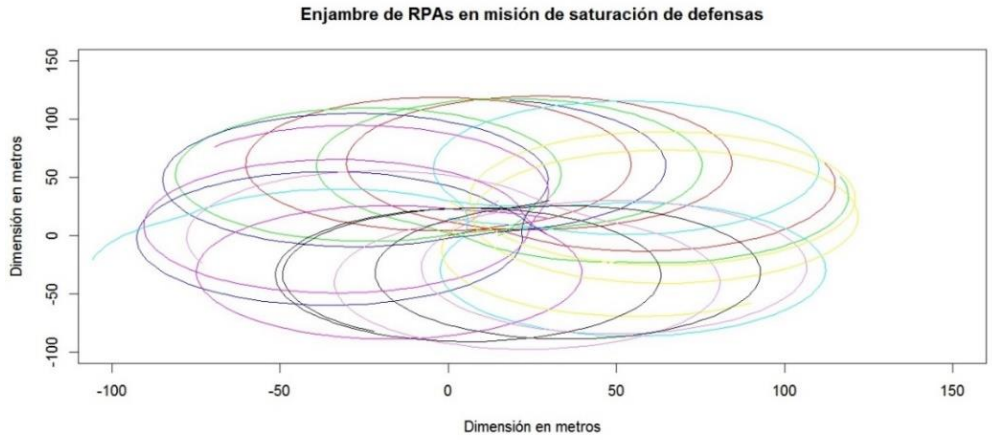

Figura 7.16. velcru=15, velper=6, velang=15, factdeca=7, factdecv=10, factMmmm=15.

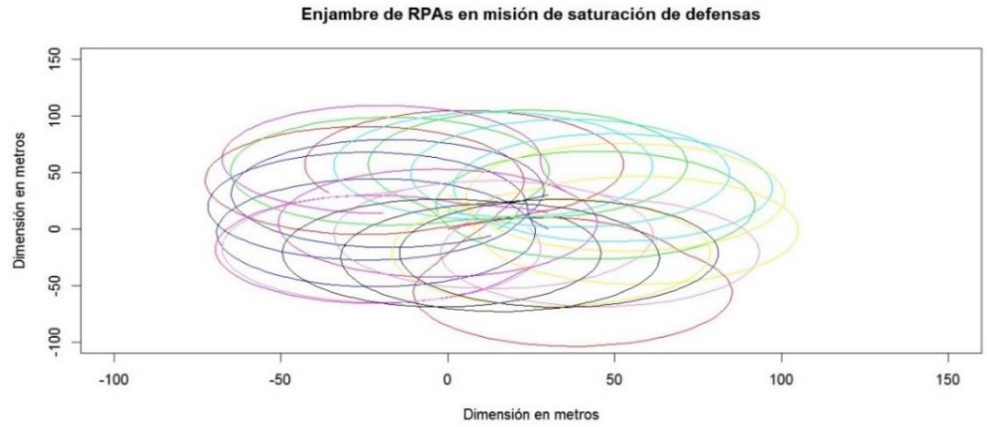

Figura 7.17. velcru=15, velper=6, velang=18, factdeca=7, factdecv=10, factMmmm=12.

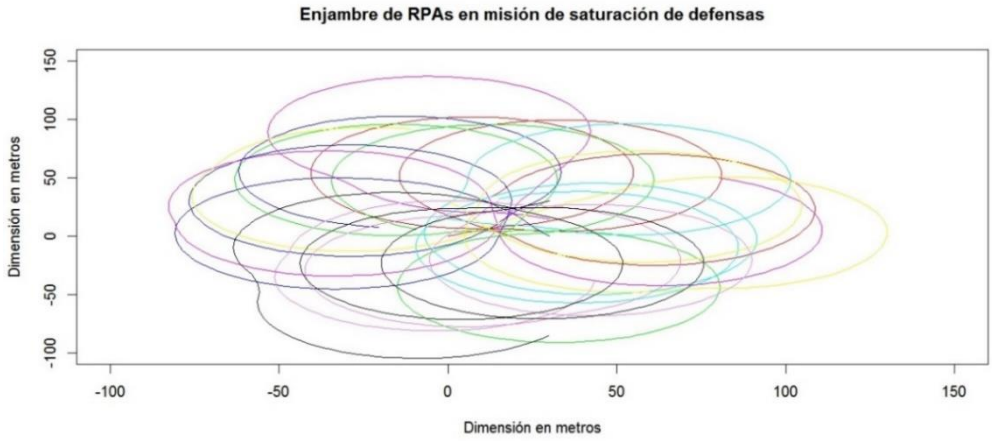

Figura 7.18. velcru=15, velper=6, velang=18, factdeca=7, factdecv=10, factMmmm=15.

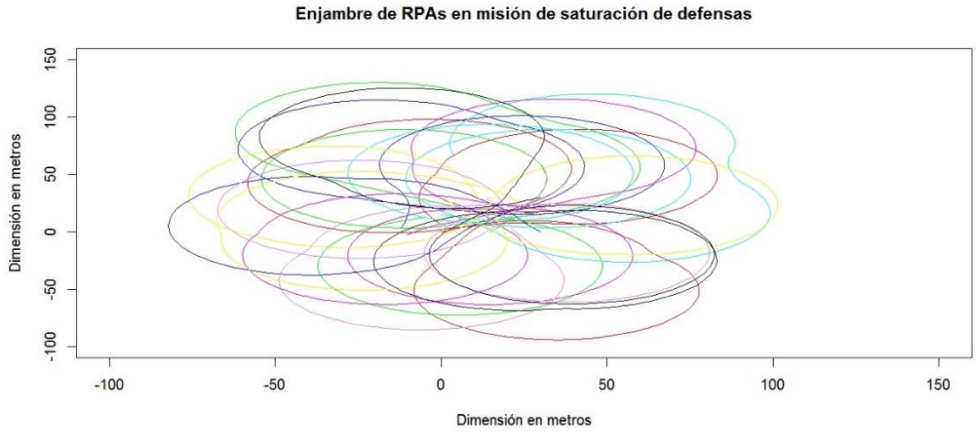

Figura 7.19. velcru=15, velper=6, velang=20, factdeca=7, factdecv=10, factMmmm=15.

Simulación para 17 m/s:

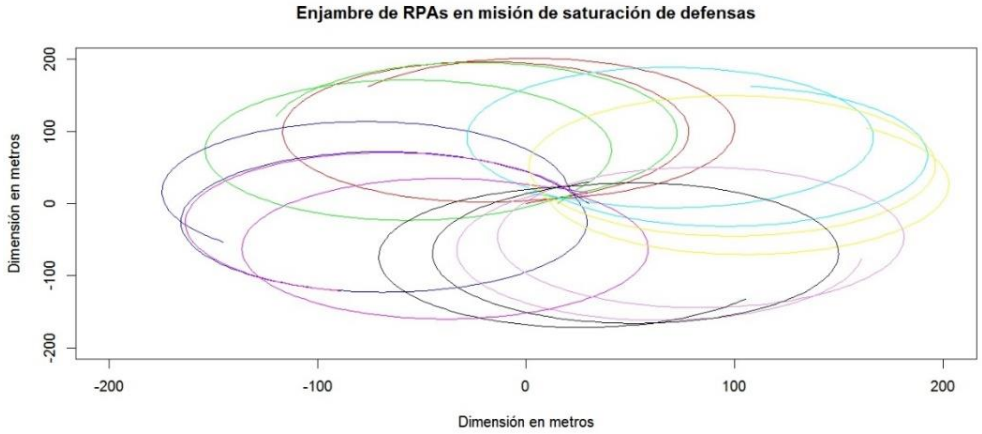

Figura 7.20. velcru=17, velper=6, velang=10, factdeca=7, factdecv=10, factMmmm=15.

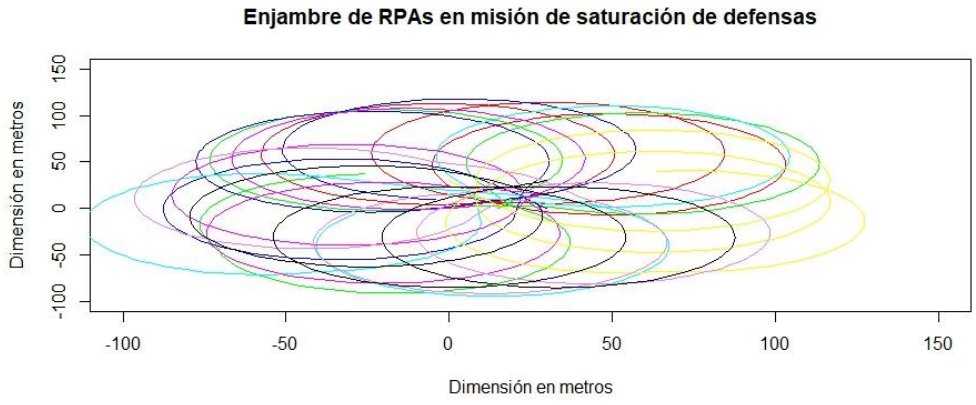

Figura 7.21. velcru=17, velper=6, velang=18, factdeca=7, factdecv=10, factMmmm=15.

Con velocidad a 18 m/s, variando la velocidad angular de viraje y factMmmm:

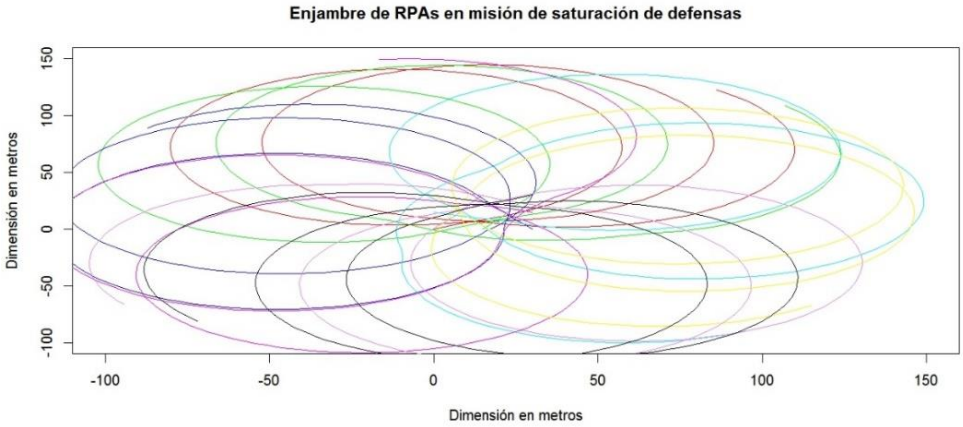

Figura 7.22. velcru=18, velper=6, velang=15, factdeca=7, factdecv=10, factMmmm=15.

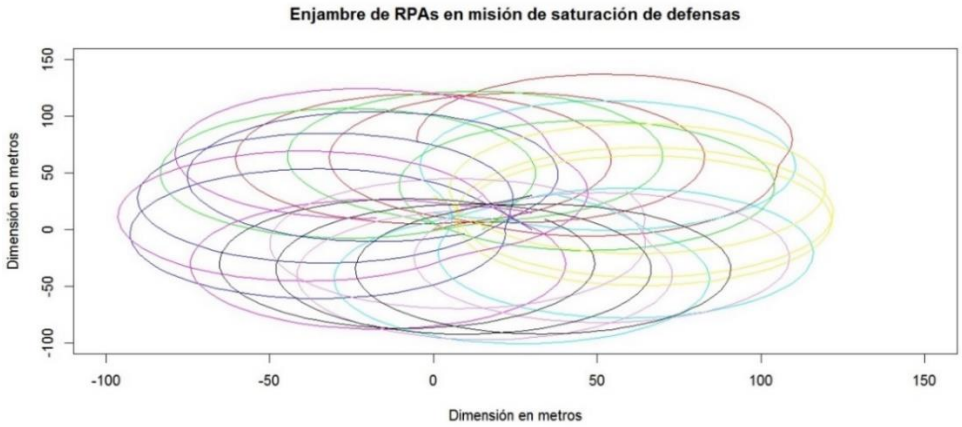

Figura 7.23. velcru=18, velper=6, velang=18, factdeca=7, factdecv=10, factMmmm=12.

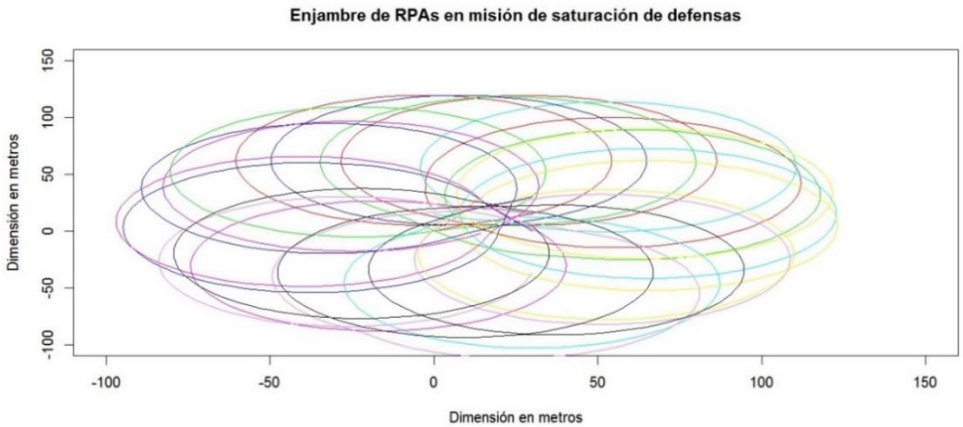

Figura 7.24. velcru=18, velper=6, velang=18, factdeca=7, factdecv=10, factMmmm=15.

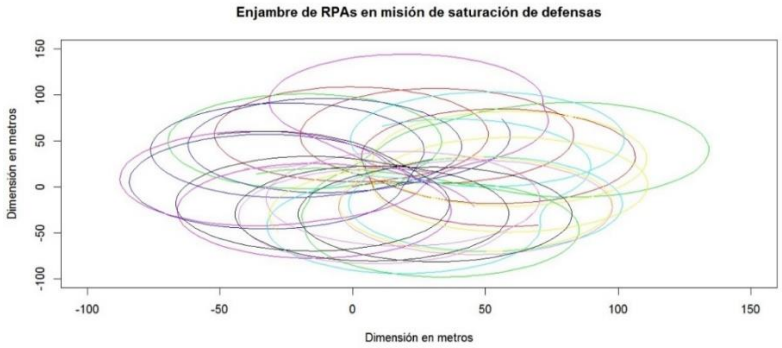

Figura 7.25. velcru=18, velper=6, velang=20, factdeca=7, factdecv=10, factMmmm=15.

El área barrida por el vector de posición de todos los RPAs del enjambre durante el tiempo de la trayectoria completa define un área barrida por el enjambre (ABE) completa que puede ser muy útil en el caso de misiones ISR y saturación de defensas. Así, atendiendo a los análisis efectuados anteriormente se extrae que ABE aumenta al incrementarse la velocidad de crucero, aumentar factMmmm o al disminuir la velocidad angular de viraje, siendo poco sensible a factdeca y factdecv.

Figura 7.26. Área barrida por el enjambre.

En este sentido, con las simulaciones efectuadas haciendo uso de los algoritmos desarrollados en este trabajo, se ha calculado la distancia mínima entre dos RPAs en sucesivas simulaciones, obteniendo valores del orden de 3,9 el tamaño promedio del RPA, lo cual estaría en consonancia con el valor inferior al que se llegó en las investigaciones del KAIST. Respecto al valor superior de 15 del KAIST, este dato se considera no así para impedir las colisiones entre RPAs sino más bien para imposibilitar que el enjambre se disperse y disgregue perdiendo sinergia.

Capítulo 8. Entropía del enjambre

En termodinámica, la entropía se define como una magnitud física para un sistema termodinámico en equilibrio. Esta magnitud mide el número de microestados compatibles con el macroestado de equilibrio, también se afirma que mide el grado de organización del sistema, o bien el grado de desorden.

Descendiendo a la teoría de la información, la denominada entropía de la información o también de Shannon (en honor al investigador Claude E. Shannon), calcula la incertidumbre de una fuente de información. También se puede considerar como la cantidad de información promedio contenida dentro de los símbolos empleados en la transmisión de la información. Así, los símbolos con probabilidad más baja son los que poseen mayor información. Cuando todos los símbolos son equiprobables, todos aportan idéntica información y consecuentemente la entropía es máxima.

En todos los libros de termodinámica la entropía se considera como una medida del desorden del sistema. Esta relación surge de la mecánica estadística dado que la termodinámica no es capaz de establecer esta relación por sí misma, al no preocuparse por los estados microscópicos. Ciertamente, el concepto entropía se emplea en termodinámica, mecánica estadística y teoría de la información. En todos los casos la entropía tiene algo en común, se concibe como una medida del desorden. La entropía puede interpretarse como una forma de medir y evaluar la incertidumbre, y también la información necesaria para, en un proceso, medir el grado de desorden. Evidentemente el concepto de información y el de entropía están relacionados intrínsecamente entre sí, aunque esto no fuera obvio en los comienzos.

Pero realmente es una cuestión de opciones disponibles que producen el máximo número de posibilidades. Sin ajustarse al mundo de lo pequeño, este fenómeno es conocido a cualquier escala, pero esta visión de desorden no es una buena idea y resulta conveniente empezar a asimilar la definición estadística de la entropía donde la entropía de un sistema es proporcional al número de estados posibles.

Descendiendo al mundo de los enjambres en misión de saturación de defensas, surge la cuestión de cómo el enjambre generará el mayor efecto saturador, degradativo, y neutralizante en el sistema defensivo enemigo potenciando el desconcierto, la confusión, la decepción y el colapso. Indudablemente, un elevado número de RPAs produciría más confusión que un número bajo. Pero aunque el número de RPAs fuera alto, si todos los RPAs tuvieran la misma velocidad en dirección y módulo, es claro que el desconcierto generado sería muy similar a un único RPA. Así, evidentemente, un enjambre de RPAs sorprenderá, desconcertará y abrumará en mayor medida incrementando el efecto degradativo, saturador y neutralizante cuanto más RPAs haya volando a la vez sobre el espacio aéreo enemigo, con velocidades muy diferentes tanto en módulo como en vector. Así, resulta claro que cuantos más RPAs haya volando, cada uno en una dirección diferente, más trabajo tendrá el sistema enemigo de defensa aérea.

Esto enlazaría con la definición de entropía, en relación al número de estados posible, que para el caso de enjambres sería el número de RPAs con velocidades diferentes tanto en dirección como en módulo. El proceso de cálculo sería el siguiente:

1. Para cada dimensión del espacio en que se define el problema, se toma el menor y el mayor valor de la componente de la velocidad de todos los RPAs del enjambre. Como $V_{RPAz} \ll V_{RPAx}, V_{RPAy}$, se consideran únicamente las componentes x e y, y por tanto $V_{RPAxmenor}$ y $V_{RPAxmayor}$, así como $V_{RPAymenor}$ y $V_{RPAymayor}$. El resto de V_{RPAix} y V_{RPAiy} estarán comprendidas entre $V_{RPAxmenor} < V_{RPAix} < V_{RPAxmayor}$ y $V_{RPAymenor} < V_{RPAiy} < V_{RPAymayor}$.
2. Se define una separación entre $V_{RPAxmenor}$ y $V_{RPAxmayor}$, así como $V_{RPAymenor}$ y $V_{RPAymayor}$, función del número de RPAs que componen el

enjambre: $\mathrm{d}_{RPAx} = (\mathrm{V}_{RPAxmayor} - \mathrm{V}_{RPAxmenor})/n\acute{u}meroRPAS$ y $\mathrm{d}_{RPAy} = (\mathrm{V}_{RPAymayor} - \mathrm{V}_{RPAymenor})/n\acute{u}meroRPAS$.

3. Se definen los intervalos

$$\begin{gathered}
[\mathrm{V}_{RPAxmenor,}\mathrm{V}_{RPAxmenor} + 1 \cdot \mathrm{d}_{RPAx}[\\
[\mathrm{V}_{RPAxmenor} + 1 \cdot \mathrm{d}_{RPAx,}\mathrm{V}_{RPAxmenor} + 2 \cdot \mathrm{d}_{RPAx}[\\
[\mathrm{V}_{RPAxmenor} + 2 \cdot \mathrm{d}_{RPAx,}\mathrm{V}_{RPAxmenor} + 3 \cdot \mathrm{d}_{RPAx}[\\
\dots \\
[\mathrm{V}_{RPAymenor,}\mathrm{V}_{RPAymenor} + 1 \cdot \mathrm{d}_{RPAy}[\\
[\mathrm{V}_{RPAymenor} + 1 \cdot \mathrm{d}_{RPAy,}\mathrm{V}_{RPAymenor} + 2 \cdot \mathrm{d}_{RPAy}[\\
[\mathrm{V}_{RPAymenor} + 2 \cdot \mathrm{d}_{RPAy,}\mathrm{V}_{RPAymenor} + 3 \cdot \mathrm{d}_{RPAy}[\\
\dots
\end{gathered} \tag{8.1}$$

Y se contabilizan el número de RPAs cuya velocidad correspondiente pertenece a cada intervalo. La función Índice de Entropía de cada Intervalo (IEI) toma el valor 1 si al menos la velocidad de un RPA pertenece a dicho intervalo y toma el valor 0 si no existe RPA alguno cuya velocidad pertenezca al intervalo correspondiente.

Obviamente el número de intervalos existentes será 2 veces el número de RPAs (1 vez por la componente x y otra vez por la componente y). Sumando para todos los intervalos, que son 2 veces el número de RPAs, la función IEI, se obtiene la función Entropía del Enjambre (EE): $\mathrm{EE} = \sum_{2 \cdot n\acute{u}mero\ de\ RPAS} IEI$.

Por su propia definición $0 \leq \mathrm{EE} \leq 2 \cdot n\acute{u}mero\ de\ RPAS$, definiendo el factor de Entropía del Enjambre como $\mathrm{fEE} = \mathrm{EE}/(2 \cdot n\acute{u}mero\ de\ RPAS)$, con lo que $0 \leq \mathrm{fEE} \leq 1$.

En los cálculos efectuados, este desarrollo del concepto de entropía realizado anteriormente se ha denominado Entropía A, EEA.

Por otra parte, también se ha trabajado con la denominada Entropía B, la cual se ha definido como: $\mathrm{EEB} = \sum_{n\acute{u}mero\ de\ RPAS} m\acute{o}dulo\ de\ velocidad\ RPAS_i$.

Análogamente, por su propia definición $0 \leq \mathrm{EEB} \leq n\acute{u}mero\ de\ RPAS \cdot velocidad\ de\ crucero$, definiendo el factor de Entropía del Enjambre como $\mathrm{fEEB} = \mathrm{EEB}/(n\acute{u}mero\ de\ RPAS \cdot velocidad\ de\ crucero)$, con lo que $0 \leq \mathrm{fEEB} \leq 1$.

Invirtiendo el orden y empezando por la Entropía B, concretamente fEEB. Los análisis efectuados con esta entropía no han sido muy relevantes. Para 4 m/s se tiene:

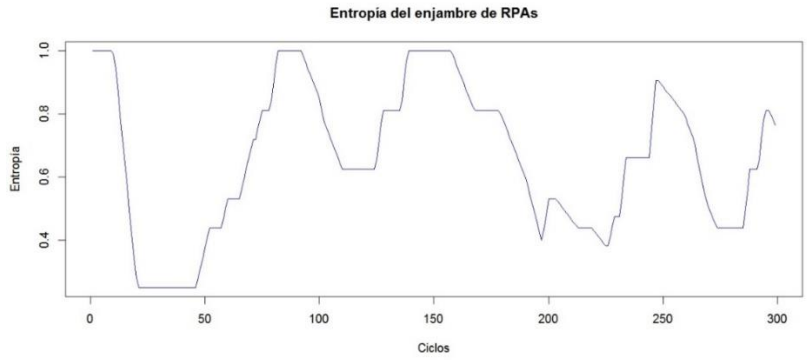

Figura 8.1. Entropía B, velcru=4, velper=1, velang=15, factdeca=7, factdecv=10, factMmmm=15.

El número de ciclos se corresponde con el tiempo de la trayectoria/paso de tiempo en segundos, que para el instante final resulta ser el tiempo de la trayectoria completa/paso de tiempo=300 ciclos.

Incrementando la velocidad a 10 m/s y 17 m/s respectivamente:

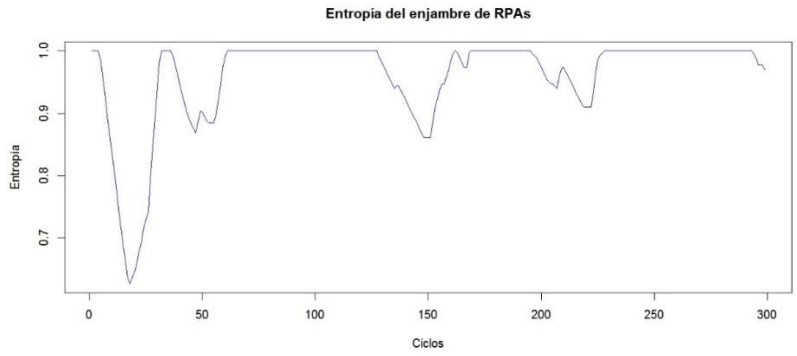

Figura 8.2. Entropía B, velcru=10, velper=3, velang=15, factdeca=7, factdecv=10, factMmmm=15.

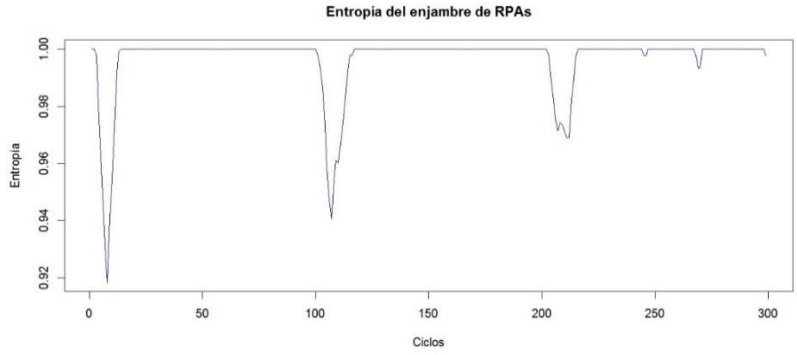

Figura 8.3. Entropía B, velcru=17, velper=6, velang=18, factdeca=7, factdecv=10, factMmmm=15.

Esta entropía fEEB presenta un perfil muy uniforme con pocos altibajos y variaciones, más al aumentar la velocidad de crucero. Ello es debido a que, como se ha comentado anteriormente, los RPAs del enjambre tienden a mantener la velocidad de crucero, como ocurre generalmente en la realidad, exceptuando situaciones puntuales de virajes cerrados, subidas, etc. Así, esta definición de entropía no aporta riqueza dado que el número de estados posible se mantiene cuasi-constante durante la mayor parte del tiempo completo de la trayectoria, con lo que con esta entropía el desconcierto generado por el enjambre sería muy similar a un único RPA.

En el caso de la entropía fEEA, para 4 m/s se tiene:

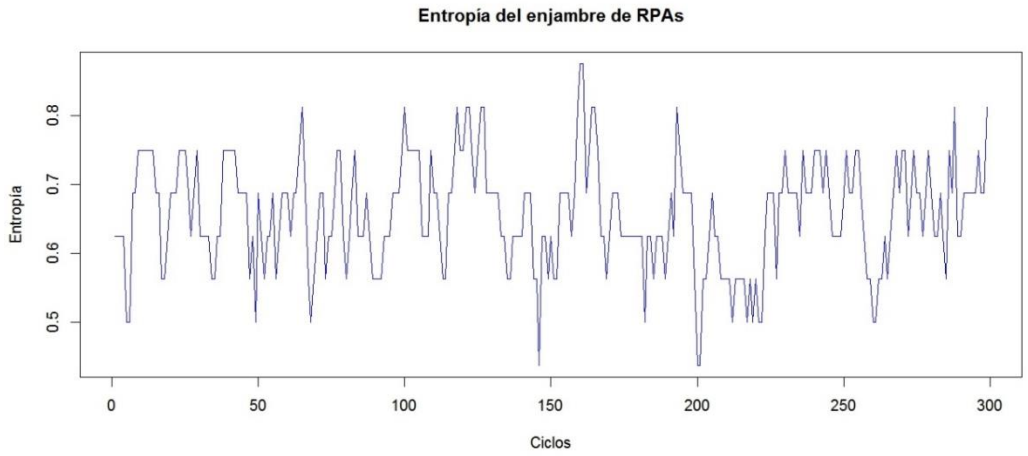

Figura 8.4. Entropía A, velcru=4, velper=1, velang=15, factdeca=7, factdecv=10, factMmmm=15.

Incrementando la velocidad a 10 m/s y 17 m/s respectivamente:

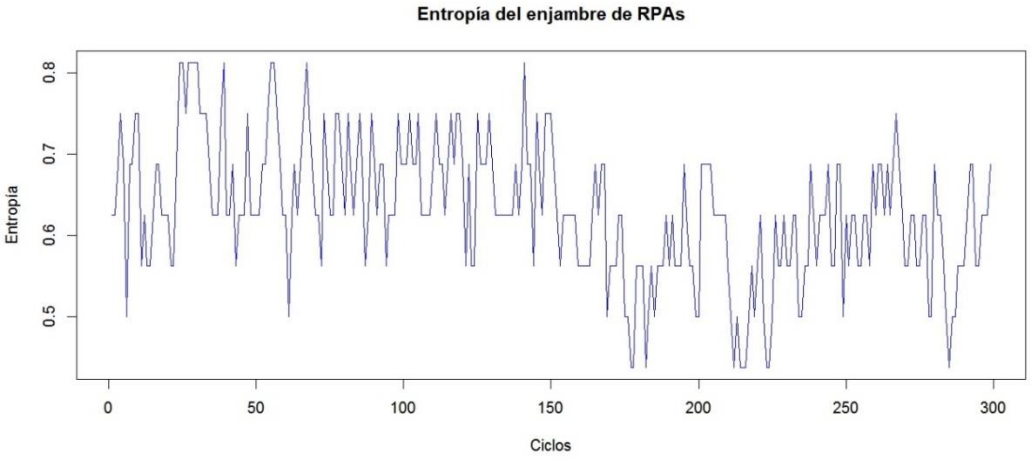

Figura 8.5. Entropía B, velcru=10, velper=3, velang=15, factdeca=7, factdecv=10, factMmmm=15.

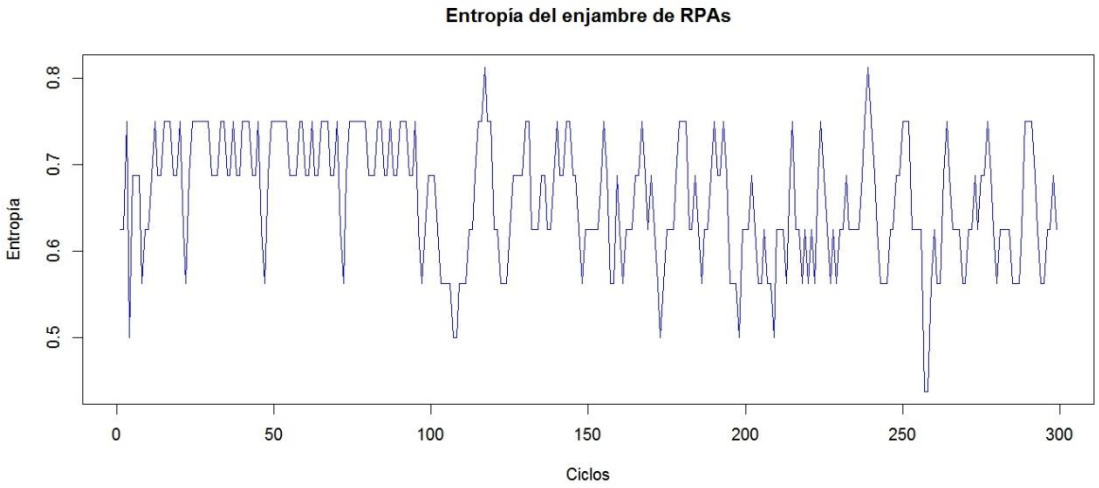

Figura 8.6. Entropía B, velcru=17, velper=6, velang=18, factdeca=7, factdecv=10, factMmmm=15.

De estas gráficas, se desprende que el fEEA se sitúa entre 0.4 y 0.9 aunque los valores más frecuentes son 0.7; es decir, el 70% del enjambre presenta vectores de velocidad diferentes. No obstante, a la hora de encajar el concepto de entropía dentro de un algoritmo de optimización, como función a maximizar, lo cual se tratará en los siguientes capítulos, resulta más rentable explorar otro concepto de entropía donde este algoritmo optimizante trabaje mejor.

Para ello, es necesario aumentar el peso de la entropía, lo cual se consigue explorando el concepto de entropía como un sumatorio de EEA para el tiempo de vuelo completo (300 ciclos para 60 segundos) donde pequeñas diferencias en cada ciclo, al final del tiempo de la trayectoria completa, proporcionan grandes diferencias en el proceso de optimización. Esta es la entropía que se empleará en el resto del trabajo denominándose entropía global del enjambre.

Capítulo 9. Optimización

La optimización matemática engloba un conjunto de técnicas de modelado matemático que tienen como objetivo encontrar el óptimo global de cualquier problema, dando una respuesta de este modo a la cuestión de resolución de problemas de asignación o situación óptima de recursos escasos y, en general, respaldando de una forma eficaz el proceso de toma de decisiones.

Desgraciadamente, solo en ciertos casos se puede garantizar la convergencia hacia el óptimo global del problema a tratar. A este respecto, las técnicas convencionales solo pueden localizar óptimos locales lo que conlleva que es necesario recurrir a la denominada optimización no lineal. No obstante, estas técnicas no aseguran la convergencia hacia el óptimo general, a menos que se empleen técnicas exhaustivas o tiempos de cálculo muy elevados que comprometen los recursos computacionales disponibles. De hecho, es bastante difícil distinguir entre un óptimo global y un óptimo local muy cercano al global.

El término heurístico desciende de la palabra griega heuriskein que viene a significar algo similar a encontrar. Así, un algoritmo heurístico es un procedimiento de búsqueda de soluciones casi óptimas con un coste computacional asequible sin que sea posible garantizar la optimalidad de la solución encontrada ni determinar a qué distancia se encuentra el óptimo global (Reeves, 1995). En ciencias de la computación se buscan algoritmos con tiempos de ejecución buenos y soluciones óptimas. Una heurística es algún tipo de técnica que no satisface ambos requerimientos; por ejemplo, se encuentran buenas soluciones sin una prueba fehaciente de que sea la óptima o se implementa muy rápidamente el proceso sin tener la certeza que permita aseverar que siempre será así. Estas técnicas permiten abordar problemas

complejos de forma estimada, son flexibles y permiten encontrar soluciones de buena calidad en tiempos computacionales razonables.

Disponer de herramientas que permitan ofrecer soluciones rápidas a problemas reales ha espoleado estas técnicas de resolución de problemas combinatorios debido a dos circunstancias primordiales: encontrar “buenas soluciones”, aunque no se pueda comprobar que sean óptimas, a un coste asequible; es decir, de una forma razonablemente rápida. Estos problemas de optimización combinatoria suponen un importante reto debido a su complejidad matemática, ya que esta crece exponencialmente con el tamaño del problema.

Un paso más allá se tiene las metaheurísticas o estrategias inteligentes que mejoran los procedimientos heurísticos con un mayor rendimiento (Glover y Kochenberger 2003). Realmente se trata de armonizar inteligentemente varias técnicas para explorar el espacio de soluciones como describen Osman y Kelly (1995) "Los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son ni efectivos ni eficientes. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de: inteligencia artificial, evolución biológica y mecanismos estadísticos".

Aunque estas técnicas abarcan una cantidad enorme en cuanto a su tratamiento, todas ellas tratan de conjugar una búsqueda intensiva seleccionando movimientos que mejoren la solución, y por otra parte diversificando, o lo que es lo mismo, aceptando otras soluciones peores pero que permiten la evasión de los óptimos locales. Esto es debido a que la mayor parte de los problemas reales tienen un espacio de soluciones enorme que es imposible de explorar en detalle por lo que estas técnicas obvian los óptimos locales para no quedar atrapados en ellos, y de este modo emprender la búsqueda del óptimo global.

El imponente aumento del número de variables de los que dependen las funciones a optimizar ha hecho difícil aplicar las técnicas tradicionales de optimización por lo que estos nuevos algoritmos heurísticos han encontrado su

caldo de cultivo, renunciado a parte de la exactitud a cambio de una mayor rapidez.

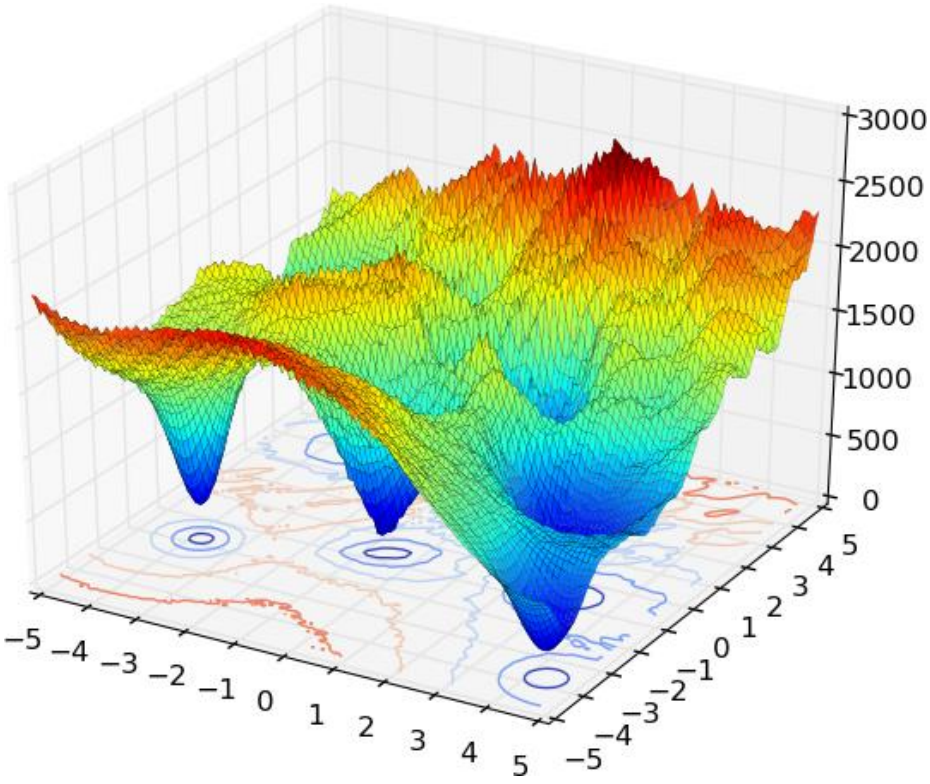

Figura 9.1. Función con múltiples óptimos.

Un problema de optimización se formula de la siguiente manera:

$$\min\{f(x) / x \in X, X \subseteq S\} \qquad (9.1)$$

Siendo S el espacio de soluciones, X la región factible, x una solución factible y f una función real. Si S es finito, el problema es de optimización combinatoria, mientras que si $S=IR^n$ se denomina de optimización continua. Un problema de optimización se representa por el par (S,f), donde S es espacio de soluciones y f la función objetivo.

Los problemas muy complejos son “difíciles de resolver“, designándoles Problemas NP, clasificándose en clases de complejidad (L, NL, P, P Completo, NP, NP-Completo, NP Duro,...). En teoría de la complejidad la mayoría de los problemas se refieren a problemas de la clase NP tales que no se conoce ningún algoritmo que pueda resolverlos exactamente y tienen complejidad polinomial, cuyo tiempo de resolución crece polinomialmente con el tamaño del problema. En este tipo de casos, los métodos heurísticos se plantean como técnicas eficientes para obtener soluciones cercanas al valor óptimo de la

función objetivo en un tiempo razonable. Este tipo de algoritmos heurísticos serían útiles en las siguientes situaciones:

- No se conoce un procedimiento exacto de resolución, o bien éste requiere un gran esfuerzo computacional.
- No es necesario obtener la solución óptima global del problema siendo suficiente con conocer únicamente una solución cercana a dicho óptimo.
- Se prefiere resolver aproximadamente un modelo ajustado a la realidad que resolver exactamente un modelo aproximado a la realidad, aunque el tiempo de resolución sea similar.
- No se dispone del suficiente conocimiento específico del problema como para diseñar un método exacto de resolución, o bien el tiempo que requeriría esta acción es inaceptable.

Las principales ventajas pueden resumirse como:

- Permiten una gran flexibilidad en el manejo y control de las características del problema permitiendo incorporar fácilmente condiciones difíciles de modelizar.
- Generalmente, ofrecen varias soluciones alternativas permitiendo la capacidad de elección.
- Suele ser más fácil de entender la fundamentación de las heurísticas que los complejos métodos matemáticos que utilizan técnicas exactas (muchas variables, fuerte no linealidad,…).

Siendo las desventajas:

- No siempre es posible conocer la bondad de la solución obtenida, siendo necesario realizar acotaciones con relajaciones, o bien generar varias soluciones y compararlas con la obtenida.
- Dependencia de la estructura del problema considerado y falta de habilidad para adaptarse a nuevas situaciones o modificaciones del problema de partida.

Para resolver el segundo de los inconvenientes indicados, se recomienda disponer de procedimientos heurísticos de propósito general con capacidad de adaptación que puedan emplearse para resolver una amplia variedad de

problemas. Así, las técnicas metaheurísticas deben cumplir una serie de particularidades:

1. Adaptabilidad a los diferentes ámbitos de aplicación y a los diferentes casos del problema.
2. Autonomía; facilitar su funcionamiento automático global, libre de parámetros o ajustados automáticamente, en la mayor parte de las facetas que lo caracterizan.
3. Coherencia; los elementos característicos que los describen deben derivarse de forma natural de los principios originales que los inspiran.
4. Multiplicidad y diversidad; proporcionan diferentes soluciones alternativas de alta calidad de tal modo que el operador pueda elegir.
5. Eficacia; deben alcanzar soluciones óptimas o cercanas al óptimo para la mayoría o todas las situaciones realistas que pretendan resolver.
6. Eficiencia; deben realizar un buen aprovechamiento de las capacidades computacionales en términos de tiempo de ejecución, espacio de memoria y, en definitiva, de costos de desarrollo.
7. Facilidad de uso; deben estar expresadas de manera clara y con el menor número de parámetros posibles, siendo fáciles de comprender y de usar.
8. Generalidad; se pueden utilizar provechosamente y con buen rendimiento en una amplia gama de situaciones y problemas.
9. Independencia; no dependiente del marco o del agente tecnológico en el que se vaya a desarrollar.
10. Interactividad; el usuario es un apoyo externo que tiene capacidad de mejorar la solución y el rendimiento, a partir de su experiencia y conocimientos.
11. Robustez; el comportamiento debe ser sensible a pequeñas alteraciones del modelo consistente con una gran variedad de problemas y no debe estar únicamente diseñado para resolver un pequeño conjunto de ellos.
12. Simplicidad; deben basarse en principios simples y claros, siendo fáciles de comprender e implementar.
13. Concreción; las fases de la metaheurística deben formularse de forma concreta y determinada.

Capítulo 10. Particle Swarm Optimization

Este trabajo tiene su objeto principal de actuación en un enjambre de RPAs donde se hace uso de una serie de algoritmos tecnológicos relativos a búsqueda de estados, técnicas de optimización con componentes evolutivos y aprendizaje automático.

A este respecto el algoritmo de optimización empleado ha sido el de Enjambre de Partículas (PSO, Particle Swarm Optimization), una técnica estocástica basada en el movimiento de partículas con cierta inteligencia, que se inspira en el comportamiento social de poblaciones y se emplea para resolver problemas complejos de optimización.

El algoritmo PSO fue propuesto por James Kennedy y Rusell Eberhart en 1995 como una heurística de búsqueda poblacional basada en el comportamiento social y cognitivo de determinadas especies. Así, cada partícula representa una posible solución potencial dentro del espacio de búsqueda que se caracteriza por una posición, una velocidad y una memoria de su comportamiento anterior. Cuando la población es inicializada, a cada individuo se le asigna estocásticamente una velocidad inicial. En cada iteración del algoritmo, las velocidades son alteradas aleatoriamente, moviendo cada partícula hacia una nueva posición que depende de su mejor posición previa y de la mejor posición del conjunto.

En cada ciclo de vuelo, se evalúa la función objetivo para cada partícula en su actual posición. El valor que se obtiene mide la calidad de la partícula para el problema que se esté considerando. En el PSO original las partículas vuelan a

través del espacio de búsqueda condicionadas por dos factores: la mejor posición que la partícula alcanzó hasta ese momento y la mejor posición alcanzada por cualquier partícula de la población o swarm.

Paradójicamente, en este trabajo se tienen dos enjambres, el real constituido por un conjunto de RPAs y el de partículas que emplea el algoritmo PSO. Aunque parezca obvio, se debe recalcar que ambos enjambres son diferentes. Uno es el real formado por RPAs reales y el otro es el enjambre de partículas que emplea el algoritmo PSO y que es virtual y no real y se utilizar como algoritmo de optimización. Qué duda cabe que aunque diferentes, ambos enjambres comparten ciertas características que también hacen uso de una cierta inteligencia.

Estos enjambres se inspiran en la naturaleza, básicamente en sistemas biológicos, generalmente compuestos por una población de agentes básicos y simples que interactúan entre ellos y con el entorno. Una colonia de abejas u hormigas es el típico ejemplo que viene a la mente que se puede encontrar en la naturaleza. Los agentes individuales (abejas u hormigas) muestran individualmente muy poca inteligencia equivocándose repetidamente y cometiendo errores, como cuando tienen que encontrar un camino hasta los alimentos. Empero, estos individuos se comunican entre ellos de alguna forma singular (danza en las abejas, feromona en las hormigas) de tal forma que el enjambre es capaz de encontrar la solución más óptima al problema, o sea el camino más corto hacia la fuente de alimento. De este modo, la inteligencia del enjambre se conjuga como múltiples interacciones entre un grupo de individuos discriminando la solución óptima entre muchas soluciones menos eficientes. Esta inteligencia suele estar caracterizada porque los elementos individuales son autónomos y están distribuidos sin una autoridad central que determine las acciones de cada individuo.

El enjambre, en busca del óptimo global de una función objetivo, comienza esparciéndose por el espacio de búsqueda según reglas matemáticas que tienen en cuenta la posición y la velocidad de las partículas, aunque con el paso del tiempo las partículas convergen hacia una zona reducida del espacio

en la que se intensifican la exploración de la solución. El movimiento de cada partícula viene condicionado de una forma operativa por su mejor posición local hallada hasta el momento concreto, así como por las mejores posiciones globales descubiertas por otras partículas a medida que se recorre el espacio de búsqueda. El algoritmo que gestiona el movimiento del enjambre posee parámetros propios (comunicación, entorno, penalización) y otros comunes a los algoritmos evolutivos (tamaño de población, inicialización, criterios de parada,…).

PSO es una metaheurística como tal, ya que asume muy pocas o bien ninguna hipótesis sobre el problema a optimizar y puede aplicarse a grandes espacios de búsqueda. No obstante, como cualquier otra metaheurística, PSO no garantiza la obtención de una solución óptima en todos los casos ni en qué tiempo de cálculo.

10.1. Implementación del algoritmo PSO

Lo primero que necesita el algoritmo PSO es una población que se suele denominar nube o enjambre de soluciones candidatas (partículas). Este grupo de partículas se mueven a lo largo del espacio de búsqueda conforme a unas reglas matemáticas. El movimiento de cada partícula es función de su mejor posición alcanzada, y por otra parte de la mejor posición global hallada por el algoritmo en todo el espacio de búsqueda. Este algoritmo va descubriendo nuevas y mejores posiciones, las cuales pasan a reorientar los movimientos futuros de las partículas. Este proceso se va repitiendo hasta encontrar una solución satisfactoria.

Considérese una función a minimizar $f:\mathbb{R}^n \rightarrow \mathbb{R}$ que toma como argumento una partícula candidata, representada con un vector de números reales, y como salida un número real que proporciona el valor de la función objetivo. El objeto es hallar una solución α que verifique $f(\alpha) \leq f(\beta)$ para todo β correspondiente al espacio de búsqueda; es decir, α sería el mínimo global. En problemas de maximización, se hace lo mismo pero con la función $h=-f$.

Designando S como el número de partículas en la nube, cada una de las cuales tiene una posición $x_i \in \mathbb{R}^n$ en el espacio de búsqueda designado y una velocidad $v_i \in \mathbb{R}^n$. Sea pos_i la mejor posición conocida de cada partícula i, y glo la mejor posición global conocida considerando el algoritmo completo. El algoritmo PSO puede describirse como sigue:

- Para cada partícula i=1…S
    - Se toma una posición inicial de la partícula aleatoriamente a través de un vector aleatorio uniformemente distribuido $x_i \approx U(l_{in}, l_{su})$, donde l_{in} y l_{su} son respectivamente el límite inferior y el límite superior del espacio de búsqueda.
    - Inicializar la mejor posición conocida de la partícula a su posición inicial $pos_i = x_i$.
    - Si $f(pos_i) < f(glo)$ entonces actualizar la mejor posición global conocida $glo = pos_i$.
    - Inicializar la velocidad de la partícula $vel_i \approx U(-|l_{su}-l_{in}|, |l_{su}-l_{in}|)$
- En tanto no se cumpla el criterio de parada, repetir
    - Para cada partícula i=1…S
        - Para cada dimensión d=1…n
            - Se eligen dos números aleatorios $r_1, r_2 \approx U(0,1)$
            - Se actualiza la velocidad de la partícula

$$vel_{i,d} = w \cdot vel_{i,d} + c_1 \cdot r_1(pos_{i,d} - x_{i,d}) + c_2 \cdot r_2(glo_d - x_{i,d}) \qquad (10.1)$$

c_1 y c_2 son los coeficientes de aceleración o aprendizaje (1 cognitivo y 2 social)

- Actualizar la posición de la partícula $x_i = x_i + vel_i$
- Si $f(x_i) < f(pos_i)$ entonces
    - Actualizar la posición de la partícula $pos_i = x_i$
    - Si $f(pos_i) < f(glo)$ se actualiza la posición global $glo = pos_i$

Posteriormente Shi y Eberhart (1999) introdujeron el factor de inercia w, cuyo fin es regular la cantidad de la velocidad anterior de la partícula que se

mantiene en el nuevo ciclo, controlar las capacidades de exploración y explotación del enjambre y converger de forma más precisa y eficiente. Realmente proporciona un recuerdo del pasado inmediato representando un freno al cambio drástico de dirección de las partículas. Si w≥1 las velocidades aumentan con el tiempo y las partículas apenas pueden cambiar su dirección para retroceder hacia el óptimo y el enjambre diverge. Si w<<1, el impulso pequeño solo se guarda desde el paso anterior y los cambios rápidos de dirección se establecen durante el proceso. Si w=0, la velocidad de las partículas desaparece y todas las partículas se mueven sin conocimiento de la velocidad previa.

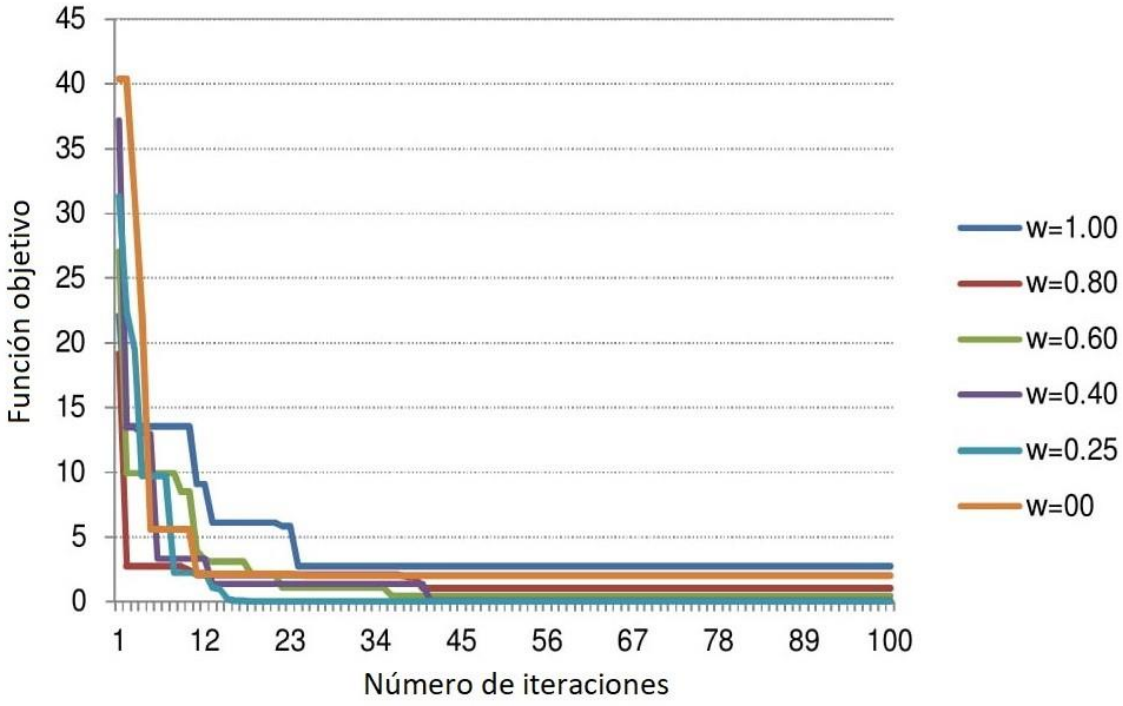

Figura 10.1. Efecto del factor de inercia en la función objetivo (Vishal A. Rane).

La elección de los parámetros más adecuados es un aspecto crucial para el buen cometido del algoritmo, habiendo sido objeto de una amplia investigación. La función objetivo da lugar a una hipersuperficie con irregularidades siendo necesarias pocas partículas e iteraciones en hipersuperficies poco irregulares y más partículas e iteraciones en hipersuperficies muy irregulares.

c_1, c_2 y w son coeficientes definidos por especialistas para regular el comportamiento y la eficacia del método PSO. Del término de inercia ha se hablado anteriormente, el factor cognitivo mide el rendimiento de las partículas en relación con los movimientos pasados a modo de recuerdo individual de la posición que fue la mejor que alcanzó la partícula. El efecto del componente cognitivo (a veces llamado nostalgia) constituye la tendencia de los individuos a regresar a las posiciones que mejor rendimiento tuvieron tiempo atrás. Por otra parte, el término social mide el rendimiento de las partículas en relación con el grupo de partículas vecinas. Resumiendo, c_1 representa la confianza de una partícula en sí misma mientras que c_2 expresa cuanta confianza tiene una partícula en sus vecinos.

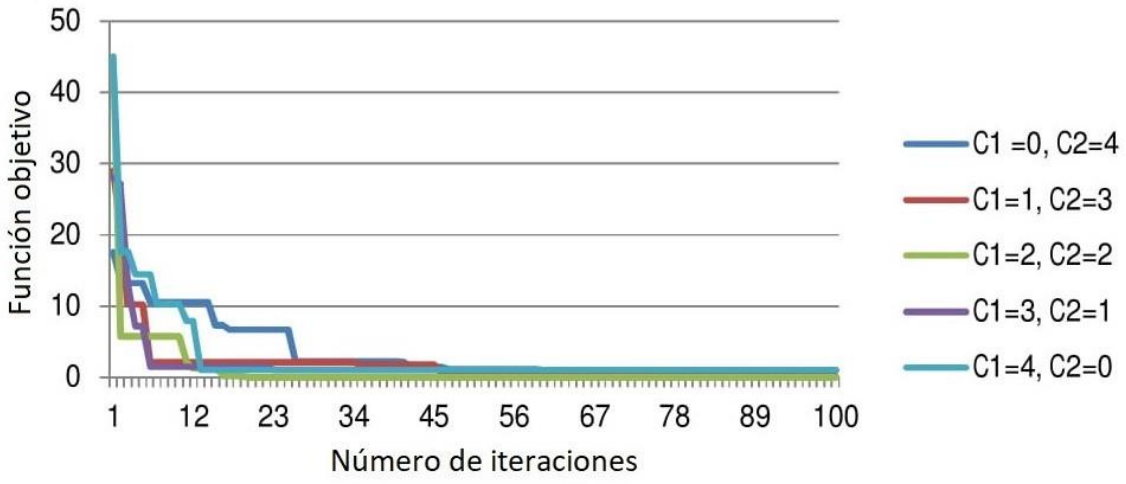

Figura 10.2. Efecto de los coeficientes de aceleración en la función objetivo (Vishal A. Rane).

Los coeficientes de aceleración junto con los valores aleatorios r_1 y r_2 mantienen la influencia estocástica de los componentes cognitivo y social de la velocidad de la partícula. Cuando $c_1=c_2=0$, todas las partículas continúan volando a su velocidad actual hasta que alcanzan el límite del espacio de búsqueda. Si $c_1>0$ y $c_2=0$, las partículas son independientes. Por el contrario, cuando $c_2>0$ y $c_1=0$, todas las partículas se focalizan en el punto glo. Otro tanto ocurre si $c_1=c_2$. En el caso en que $c_1>>c_2$, cada partícula está más fuertemente influenciada por su mejor posición personal, lo que se traduce en un merodeo excesivo. Cuando $c_2>>c_1$, todas las partículas están fuertemente influenciadas

por la mejor posición global lo que hace que todas las partículas se focalicen prematuramente en un óptimo local.

c_1 y c_2 toman valores entre 0 y 4 con aproximaciones matemáticas como 0.5+log(2) mientras que w toma valores entre 0,25 y 1,2 o bien 1/(2*log(2)) (paquete hydroPSO de r-project). En este trabajo se han encontrado buenos resultados empleando c_1=2 c_2=2 y w=1 que va en concordancia con las investigaciones empíricas efectuadas.

Para evitar la convergencia prematura se ignora la mejor posición global sustituyéndola por la mejor posición de un sub-enjambre el cual puede obtenerse reduciendo el área a la zona circundante a la partícula en movimiento. PSO modela la cooperación entre individuos mediante la definición de un entorno o neighborhood (vecindario) para cada partícula, de modo que en cada iteración se selecciona un candidato del vecindario como guía en la búsqueda sucesiva. Las topologías del vecindario más interconectadas distribuyen el detalle relativo a la información de la mejor solución del enjambre con mayor rapidez produciendo consecuentemente una convergencia más rápida. Sin embargo, si esta velocidad es demasiado rápida aparece con más frecuencia el problema de la convergencia prematura al no explorar completamente el espacio de búsqueda pudiendo llegar a producirse la convergencia a un óptimo local en lugar de a uno global.

El algoritmo muestra un comportamiento típico metaheurístico alternando entre un comportamiento de exploración (búsqueda en una amplia zona) y de explotación (búsqueda local), siendo esencial la selección de parámetros para así alcanzar un equilibrio idóneo entre exploración y explotación, evitando una convergencia prematura al no explorar completamente el espacio de búsqueda pudiendo alcanzar un óptimo local en lugar de a uno global.

No obstante, la compresión detallada de como el enjambre es capaz de mejorar el proceso de optimización aún no se ha comprendido exactamente, especialmente en problemas multidimensionales, discontinuos o variables fuertemente con el tiempo.

Otro tanto ocurre con la convergencia que puede utilizarse con dos acepciones. Como la mejor posición global glo conocida que tiende a aproximarse al óptimo del problema, o bien convergencia puede referirse a una concentración donde todas las partículas se aglutinan en un punto del espacio de búsqueda.

Diversos autores han realizado intentos de demostrar la convergencia si bien estos análisis adolecen de ser demasiados simplistas al considerar enjambres de una única partícula o se permiten un número infinito de iteraciones lo cual no resulta en modo alguno realista. Por lo cual, el análisis de la convergencia de los algoritmos PSO y la elección de los parámetros regulatorios depende de los resultados empíricos.

10.2. Población y número de iteraciones

Una pregunta que surge en este tipo de algoritmos es cuál es el tamaño idóneo de la población o más apropiadamente cual es el tamaño idóneo de partículas en el enjambre. Las pequeñas poblaciones corren un riesgo importante al no alcanzar totalmente el espacio de búsqueda mientras que centrarse en enjambres de gran tamaño supone un coste computacional excesivo al tener iteraciones que abarcan demasiado espacio de búsqueda.

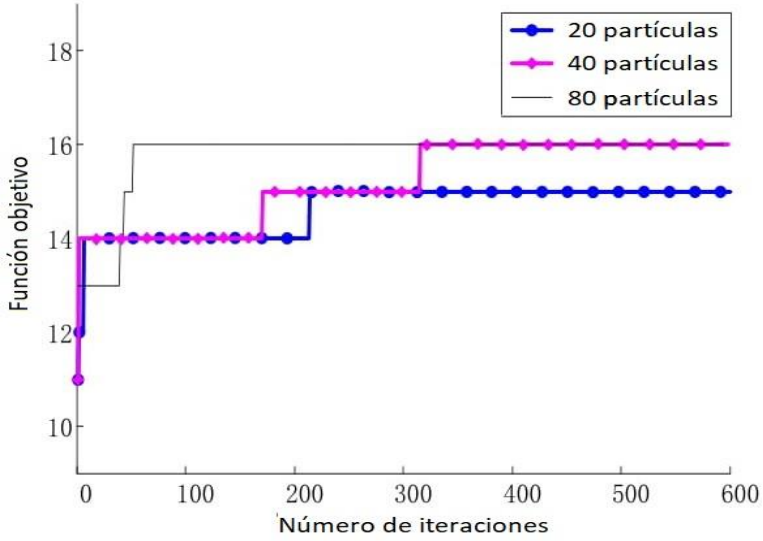

Figura 10.3. Relación entre número de iteraciones y número de partículas (Junbin, Jinyan, Yafeng y Tianzhen).

Obviamente un gran número de partículas reducen el número de iteraciones necesarias para obtener un buen resultado aunque, por el contrario, incrementan el costo computacional ya seriamente comprometido en este tipo de problemas lo que finalmente se traduce en tiempo de cálculo. Para concluir, los estudios empíricos efectuados revelan que un tamaño de enjambre en algoritmos PSO debe estar comprendido entre 20 y 60 (no obstante, aunque la mayoría de los autores coinciden en el límite inferior, el superior varía entre 50 y 80 dependiendo del investigador).

Por otra parte, la cuestión del número de iteraciones que permitan obtener un buen resultado es una cuestión crucial. Un número demasiado bajo de iteraciones puede truncar la búsqueda de una forma prematura mientras que un número muy grande de iteraciones complica en demasía el proceso computacional con operaciones innecesarias y el correspondiente tiempo de cálculo asociado. Atendiendo a la necesidad de obtener resultados rápidos, con precisión suficiente y la capacidad computacional disponible, en este trabajo se han encontrado buenos resultados empleando un número de partículas de 20 y un número de iteraciones de 200, siempre que no se alcanza la condición de parada antes.

Capítulo 11. Evolutionary PSO

Actualmente existe una cierta competición dentro de las técnicas metaheurísticas por saber cuál es la mejor o cual tienes mayores ventajas y menores inconvenientes. La técnica PSO es un método de búsqueda metaheurística muy reciente considerado como uno de los métodos más poderosos para resolver problemas de optimización global cuya mecánica se inspira en los enjambres o comportamiento colaborativo de poblaciones biológicas.

PSO presenta indudables ventajas como facilidad de implementación, simplicidad, presenta elementos inteligentes, robustez, requiere de pocos ajustes, poco sensible a la naturaleza de la función objetivo, número limitado de parámetros, el impacto de los parámetros en las soluciones se considera que es menos sensible en comparación con otros algoritmos heurísticos, poco dependiente de las condiciones iniciales en comparación con otros métodos evolutivos, lo cual implica con todo esto que el algoritmo de convergencia es bastante robusto. Estas técnicas PSO pueden obtener soluciones de alta calidad con tiempos de más cortos y características estables de convergencia en comparación con otros métodos estocásticos.

Indudablemente y aunque todo parezcan ventajas, también existen desventajas. Una importante desventaja es que no está soportado por una base matemática sólida aún en desarrollo (este inconveniente no es exclusivo del PSO sino que también aparecen en otras técnicas de optimización heurística) pero quizás la desventaja más significativa es la facilidad que tiene

este algoritmo de caer en un óptimo local (convergencia prematura) en el espacio de alta dimensión con una baja tasa de convergencia en el proceso iterativo. Este problema no está en absoluto aclarado aunque se piensa que la velocidad de las partículas disminuye en el espacio de búsqueda lo cual conduce a un efecto “implosivo” con un estancamiento del enjambre.

Para tratar este problema, se propone una solución basada en emplear un algoritmo de optimización de enjambre de partículas adaptativo basado en la combinación con las técnicas evolutivas del algoritmo genético GA (Genetic Algorithm) lo cual permite alcanzar una mayor capacidad de convergencia global y una mayor eficiencia en la búsqueda. Este método ha sido desarrollado para este trabajo prácticamente desde cero mejorando la diversidad del algoritmo y evitando la caída de partículas en el óptimo local (PSO) a la par que se mejora un problema inherente a los algoritmos GA como es su poca velocidad de convergencia. Los resultados alcanzados en la simulación desarrollada para este trabajo indican que el algoritmo propuesto, combinación de PSO y GA, mejora ostensiblemente el problema de convergencia prematura.

Los algoritmos genéticos y la técnica de optimización PSO han atraído una atención considerable entre diversas técnicas modernas de optimización heurística. El GA ha gozado de una gran popularidad en el mundo académico y en la industria principalmente por su facilidad de implementación, intuitividad, y capacidad de resolver de manera efectiva problemas de optimización no lineales.

La adición de técnicas evolutivas al algoritmo PSO mejora los resultados claramente al emplear elementos de optimización avanzados basadas en inteligencia computacional y poder superar las limitaciones computacionales de las técnicas numéricas existentes como el alto tiempo de cálculo y la incapacidad de converger en el punto óptimo al sufrir el problema de convergencia prematura. En relación con las técnicas evolutivas, estas se han incorporado en este trabajo de forma más exhaustiva como elemento aparte en forma de Apéndice.

Esta técnica aquí desarrollada se ve útil en problemas con un paisaje adaptativo extremadamente complejo, con funciones objetivo fuertemente discontinuas, estridentes y variables con el tiempo, y con muchos óptimos locales. La mayor parte de los problemas reales tienen un espacio de soluciones gigante imposible de explorar en detalle. En este caso, es esencial disponer de tecnologías capaces de obviar los óptimos locales y no quedar atrapados en ellos, para lanzarse a la búsqueda del óptimo global.

11.1. Implementación del algoritmo EPSO

Lo primero que necesita el algoritmo PSO es una nube de partículas o lo que es lo mismo, de soluciones candidatas. En este trabajo se han conseguido buenos resultados eligiendo una nube de partículas de 20 partículas e incrementando el número de iteraciones hasta 200 (en el caso de no cumplirse antes el criterio de parada) lo cual permite obtener buenos resultados con precisión suficiente de acuerdo a la capacidad computacional disponible.

Esta nube de partículas se mueve por el espacio de búsqueda de acuerdo a unas reglas matemáticas. El movimiento de cada partícula en cuestión es función de su mejor posición alcanzada por un lado, y por otra parte de la mejor posición global hallada por el algoritmo en todo el espacio de búsqueda. Este algoritmo va descubriendo nuevas y mejores posiciones, las cuales reorientan los movimientos futuros de las partículas. Este proceso se va repitiendo hasta encontrar una solución satisfactoria.

Así, estos algoritmos son métodos de optimización que tratan de hallar $(x_1,...,x_n)$ tales que $E(x_1,...,x_n)$ sea máximo. Para ello, es necesario parametrizar el problema en una serie de variables, $(x_1,...,x_n)$. Inicialmente la nube de partículas se selecciona aleatoriamente cumpliendo las condiciones del problema. En el problema tratado en este trabajo, las variables de las que depende la función a optimizar son los factores de decisión factMmmm, factdecv y factdeca, según se ha desarrollado anteriormente, los cuales tienen que cumplir la condición factMmmm≥factdecv≥factdeca, siendo el objetivo del

algoritmo de optimización encontrar estos factores de tal modo que la entropía del enjambre sea máxima.

Al aplicar el algoritmo PSO es necesario definir una serie de parámetros inherentes al propio algoritmo. Concretamente, en este trabajo se han alcanzado buenos resultados empleando, respectivamente, los coeficientes de aceleración cognitivo y social c_1=2 y c_2=2 y el factor de inercia inicial w=1 de acuerdo con las investigaciones empíricas efectuadas.

Por otra parte para cumplir la condición factMmmm≥factdecv≥factdeca, el factor de inercia puede disminuirse automáticamente en concordancia con la función $v\square l_{i,d} = w \cdot vel_{i,d} + c_1 \cdot r_1(pos_{i,d} - x_{i,d}) + c_2 \cdot r_2(glo_d - x_{i,d})$.

En este capítulo, las simulaciones se han implementado empleando un enjambre de RPAs bien conocido, según los resultados obtenidos en capítulos previos:

- tamaño promedio de los RPAs=1 metro
- aceleración=5 m/s^2
- deceleración=1.5 m/s^2
- tiempo de la trayectoria completa=60 segundos
- paso de tiempo=0.2 segundos
- velocidad de crucero=17 m/s
- velocidad de pérdida=6 m/s
- velocidad angular de viraje=18 grados sexagesimales/segundo

Indudablemente, ya se ha comentado el problema relativo a la convergencia prematura que sufre especialmente el algoritmo PSO, aunque este método de búsqueda metaheurística es considerado uno de los métodos más poderosos para resolver problemas de optimización global, a pesar de su relativa modernidad.

Para paliar el problema de convergencia prematura se ha propuesto una solución basada en emplear un algoritmo de optimización de enjambre de partículas adaptativo basado en la combinación con las técnicas evolutivas del

algoritmo genético lo cual permite alcanzar una mayor capacidad de convergencia global y una mayor eficiencia en la búsqueda. Este método se ha desarrollado para este trabajo prácticamente desde cero mejorando la variedad del algoritmo y evitando la caída de partículas en el óptimo local (PSO) a la par que se mejora un problema asociado a los algoritmos GA como es su poca velocidad de convergencia.

Por otra parte, el objetivo global de este trabajo es que la actuación del enjambre adquiera un comportamiento colectivo, sincronizado, organizado y sinérgico, fruto de la cooperación y colaboración de los miembros del enjambre, con rasgos de inteligencia artificial. Para ello, se ha adoptado una solución basada en un conjunto de algoritmos que comprenden un EPSO de optimización evolutiva y búsqueda de estados, combinado con una SVM de aprendizaje automático, que en su conjunto configura una máquina de inteligencia artificial con capacidad de seleccionar la mejor decisión entre varias posibilidades en un enjambre de RPAs.

Así, el enjambre de RPAs se ha ideado como un grupo de vehículos con un tamaño más bien reducido por varias razones, como su furtividad, maniobrabilidad, baja firma radar, además de su bajo precio frente a los RPAs más grandes. Bien desplegables desde aviones de combate o bien desde sistemas terrestres, este pequeño tamaño no permite albergar grandes microprocesadores, lo que conduce a una baja capacidad de computación y por tanto a una lentitud en los cálculos. Dado que en las misiones operativas resulta cardinal la presteza en la toma de decisiones, en este trabajo se plantea un procedimiento basado en efectuar un conjunto de simulaciones, anteriormente al lanzamiento del enjambre, y aplicar un algoritmo de aprendizaje automático SVM (Support Vector Machine) donde partiendo de estas simulaciones como datos de aprendizaje entrenar al SVM y construir un modelo que prediga el resultado y proporcione una respuesta para una nueva muestra, de tal modo que el enjambre adquiera cierta capacidad para tomar decisiones.

Bien, para esto se necesita un conjunto de datos de aprendizaje para entrenar al SVM, los cuales provienen del algoritmo EPSO. En el caso de que el EPSO funcione muy bien convergerá al óptimo global y la nube de puntos de aprendizaje será muy pequeña. La adición de técnicas evolutivas al algoritmo PSO tenía la intención de paliar el problema de la convergencia prematura pero también podría contemplarse como una forma de controlar y graduar la obtención de óptimos locales cercanos al óptimo global de manera que la nube de puntos de aprendizaje tenga una dimensión aceptable proporcionando variedad a la par que precisión y eficiencia.

Dado que la función de optimización en cuestión, presenta un perfil complejo y adaptativo, con ruido, discontinua y cambiante con el tiempo, se han alcanzado buenos resultados, en este trabajo, reduciendo el porcentaje de mutación al 5% de las ocasiones y el cruce solo aparecerá una vez por generación.

Los códigos de cálculo se han realizado haciendo uso del entorno r-project, los cuales se encuentran al final del documento en los Apéndices.

11.2. Condición de parada

Generalmente, la condición de terminación debería ser la convergencia del algoritmo hacia una solución estable o alcanzar un número prefijado de iteraciones. Como el PSO trabaja con una nube de partícula, teóricamente se espera que el procedimiento converja de manera que al final del proceso la nube presenta una estructura similar y en el infinito esta nube sea una clonación de la misma partícula.

El PSO es un algoritmo muy reciente e innovador donde, y aunque ha habido diversos intentos por demostrar la convergencia, o no se ha conseguido, o dichos análisis presentan hipótesis muy simplificatorias al considerar una única partícula o presentar un número infinito de iteraciones que los hacen irreales. Así, estos algoritmos están fundamentados en la experiencia empírica al elegir los parámetros regulatorios encontrándose el investigador que estos algoritmos hay veces que son extraordinariamente potentes y otras veces no.

Se van a establecer tres condiciones de parada que detienen el algoritmo si se cumple una cualquiera de ellas.

- Condición 1

  Si la solución no ha mejorado, tras K_1 series consecutivas de L pasos, una cierta cantidad definida por $\epsilon_1>0$. Es decir, $E(K_1 \cdot L+n)> E(n)(1+\epsilon_1) \Longrightarrow$FIN

- Condición 2

  La convergencia de la nube se alcanza en la medida que al final del proceso presente cierta similitud, y en el infinito se reduzca a una única partícula. Con esta condición, es necesario comprobar el grado de semejanza de la población mediante un análisis estadístico.

  Supóngase una nube concreta compuesta por N partículas y supóngase que esta nube concreta es una muestra extraída aleatoriamente de una población madre mayor. Con esta hipótesis de partida, es posible aplicar la teoría de la estimación. Considerando el estadístico media y el estadístico varianza:

$$a = \frac{x_1 + \cdots + x_n}{N}$$
$$s^2 = \frac{(x_1 - a)^2 + \cdots + (x_n - a)^2}{N - 1} \qquad (11.1)$$

  Se considera que se ha alcanzado la condición de parada cuando un alto porcentaje de la nube se encuentra dentro del intervalo de confianza elegido. Se ha tomado un intervalo de confianza del 90%, aunque el análisis puede efectuarse para cualquier otro intervalo de confianza. De esta forma, se considera que la nube converge cuando el 90% de las partículas se encuentran dentro del intervalo de confianza de la distribución estadística. Asumiendo que la población madre presenta una distribución normal, se puede estimar la media para una gran muestra (N>30) como:

$$\alpha = a \pm w_{90}[N(0,1)]\frac{s}{\sqrt{N}} \qquad (11.2)$$

Siendo $w_{90}[N(0,1)]$ el valor que define el intervalo de confianza de la distribución normal N(0,1), de tal modo que el 90% de las partículas se encuentren dentro del intervalo de confianza de la campana de Gauss.

Para una pequeña muestra (N<30):

$$\alpha = a \pm w_{90}[t_{N-1}]\frac{s}{\sqrt{N}} \tag{11.3}$$

Siendo $w_{90}(t_{N-1})$ el valor que define el intervalo de confianza de la distribución t de Student con N-1 grados de libertad, de tal forma que el 90% de las partículas se encuentren dentro del intervalo de confianza de la distribución.

Téngase en cuenta que, en este criterio de parada, más que un estimador, lo que se busca es una medida del agrupamiento a fin de proporcionar un criterio de parada del algoritmo que sea coherente. Esta condición se comprobará tras K_2 series consecutivas de L pasos.

- Condición 3

  Si el número de iteraciones alcanza un valor superior al valor máximo establecido.

11.3. Resultados obtenidos

Continuando con lo indicado anteriormente, se ha ejecutado el algoritmo EPSO en 100 ocasiones obteniendo una nube de puntos que se emplearan como datos de aprendizaje en el algoritmo SVM. En primer lugar, la cantidad 100 podría ser cualquier otra y dependerá del tiempo de que se disponga en la misión para generar simulaciones de aprendizaje. Obviamente, cuantas más simulaciones se tengan, mejores resultados se alcanzaran pero todo ello va a estar supeditado al tiempo que se disponga para ejecutar simulaciones. En segundo lugar, esta nube de puntos constituye un conjunto de óptimos locales que se encuentran cerca del óptimo global, si bien esto no es en sí importante, dado que lo que se pretende es que la entropía alcance valores elevados a fin de que el efecto degradativo, saturador y neutralizante en el sistema defensivo enemigo potencie la confusión, decepción, el desconcierto y el colapso.

Anteriormente ya se apuntó que es necesario aumentar el peso de la entropía, lo cual se ha conseguido definiéndose un nuevo concepto de entropía (que se ha denominado entropía global), la cual se obtiene como un sumatorio de EEA para el tiempo de vuelo completo (300 ciclos para 60 segundos) donde pequeñas diferencias en cada ciclo, al final del tiempo de la trayectoria completa, proporcionan grandes diferencias en el proceso de optimización.

Ordenando las simulaciones obtenidas en función de la entropía global, de forma creciente, para facilitar la visualización:

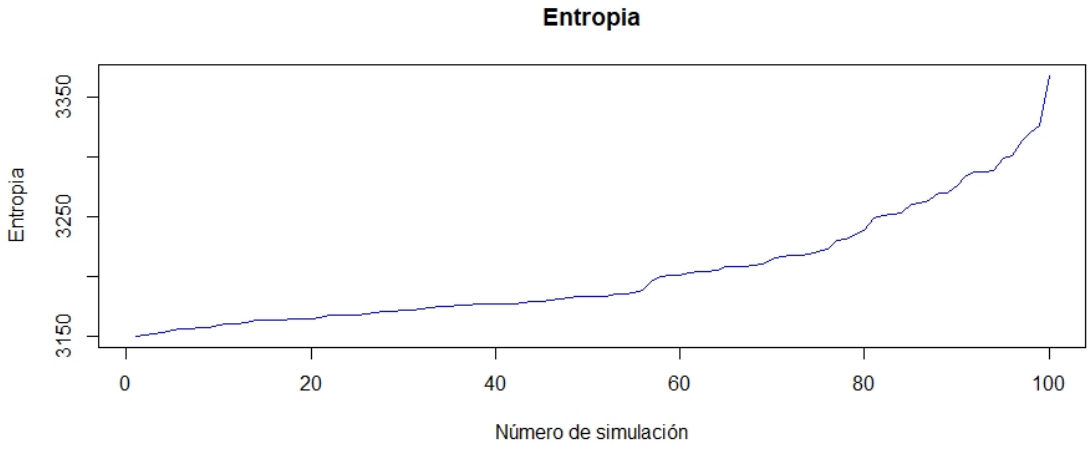

Figura 11.1. Entropía global.

Análogamente, se tienen los factores de decisión factMmmm, factdecv y factdeca:

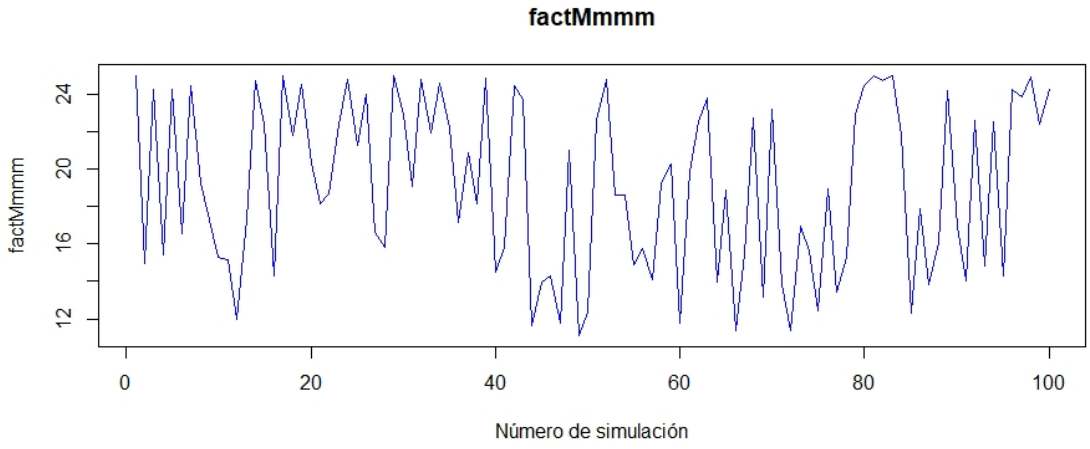

Figura 11.2. factMmmm.

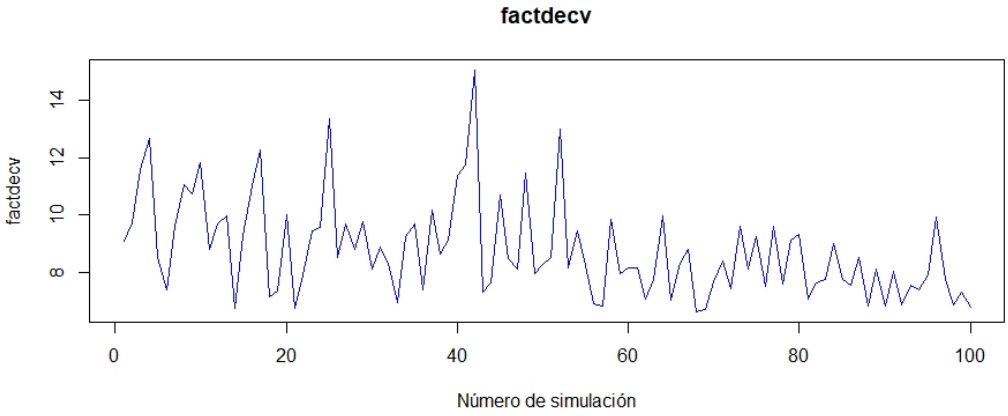

Figura 11.3. factdecv.

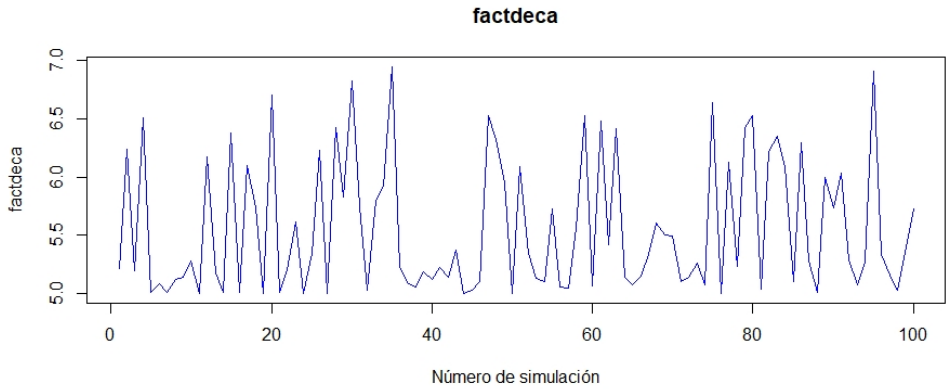

Figura 11.4. factdeca.

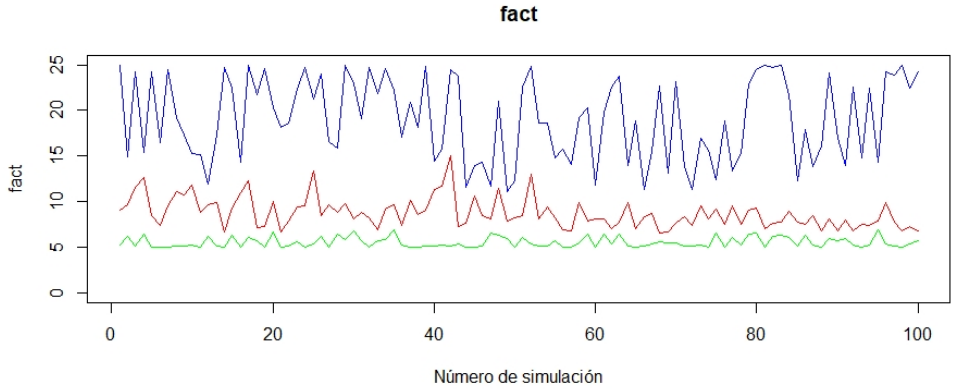

Figura 11.5. Factores de decisión factMmmm, factdecv y factdeca.

Quizás más interesante representar los factores de decisión factMmmm, factdecv y factdeca en función de la entropía:

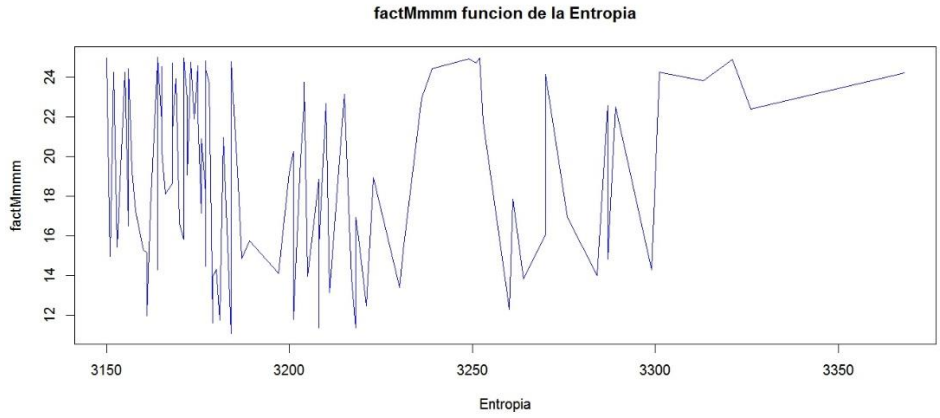

Figura 11.6. factMmmm función de la entropía.

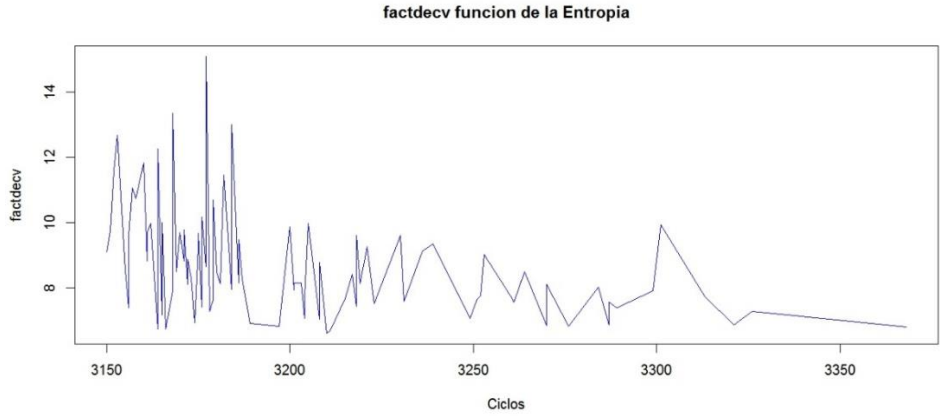

Figura 11.7. factdecv función de la entropía.

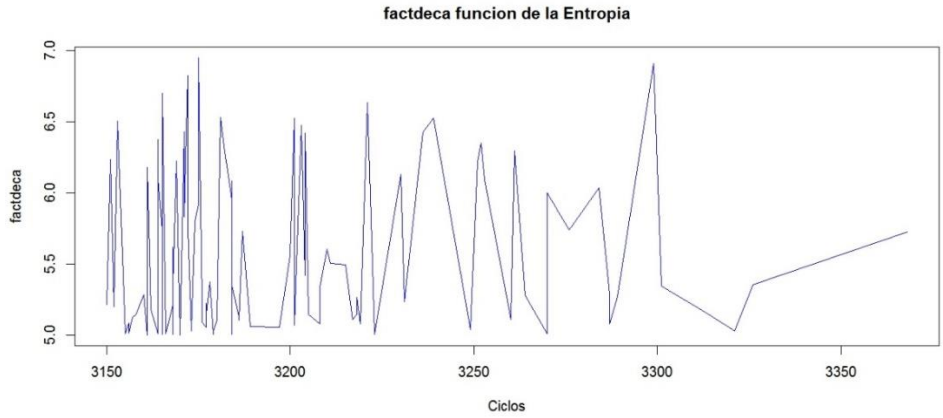

Figura 11.8. factdeca función de la entropía.

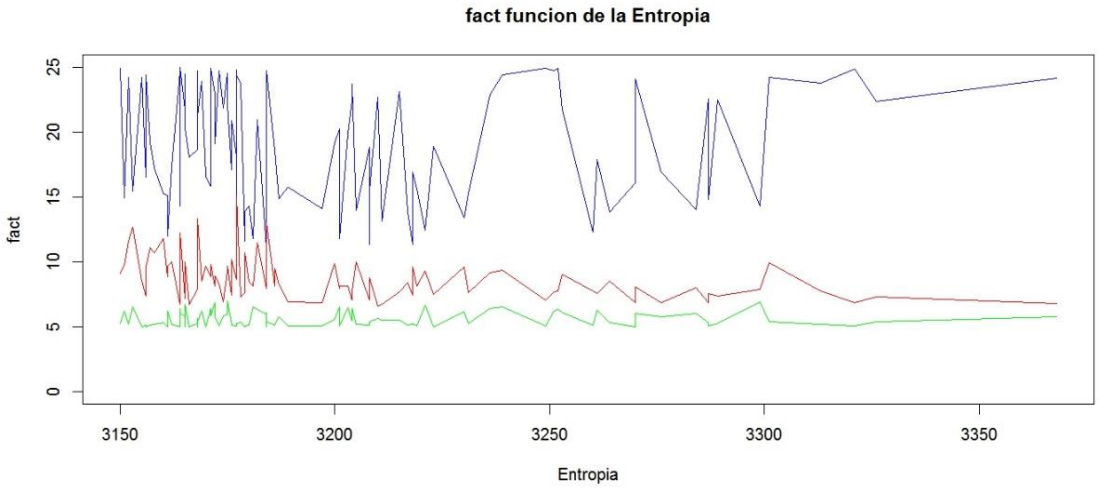

Figura 11.9. Factores de decisión factMmmm, factdecv y factdeca función de la entropía.

Capítulo 12. Inteligencia Artificial. Aplicación a enjambres

12.1. Inteligencia artificial

El primer investigador que menciono el concepto de Inteligencia Artificial (AI, Artificial Intelligence) fue John McCarthy en 1956 como "...la ciencia e ingenio de hacer máquinas inteligentes, especialmente programas de cómputo inteligentes". Unos cuantos años antes, otro genio, Alan Turing, se planteaba si las máquinas podrían pensar. Turing atisbo enseguida la necesidad de hallar una evidencia empírica, por lo que propuso el denominado Test de Turing para determinar si una máquina es capaz de pensar. Este test es una prueba de habilidad donde se determina si una máquina muestra un comportamiento inteligente similar al de un ser humano.

Básicamente, podría considerarse una inteligencia artificial como un conjunto de algoritmos estructurados en programas informáticos, que corren sobre un hardware, y que intentan imitar el modo de funcionamiento del cerebro humano. Así, una inteligencia artificial que intentará imitar a una mente humana, se percataría de lo que ocurre a su alrededor, procesa esa información, e infiere conclusiones que no le han sido programadas.

No obstante, los científicos aún no han alcanzado la meta de conseguir crear una inteligencia artificial del tipo generalista, aunque se han conseguido importantes avances en inteligencias artificiales especializadas. Ello es debido a diferentes razones, aunque es de significar que los computadores actuales no son lo suficientemente potentes para el volumen de cálculo requerido que la

inteligencia artificial general necesitaría, si bien se tiene esperanzas en que este obstáculo pueda ser salvado mediante la aplicación de la computación cuántica.

Quizás el ejemplo más representativo de la inteligencia artificial sea el de los programas de ajedrez; donde una máquina se enfrenta a un oponente humano. Claude Shannon en 1950 predijo las dos posibles principales formas de abordar el problema, a las que nombró de “Tipo A”, y de “Tipo B”. Los programas “Tipo A” basan su funcionamiento en la "fuerza bruta", examinando todas y cada una de las posibles alternativas usando técnicas especializadas. Por el contrario, los de “Tipo B” harían uso de una especie de "inteligencia artificial especializada" analizando únicamente las mejores alternativas, de forma similar a lo que hacen los jugadores humanos.

A primera vista, podría pensarse que la solución son los de “Tipo B”, pero estos programas presentan el inconveniente de tener que decidir qué movimientos son lo suficientemente buenos para ser tenidos en cuenta. De hecho, la Universidad de Northwestern, ganadora de varios torneos ACM (Computer Chess Championships) abandonó los programas de “Tipo B” en 1973 pasándose a los de “Tipo A” volviendo a ganar el ACM cinco años seguidos.

El principal inconveniente de los programas de “Tipo B” es que imitan el pensamiento humano y por tanto eran predecibles por los grandes maestros; sin embargo, los de “Tipo A” no intentan imitar el pensamiento humano empleando en su lugar técnicas de búsqueda complejas, las cuales emplean soluciones inteligentes de profundidad de búsqueda y poda. De este modo, los programas de ajedrez buscan obtener lo mejor de los dos mundos dejando que las computadoras actúen donde son punteras, calculando a gran velocidad, en vez de intentar emular la inteligencia humana. Pero a finales de los 90, el efecto se invierte y los programas de “Tipo A” vuelven a entrar en liza con fuerza. Realmente, estos programas son una combinación de ambos tipos donde “la inteligencia de las máquinas” se combina con una gran velocidad de cálculo.

Otro ejemplo a tener en cuenta son los sistemas expertos, los cuales presentan un conocimiento especializado sobre sobre un tema en concreto, como la bolsa

por ejemplo. Así, los sistemas expertos analizan la información existente y mediante algoritmos complejos intentan predecir el comportamiento futuro de los títulos, vendiendo y comprando en el momento adecuado para obtener beneficios.

Otro científico, Nils John Nilsson considera cuatro los pilares en los que se basa la inteligencia artificial:

- Búsqueda del estado requerido en el conjunto de los estados analizados con la meta de encontrar un estado final con las características deseadas.
- Algoritmos evolutivos, métodos de optimización y búsqueda de soluciones basados en los postulados de la evolución biológica y su base genético-molecular (mutaciones, recombinaciones genéticas y selección de individuos).
- Aprendizaje automático (Machine Learning), cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender (redes neuronales artificiales, máquinas de vectores soporte, redes bayesianas, árboles de decisión, deep learning,…).
- Razonamiento mediante una lógica formal, donde partiendo de uno o más juicios, se deriva otro juicio distinto.

Tradicionalmente un enfoque que se le ha dado a la inteligencia artificial son las redes neuronales artificiales, en el cual máquinas y algoritmos intentan simular el comportamiento de las neuronas del cerebro humano; no obstante, los científicos se preguntan si este es el enfoque apropiado. Si las máquinas son diferentes de las personas, porque las obligamos a pensar como los humanos? No deberíamos desarrollar nuevas estrategias de aprendizaje automático generando comportamientos a partir de una información suministrada externamente y permitir que los computadores trabajen haciendo lo que mejor saben hacer, calcular de forma muy rápida y con gran precisión.

De cualquier forma, actualmente la inteligencia artificial nos rodea por doquier aunque únicamente en tareas muy específicas, como por ejemplo el programa de área táctil en el cual escribimos a mano alzada en nuestros teléfonos

móviles y el propio aparato es capaz de identificar todos los trazos de diferentes personas. También los sistemas de atención telefónica automáticos que nos solicitan respuestas para derivar nuestras llamadas a un operario dependiendo de nuestro problema. Sin olvidar el reconocimiento de patrones para identificación de usuarios atendiendo a diversas características como estructura facial, pulsaciones en teclado, imagen retiniana, etc.

12.2. Inteligencia de enjambres

Inteligencia de enjambre es una parte de la Inteligencia artificial que estudia y analiza el comportamiento colectivo de sistemas autoorganizados, autónomos, descentralizados, ya sean artificiales o naturales. Este concepto apareció por primera vez en 1989 en los trabajos de Gerardo Beni y Wang Jing. Un grupo de agentes no inteligentes (sensores, robots, RPAs,…) interactúa entre ellos y con el entorno para producir un comportamiento que se ha venido en denominar colectivamente inteligente. Inspirados por sistemas biológicos de los que se encuentran en la naturaleza, los sistemas de inteligencia de enjambre están normalmente formados por un conjunto de agentes primarios elementales que interaccionan con el medio y con ellos mismos.

Una colonia de hormigas o abejas es el ejemplo más típico que se puede encontrar en la naturaleza. Los agentes individuales (hormigas o abejas) muestran de forma individual muy poca inteligencia equivocándose repetidamente, como cuando tienen que encontrar un camino hasta los alimentos. No obstante, estos individuos se comunican entre ellos de alguna forma (feromona en las hormigas, danza en las abejas) de tal modo que el enjambre es capaz de encontrar la mejor solución al problema en cuestión; es decir, el camino más corto hacia la fuente de alimento. De esta forma, la inteligencia del enjambre surge de múltiples interacciones entre muchos agentes y permite discriminar la solución óptima entre muchas soluciones menos eficientes.

La inteligencia de enjambre se caracteriza porque los elementos individuales son autónomos y están distribuidos, y en principio no hay una autoridad central y orientadora que determine las acciones de cada agente (si bien si puede

haber una autoridad que proporcione órdenes al enjambre como el caso del piloto desde la GCS). Cada agente va equipado con actuadores o sensores para percibir el entorno y comunicarse siendo la toma de decisiones del grupo distribuida a través de muchos dispositivos y plataformas.

Capítulo 13. Support Vector Machine

El aprendizaje automático es una disciplina que tiene como objetivo que una máquina sea capaz de utilizar experiencias previas en forma de conjunto de datos para plantear un esquema de resolución de un problema. A priori, un ordenador es la primera máquina que viene a la mente que puede realizar de forma automática este tipo de tareas de aprendizaje. Ello requiere estructurar de donde aprender, qué hacer con esos datos de aprendizaje y que se pretende obtener.

Actualmente, se dispone de la tecnología y capacidad para entrenar a un ordenador para realizar diversos tipos de tareas de diversos grados de complejidad. Como una máquina puede aprender es lo que ha venido en denominarse aprendizaje automático ("Machine Learning"), que se constituye como una parte de las ciencias de la computación y de la inteligencia artificial, cuyo fin primordial es desarrollar técnicas y estructuras que permitan a las computadoras aprender de una forma similar a como se entiende lo hacen los humanos. De forma más específica, los programas desarrollados bajo este concepto deben ser capaces de sistematizar comportamientos a partir de un conjunto de datos e información suministrada a partir de ejemplos de partida análogamente a un proceso de inducción del conocimiento.

Este concepto de aprendizaje automático o más popularmente conocido como "Machine Learning" se desarrolla a partir de los años cincuenta como una forma compleja de tratar datos. En 1956, Arthur Samuel, que trabajaba en IBM como ingeniero entrena un ordenador IBM 701 en el juego de las damas siendo

el primer ejemplo más conocido de aprendizaje automático. No obstante, el aprendizaje automático se fundamente en otras disciplinas como la “ciencia de datos" que puede sumarizarse como una serie de patrones, estadísticas, análisis, sistematizaciones, cálculos y algoritmos, que tienen como objetivo extraer información y conclusiones de conjuntos numéricos y bases de datos.

Descendiendo al mundo de las definiciones, Arthur Samuel define el aprendizaje automático como "un área de estudio que permite a las computadoras aprender sin estar programadas manifiestamente para ello". Otra forma de abordar este concepto es también conocida coloquialmente como "máquinas de entrenamiento", una parte de la inteligencia artificial que estructura algoritmos a partir de bases y series de datos, confecciona análisis predictivos y correlaciona hechos dispares sin relación aparente a priori.

En una primera etapa de aprendizaje se analizan los datos de partida para en una segunda fase de "verificación" analizar una segunda tanda de datos. Básicamente, una primera clasificación podría definirse como “aprendizaje supervisado” y “aprendizaje no supervisado”. El primero construye un modelo de correlación entre las variables de entrada y salida del sistema mientras que el segundo únicamente considera las entradas al sistema extrayendo “correlaciones fuertes” reconociendo patrones.

El aprendizaje automático se aplica en múltiples tareas como reconocimiento facial o software de detección de fisonomías, entrenamiento de ajedrez, traducción automática, reconocimiento de voz, detección y análisis de fraude bancario, páginas web, detección de spam, etc.

Existen múltiples algoritmos de aprendizaje automático pero en este trabajo nos vamos a centrar en las máquinas de vectores soporte (SVM, Support Vector Machine). Una máquina de vectores soporte es un procedimiento de aprendizaje automático que toma datos de entrada y los clasifica en dos categorías diferentes. Esta máquina toma los datos referenciados en un espacio multidimensional, y después, mediante técnicas de regresión busca un hiperplano o hipersuperficie n-dimensional que separa el espacio en dos semiespacios, separando de acuerdo a cierto criterio las dos clases de datos.

Una vez creada la máquina de vectores soporte, analiza los nuevos inputs con respecto a la hipersuperficie y los clasifica adecuadamente en una de las dos categorías.

Inicialmente, las SVMs fueron desarrolladas como una técnica de aprendizaje supervisado, idóneos para resolver problemas de clasificación, aunque a día de hoy se utilizan para resolver otros tipos de problemas como multiclasificación, agrupación, regresión, etc., habiéndose aplicado con éxito en reconocimiento de patrones, visión artificial, categorización de textos, procesamiento, análisis de series temporales,…

Si bien las máquinas de vectores soporte son estrategias muy eficientes, presentan tanto ventajas como desventajas. Entre las ventajas, se tiene:

- Muy eficaz cuando el número de dimensiones es superior al número de muestras.
- Muy eficaz en altas dimensiones.
- Emplea los vectores soporte (factores de entrenamiento de la función de decisión) optimizando su número.
- La función de decisión se sintetiza mediante diversos tipos de kernels lo que le da versatilidad para resolver problemas de diversa índole.
- Utiliza un subconjunto de puntos de entrenamiento en la función de decisión (llamados vectores soporte), por lo que también es eficiente desde el punto de vista de la memoria.

Los kernels son funciones matemáticas donde los datos iniciales se redimensionan buscando una separabilidad, transformando un problema de clasificación no lineal en el espacio dimensional de origen, en un problema mucho más sencillo, pero a costa de incrementar el número de dimensiones no linealmente relacionado con el espacio origen.

Otro método teórico que goza de gran profusión son las Redes Neuronales Artificiales (RNA), habiéndose constituido en una parte de la inteligencia artificial que goza de gran número de adeptos. Básicamente el principal motivo de interés es que permiten resolver problemas difíciles que de otro modo

resultaría mucho más complicado por otros métodos. No obstante, a pesar de presentar semejanzas con las SVMs, estos últimos se encuentran mejor asentados, desde un punto de vista teórico, y son más fáciles de extrapolar.

Las RNA intentan imitar a los cerebros biológicos con un modelo de cálculo basado esencialmente en neuronas simples artificiales; sin embargo, podría asegurarse que la mejor forma de “pensar” para una máquina sea imitar el pensamiento humano o de otro modo, podría encontrarse otra forma más eficiente de acometer la resolución de un problema. Ese es un dilema arduamente complejo de resolver.

Las máquinas SVMs surgen de la teoría de aprendizaje estadístico de Vapnik (1998), que se desarrolló para resolver problemas de clasificación binaria, aunque actualmente se emplean para resolver otros tipos de problemas (agrupamiento, regresión, multiclasificación). Más concretamente, dentro de los clasificadores, las SVMs pertenecen a la categoría de los clasificadores lineales mediante separadores lineales o hiperplanos, bien en el espacio original de las entradas o en un espacio transformado (espacio de características) en el caso de que las entradas no fueran linealmente separables en el espacio origen. Este espacio transformado está constituido por un número elevado de dimensiones y la localización del hiperplano de separación emplea unas funciones matemáticas que se han venido en denominar kernels.

Generalmente la mayor parte de los métodos de aprendizaje se concentran en minimizar los errores que genera el modelo formado a partir de los datos de entrenamiento (error empírico), si bien las SVMs minimizan el riesgo estructural, creando un hiperplano de separación que equidista de los datos más cercanos de cada tipo para conseguir maximizar el margen a cada lado del hiperplano.

Por otra parte, en la definición de este hiperplano, sólo se tienen en cuenta los datos de entrenamiento localizados justamente en la frontera de dichos márgenes, siendo estos datos en sí los llamados vectores soportes. Este hiperplano de margen máximo ha dado buenos resultados en la literatura,

soslayando en su mayor parte el problema de sobreajuste de los datos de entrenamiento.

Descendiendo a un nivel más teórico, la optimización del margen geométrico constituye un problema de optimización cuadrático con restricciones lineales que puede ser abordado con éxito a través de técnicas de programación cuadrática. La convexidad requerida para su resolución permite garantizar la existencia y unicidad de la solución, mientras que en una RNA no existe esa garantía de unicidad.

La aplicación de las SVMs a los problemas no lineales ha gozado de gran aceptación debido a que las SVMs están basadas en el principio de minimización del riesgo estructural (SRM, Structural Risk Minimization), base de la teoría de aprendizaje estadístico de Vapnik. Este principio se ha demostrado mejor que el principio de minimización del riesgo empírico (ERM, Empirical Risk Minimization) utilizado por las RNA. Las ventajas que tiene las SVMs se pueden reducir a:

- Minimización del riesgo estructurado.
- Pocos parámetros a ajustar.
- La estimación de los parámetros hace uso de la optimización de una función de costo convexa, evitando de este modo la existencia de un mínimo local.
- La solución final puede escribirse como un número pequeño de vectores soporte.

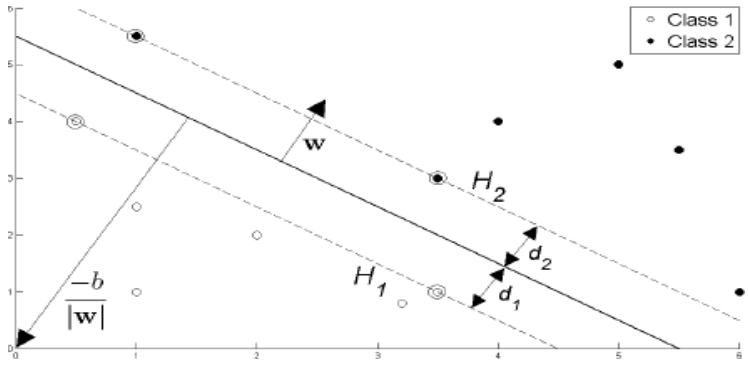

Figura 13.1. Hiperplano de separación en un espacio bidimensional con dos clases separables.

Resumiendo, las SVM constituyen un procedimiento de clasificación de una población en función de la partición en subespacios de múltiples variables definiendo y construyendo un hiperplano óptimo como superficie de decisión, de manera que el margen de separación entre las dos clases en los datos se maximiza. Los vectores soporte se alinean con las observaciones de entrenamiento como soporte para ubicar de forma óptima la superficie de decisión.

El entrenamiento de una SVM consiste en transformar los datos de entrada en un espacio de características de altas dimensiones, definiendo el kernel, para seguidamente resolver un problema de optimización cuadrática que permita ajustar el hiperplano óptimo y clasificar las transformadas características en dos tipos, siendo el número de características definido por el número de vectores de soporte.

Las SVM parten de una función Φ implícita que transforma los datos de entrada en un espacio de características de alta dimensión definido por alguna función tipo kernel, como por ejemplo una función que devuelve el producto interno $\langle \Phi(x), \Phi(x') \rangle$ relativa a la operación entre las imágenes de dos puntos de datos x, x´ en el espacio de características donde se realiza el aprendizaje. Utilizando una aplicación de transformación no lineal $\Phi: X \rightarrow H$, el producto escalar $\langle \Phi(x), \Phi(x') \rangle$ se puede representar mediante una función kernel k:

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle \tag{13.1}$$

Una propiedad interesante de las máquinas de vectores soporte y cualquier otro sistema basado en kernels es que, una vez que se ha escogido una función kernel válida, se puede trabajar en espacios de cualquier dimensión con poco costo computacional.

Otra ventaja de las SVMs y los métodos kernel es que se puede diseñar un kernel para un problema concreto que podría aplicarse directamente a los datos sin la necesidad de un proceso de extracción. Esto es importante en los problemas donde el proceso de extracción pierde mucha estructura de datos.

Un análisis detallado de las máquinas de vectores soporte se encuentra en los Apéndices al final de este trabajo.

13.1. Paquete e1071 de r-project

La mayor parte del software existente se ha desarrollado en C++, como el galardonado libsvm (Chang and Lin 2001), el cual se empleó como base de desarrollo del paquete e1071 de r-project.

Así, de nuevo se vuelve a usar R como entorno de programación debido a su enfoque al mundo estadístico y su capacidad de software libre heredada del lenguaje S. El paquete de r-project empleado ha sido el e1071 que presenta un interface a libsvm, heredando así su robustez, aunque también complementado con la posibilidad de visualización y localización de parámetros óptimos con la función tune. Este paquete es fruto de los trabajos de David Meyer y su equipo (Technische Universität en Wien, Austria), e incluye los kernels más usuales (lineal, polinomial, RBF, and sigmoidal), presentando capacidades de multiclasificación y validación cruzada entre otras.

Básicamente, las SVM permiten clasificar una población en función de la partición en subespacios de múltiples variables construyendo un hiperplano óptimo como superficie de decisión, de modo que el margen de separación entre las dos clases en los datos se amplía al máximo. Los vectores soporte se refieren a las observaciones de entrenamiento como soporte para ubicar óptimamente la superficie de decisión.

El entrenamiento de una SVM consiste en transformar los datos de entrada en un espacio de características de altas dimensiones, definiendo el kernel, para seguidamente resolver un problema de optimización cuadrática que permita ajustar el hiperplano óptimo y clasificar las transformadas características en dos tipos, siendo el número de características definido por el número de vectores de soporte.

Por otra parte, tal como ya se ha explicado previamente, se ha ejecutado el algoritmo EPSO en 100 ocasiones obteniendo una nube de puntos que se

emplearan como datos de aprendizaje en el algoritmo SVM. Debido a la filosofía de programación, los datos de partida proceden de una matriz llamada matrizfact, la cual presenta la siguiente estructura de información entropia<-matrizfact[,1], factMmmm<-matrizfact[,2], factdecv<-matrizfact[,3] y factdeca<-matrizfact[,4].

Lo primero es configurar las variables empleando dataframes, una clase de objetos especial en r-project de más alto nivel que las matrices: framematrizfact<-data.frame(entropia,factMmmm,factdecv,factdeca).

Se toman los datos de partida, de los cuales se ha considerado el 90% para entrenamiento (training) y el restante 10% para prueba, verificación y/o validación (test). Recuérdese que estos datos se encuentran ordenados por la entropía global de menor a mayor donde se ha considerado más conveniente que los datos de entrenamiento sean los de entropía más baja y los de prueba los de entropía más alta, dado que lo que se pretende es efectuar el proceso de verificación con las entropías más altas debido a que, como se ha visto previamente, alcanzar entropías más altas se va volviendo cada vez más complejo en el procedimiento general. Con estos datos se construye una SVM llamando al paquete e1071.

aprendizajematrizfact<-framematrizfact[1:90,]

pruebamatrizfact<-framematrizfact[91:100,]

Para factMmmm, se parte del aprendizaje:

svm_ajustefactMmmm<-
svm(factMmmm~entropia+factdecv+factdeca,data=aprendizajematrizfact)

Con los siguientes parámetros de ajuste

SVM-Type: eps-regression

SVM-Kernel: radial

cost: 1

gamma: 0.3333333

epsilon: 0.1

Number of Support Vectors: 85

En cuanto a la verificación:

svm_prediccionfactMmmm<-
predict(svm_ajustefactMmmm,newdata=pruebamatrizfact)

Una forma rápida de analizar los datos proporcionados por el algoritmo de predicción es calcular el error respecto a los datos de partida:

errorfactMmmm<-sqrt((sum((pruebamatrizfact$factMmmm-
svm_prediccionfactMmmm)^2))/nrow(pruebamatrizfact))<-6.045678

Se puede repetir este cálculo para otros tipos de kernels:

svm_ajustefactMmmm<-
svm(factMmmm~entropia+factdecv+factdeca,data=aprendizajematrizfact,kernel
="polynomial")

Parámetros:

SVM-Type: eps-regression

SVM-Kernel: polynomial

cost: 1

degree: 3

gamma: 0.3333333

coef.0: 0

epsilon: 0.1

Number of Support Vectors: 83

svm_prediccionfactMmmm<-
predict(svm_ajustefactMmmm,newdata=pruebamatrizfact)

errorfactMmmm<-sqrt((sum((pruebamatrizfact$factMmmm-
svm_prediccionfactMmmm)^2))/nrow(pruebamatrizfact))<-18.54227

Y así sucesivamente, para el kernel linear el error es 7.170325 y para el kernel sigmoid el error es 10.3293. Si se emplea un tipo de nu-regression dentro de los parámetros de la svm, se obtiene para kernel radial un error de 4.918624, para kernel polynomial el error es 7.796327, para kernel linear el error es 6.587848, y para kernel sigmoid el error es 15.81406. Por tanto el error más pequeño se obtiene para nu-regression y kernel radial aunque seguido de cerca por eps-regression y kernel radial.

En el caso de factdecv con eps-regression, con kernel radial el error es 1.00045, para kernel polynomial el error es 5.888459, para kernel linear el error 1.397309 y para kernel sigmoid el error es 6.310831. En el caso de nu-regression, con kernel radial el error es 1.061577, con kernel polynomial el error es 9.474694, con kernel linear el error es 1.007115 y con kernel sigmoid el error es 4.271177. Por tanto el error más pequeño se obtiene para kernel radial con eps-regression seguido de cerca por nu-regression y kernel linear.

En el caso de factdeca con eps-regression, con kernel radial el error es 0.6304728, para kernel polynomial el error es 4.082512, para kernel linear el error 0.8371481 y para kernel sigmoid el error es 1.985999. En el caso de nu-regression, con kernel radial el error es 0.5972637, con kernel polynomial el error es 3.002158, con kernel linear el error es 0.6955169 y con kernel sigmoid el error es 1.709564. En este caso se tienen varios valores que proporcionan buenos resultados con errores pequeños, como nu-regression y kernel radial, y algo peor eps-regression con kernel radial.

Un análisis más detallado puede realizarse con r-project del paquete e1071, que usa la función tune() de visualización y localización de parámetros óptimos. Para ello, el modelo realiza un análisis de validación cruzada empleando una búsqueda aleatoria, para construir a continuación un conjunto de splits que constituyen la base del proceso de entrenamiento. El parámetro cost significa el grado de penalización que permite el modelo; es decir, el algoritmo busca la mejor hipersuperficie que separe los datos y el costo para encontrar ese plano requiere tolerar posibles datos que afectan a la estimación.

Para factMmmm, y por simplicidad en la ejecución del algoritmo se toma eps-regression y kernel radial en lugar de nu-regression y kernel radial, aunque este último proporcionaba el error más pequeño. Posteriormente se volverá sobre ello.

tunefactMmmm<-
tune(svm,factMmmm~entropia+factdecv+factdeca,data=pruebamatrizfact,ranges=list(gamma=2^(-1:1),cost=2^(2:9)))

ranges es una lista con nombre de vectores parámetros que abarca el espacio de muestreo y va a depender de la experiencia del operador.

- best parameters:

gamma cost

0.5 4

- best performance: 23.04368

tunefactMmmm se puede dibujar con plot(tunefactMmmm) con cost y gamma.

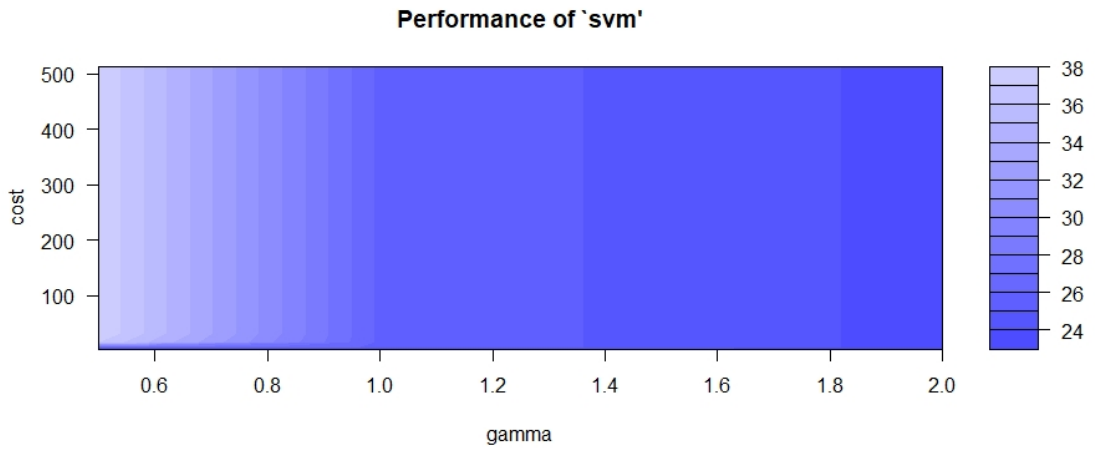

Figura 13.2. Representación obtenida con tune para cost y gamma.

También puede obtenerse información detallada con summary(tunefactMmmm)

- best parameters:

gamma cost

0.5 4

- best performance:23.04368

- Detailed performance results:

	gamma	cost	error	dispersion
1	0.5	4	23.04368	20.18678
2	1.0	4	24.30662	20.35224
3	2.0	4	23.60904	19.35093
4	0.5	8	28.86384	19.84004
5	1.0	8	25.79109	20.04351
6	2.0	8	23.60904	19.35093
7	0.5	16	37.02417	26.64987
8	1.0	16	25.79109	20.04351
9	2.0	16	23.60904	19.35093
10	0.5	32	37.96708	29.02533
11	1.0	32	25.79109	20.04351
12	2.0	32	23.60904	19.35093
13	0.5	64	37.96708	29.02533
14	1.0	64	25.79109	20.04351
15	2.0	64	23.60904	19.35093
16	0.5	128	37.96708	29.02533
17	1.0	128	25.79109	20.04351
18	2.0	128	23.60904	19.35093
19	0.5	256	37.96708	29.02533
20	1.0	256	25.79109	20.04351
21	2.0	256	23.60904	19.35093

22 0.5 512 37.96708 29.02533

23 1.0 512 25.79109 20.04351

24 2.0 512 23.60904 19.35093

Si en la formula anterior se entra con épsilon en lugar de con gamma:

tunefactMmmm1<-
tune(svm,factMmmm~entropia+factdecv+factdeca,data=pruebamatrizfact,ranges=list(epsilon=seq(0,1,0.1),cost=2^(2:9)))

- best parameters:

epsilon cost

0.6 4

- best performance: 18.35122

Análogamente:

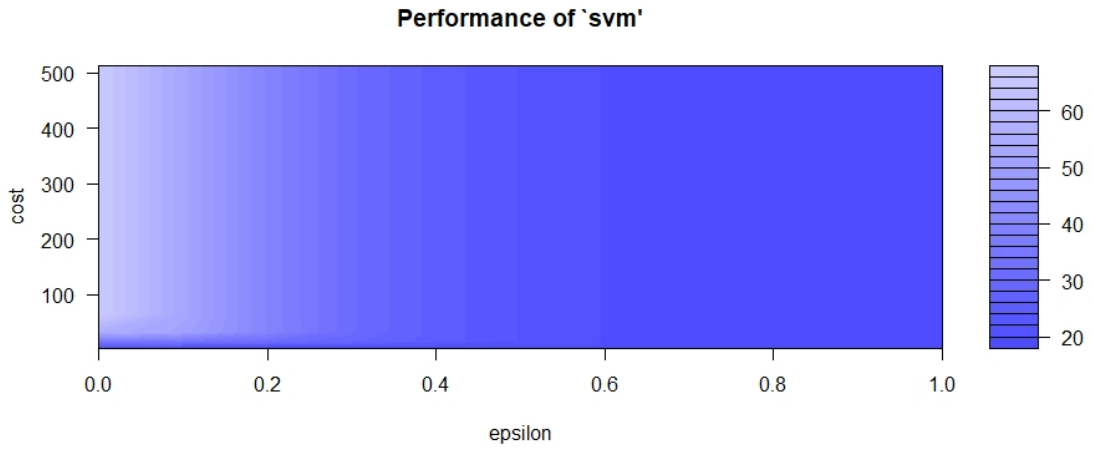

Figura 13.3. Representación obtenida con tune para cost y epsilon.

Una optimización de los resultados se puede conseguir mediante

besttunefactMmmm<-
best.tune(svm,factMmmm~entropia+factdecv+factdeca,data=pruebamatrizfact,ranges=list(gamma=2^(-1:1),cost=2^(2:9)))

summary(besttunefactMmmm)

Parámetros:

SVM-Type: eps-regression

SVM-Kernel: radial

cost: 4

gamma: 0.5

epsilon: 0.1

Number of Support Vectors: 10

Este mismo resultado se consigue con

besttunefactMmmm1<-tunefactMmmm$best.model

Realizando la verificación con este nuevo resultado:

predictbesttunefactMmmm<-predict(besttunefactMmmm,pruebamatrizfact)

Y calculando el error respecto a los datos de partida de nuevo:

besterrorfactMmmm<-sqrt((sum((pruebamatrizfact$factMmmm-predictbesttunefactMmmm)^2))/nrow(pruebamatrizfact))<-1.820814

Ostensiblemente mejor que el resultado calculado anteriormente sin optimizar (6.045678).

Por otra parte, ahora con épsilon:

besttunefactMmmm1<-tunefactMmmm1$best.model

Obteniendo

SVM-Type: eps-regression

SVM-Kernel: radial

cost: 4

gamma: 0.3333333

epsilon: 0.6

Number of Support Vectors: 8

Realizando la verificación con este nuevo resultado:

predictbesttunefactMmmm1<-predict(besttunefactMmmm1,pruebamatrizfact)

Y calculando el error respecto a los datos de partida de nuevo:

besterrorfactMmmm1<-sqrt((sum((pruebamatrizfact$factMmmm-predictbesttunefactMmmm1)^2))/nrow(pruebamatrizfact))<-2.608981

El cual es peor que el anterior (1.820814) lo cual hace pensar en emplear gamma en lugar de epsilon.

Volviendo al comienzo con gamma y empleando nu-regression y kernel radial:

besttunefactMmmm2<-best.tune(svm,factMmmm~entropia+factdecv+factdeca,data=pruebamatrizfact,ranges=list(gamma=2^(-1:1),cost=2^(2:9)),type="nu-regression")

predictbesttunefactMmmm2<-predict(besttunefactMmmm2,pruebamatrizfact)

Y calculando el error respecto a los datos de partida de nuevo:

besterrorfactMmmm2<-sqrt((sum((pruebamatrizfact$factMmmm-predictbesttunefactMmmm2)^2))/nrow(pruebamatrizfact))<-0.002162658

Este resultado empleando técnicas de optimización svm de r-project ha resultado excelente a raíz del error de estimación obtenido prácticamente nulo con la técnica desarrollada en este trabajo.

Ahora, para el caso de factdecv con eps-regression y kernel radial, que se ha visto antes proporcionaba el error más pequeño, se procede análogamente con el análisis de la función tune() de r-project empleando el parámetro gamma.

tunefactdecv<-tune(svm,factdecv~entropia+factMmmm+factdeca,data=pruebamatrizfact,ranges=list(gamma=2^(-1:1),cost=2^(2:9)))

- best parameters:

gamma cost

2 64

- best performance: 0.6206965

tunefactdecv se puede dibujar con plot(tunefactdecv) con cost y gamma.

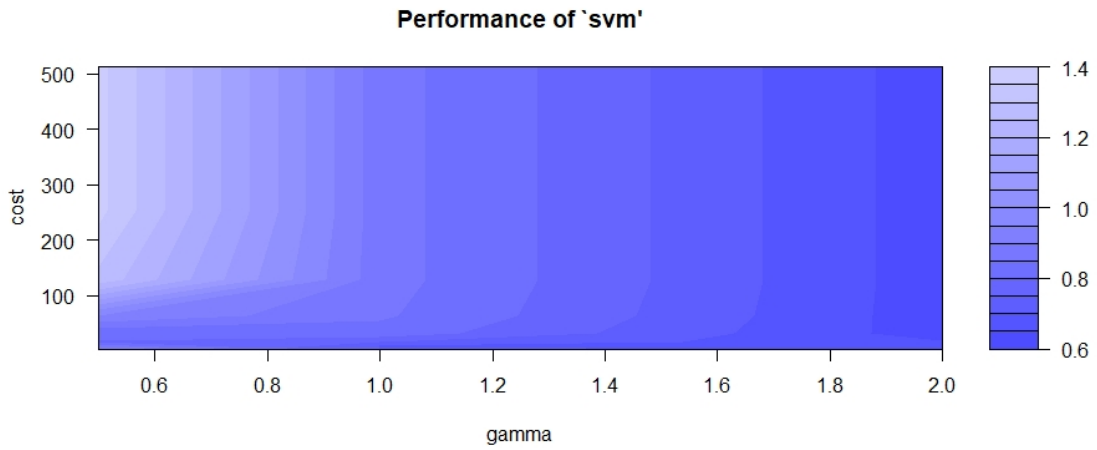

Figura 13.4. Representación obtenida con tune para cost y gamma.

Una optimización de los resultados se puede conseguir mediante

besttunefactdecv<-
best.tune(svm,factdecv~entropia+factMmmm+factdeca,data=pruebamatrizfact,ranges=list(gamma=2^(-1:1),cost=2^(2:9)))

summary(besttunefactdecv)

Parámetros:

SVM-Type: eps-regression

SVM-Kernel: radial

cost: 64

gamma: 2

epsilon: 0.1

Number of Support Vectors: 8

Realizando la verificación:

predictbesttunefactdecv<-predict(besttunefactdecv,pruebamatrizfact)

Y calculando el error respecto a los datos de partida de nuevo:

besterrorfactdecv<-sqrt((sum((pruebamatrizfact$factdecv-predictbesttunefactdecv)^2))/nrow(pruebamatrizfact))<-0.08270507

Este resultado empleando técnicas de optimización svm de r-project ha mejorado considerablemente el resultado obtenido inicialmente haciendo uso de la técnica desarrollada en este trabajo al alcanzar un error bastante pequeño.

Considerando el caso de factdeca con nu-regression y kernel radial, que se ha visto antes proporcionaba el error más pequeño, se procede análogamente con el análisis de la función tune() de r-project empleando el parámetro gamma.

tunefactdeca<-tune(svm,factdeca~entropia+factMmmm+factdecv,data=pruebamatrizfact,ranges=list(gamma=2^(-1:1),cost=2^(2:9)),type="nu-regression")

- best parameters:

gamma cost

2 4

- best performance: 0.3555831

tunefactdeca se puede dibujar con plot(tunefactdeca) con cost y gamma.

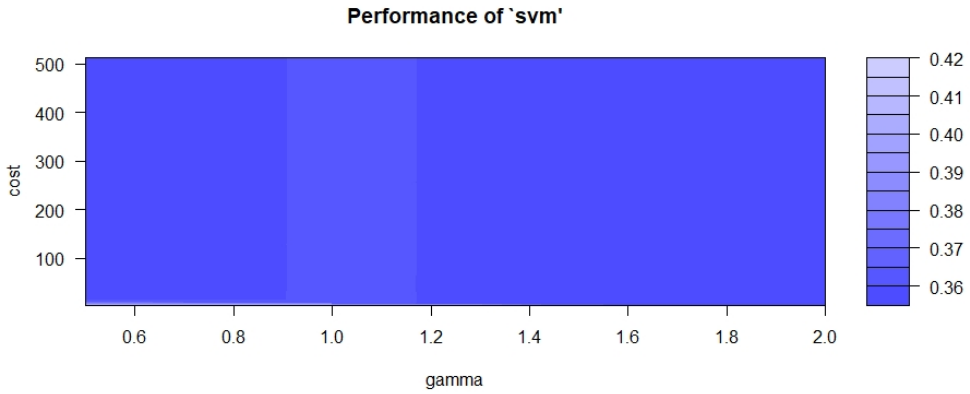

Figura 13.5. Representación obtenida con tune para cost y gamma.

Una optimización de los resultados se puede conseguir mediante

besttunefactdeca<-
best.tune(svm,factdeca~entropia+factMmmm+factdecv,data=pruebamatrizfact,ranges=list(gamma=2^(-1:1),cost=2^(2:9)),type="nu-regression")

summary(besttunefactMmmm)

Parámetros:

SVM-Type: eps-regression

SVM-Kernel: radial

cost: 4

gamma: 2

nu: 0.5

Number of Support Vectors: 10

Realizando la verificación:

predictbesttunefactdeca<-predict(besttunefactdeca,pruebamatrizfact)

Y calculando el error respecto a los datos de partida de nuevo:

besterrorfactdeca<-sqrt((sum((pruebamatrizfact$factdeca-
predictbesttunefactdeca)^2))/nrow(pruebamatrizfact))<-0.0001862315

Este resultado empleando técnicas de optimización svm de r-project ha mejorado considerablemente el resultado obtenido inicialmente haciendo uso de la técnica desarrollada en este trabajo al llegar a un resultado con un error muy pequeño.

Así, de esta forma, se puede concluir que la aplicación de técnicas SVM, concretamente el paquete e1071 de r-project, permite confeccionar un procedimiento de cálculo que toma un conjunto de simulaciones de partida como datos de aprendizaje para entrenar la SVM y construir un modelo que predice el resultado y proporciona una respuesta para una nueva muestra. Especialmente importante ha sido la contribución para ello de la función tune del paquete e1071 con su capacidad de visualización y localización de

parámetros óptimos mediantes técnicas de validación cruzada con búsqueda aleatoria.

Realizando la verificación a partir de haber empleado las capacidades de optimización de la función tune y calculando el error respecto a los datos de partida, se ha mostrado como estos errores eran prácticamente cero, mejorando sustancialmente los errores calculados antes de optimizar con tune. Para finalizar se ha representado los factores de decisión factMmmm, factdecv y factdeca en función de la entropía originales en comparación con los obtenidos optimizados con la función tune, mostrando un paralelismo casi coincidente.

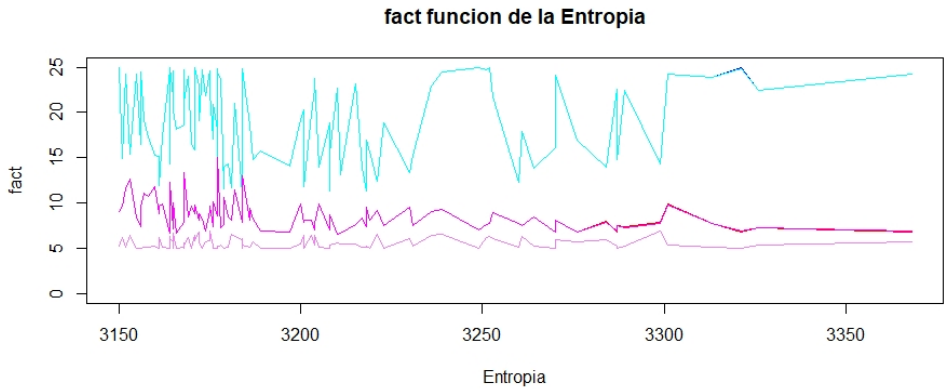

Figura 13.6. Factores de decisión factMmmm, factdecv y factdeca función de la entropía optimizados con tune.

Capítulo 14. Conclusiones (y trabajo futuro)

En ocasiones han aparecido anuncios en prensa acerca de enjambres de RPAs realizando misiones de índole operativa cuando en realidad se trataba de meras formaciones libres de RPAs que no trabajan de forma cooperativa. Estrictamente hablando, a lo largo del presente trabajo se ha podido ver que un enjambre de RPAs involucra muchos más factores técnicos que una agrupación de individuos, donde un conjunto de RPAs se coordinan bajo el paraguas de la inteligencia artificial, comunicándose de forma segura en vuelo, detectando y evitando obstáculos, colaborando y cooperando entre ellos, y actuando autónomamente en función de las condiciones cambiantes del entorno.

El imparable aumento de los RPAs que componen un enjambre conlleva a una arquitectura física de enjambre cada vez más compleja, donde el piloto ya no puede actuar sobre cada vehículo remoto, viéndose en la necesidad de coordinar y controlar a todo el enjambre, auxiliándose de tecnologías tales como de redes de comunicaciones avanzadas que involucren al enjambre completo, nuevos protocolos de reciprocidad y ayuda mutua entre individuos donde cada una de las acciones de cada RPA se adapta y optimiza hacia un comportamiento de inteligencia colectiva, lo cual, a la postre, ha venido en denominarse inteligencia de enjambre, como la aplicación de la inteligencia artificial a los enjambres, o lo que es lo mismo, la capacidad de una máquina o un conjunto de máquinas de tomar la mejor decisión entre varias posibilidades.

Sin embargo, ya se anticipó que antes de integrar, organizar y sincronizar un grupo de RPAs, se deben resolver otros problemas como el de las colisiones, cuestión ciertamente importante que cobra más relevancia en la medida que

aumenta el número de vehículos remotos que componen el enjambre. En este campo, se han significado los avances realizados por diversos investigadores, como Reynolds que estableció las primeras reglas en simulación de enjambres, y posteriormente, las certeras conclusiones del KAIST en el campo de la elusión de colisiones, como así lo ha atestiguado el Congreso de los Estados Unidos en su documento Mini, Micro, and Swarming Unmanned Aerial Vehicles: a baseline study. De esta forma, el KAIST efectuó múltiples simulaciones de conducta de enjambres, donde partiendo del enjambre como un grupo descentralizado, cada agente del enjambre maniobra para evitar colisionar con otros RPAs del enjambre, llegando a la conclusión de que una zona de seguridad con una dimensión de 5 a 15 veces la longitud media del RPA, permite evitar las colisiones en su mayor parte, cuestión que ha sido corroborada en este trabajo mediante los algoritmos de cálculo desarrollados.

También cabe destacar otro avance del KAIST, que aplicó la técnica de navegación proporcional (NP), utilizada con éxito en el guiado de misiles, como base de desarrollo de una herramienta de evitación de colisiones. Mediante modelizaciones matemáticas, el KAIST encontró una ley de elusión de colisiones entre los RPAs del enjambre; no obstante, en el presente trabajo se ha llegado a la conclusión que la ley NP resulta más apropiada para mantener el vuelo en formación que para acometer misiones operativas.

De este modo, tomando como punto de partida los postulados de Reynolds y el KAIST, en el presente trabajo se ha implementado un algoritmo de elusión de colisiones con RPAs del propio enjambre, mediante la definición de tres variables de decisión (factdeca, factdecv y factMmmm). Esencialmente, el procedimiento implementado calcula para cada RPA las distancias al resto de RPAs, y se asocia al primer vehículo remoto la distancia más corta (dis), de manera que cuando dis es inferior a factdecv, el RPA efectúa un viraje de acuerdo a la velocidad angular de viraje proporcionada por la Mecánica del Vuelo. Si aun así, sigue existiendo riesgo de colisión y dis es inferior a factdeca (factdecv>=factdeca), el RPA se decelera pero sin llegar a la pérdida. Sin embargo, también se ha visto en este trabajo que es necesario aplicar un algoritmo que evite que el enjambre se disperse y disgregue mediante la aplicación de un factor llamado factMmmm, que provoca que al sobrepasar

este valor, el RPA vire y se dirija al centroide del enjambre. Indudablemente un asunto más complejo es el impacto con elementos ajenos al enjambre o infraestructuras, lo cual involucra algo más que la información relativa a la posición de los otros RPAs, como información de sensores del entorno exterior. Así, también se han explorado en este trabajo tecnologías de tratamiento estadístico de imágenes con las que se podría identificar un obstáculo evitando su colisión.

Este tipo de tecnologías de enjambres resultaría de aplicación a múltiples misiones operativas; no obstante, para este trabajo se han seleccionado dos misiones prototipo para mostrar la potencia de este tipo de tecnologías. En primer lugar, la misión de saturación de defensas liderada por un enjambre de RPAs sería un clarísimo candidato a la hora de neutralizar y/o degradar los sistemas defensivos enemigos, lo cual daría ocasión a que otras operaciones aéreas puedan desarrollarse sin que se produzcan pérdidas innecesarias humanas o/y materiales. Este grupo de vehículos remotos jerarquizados en forma de enjambres saturarían las defensas enemigas inundando el espacio aéreo de numerosos objetivos con poco valor económico, los cuales serían derribados por carísimos misiles o bien provocarían a los radares enemigos que al encenderse revelarían su posición lo que sería aprovechado por las fuerzas propias para destruirlos. De esta manera tan simple, un sistema de defensa aérea, proyectado para repeler agresiones de misiles y grandes aviones, podría ser abrumado, colapsado y confundido por un grupo de pequeños y baratos RPAs que además, potenciarían su firma radar con paneles radar con el objetivo de ser confundidos con aviones mayores.

En segundo lugar, otra misión operativa que podría ser acometida por un enjambre de RPAs sería la ISR al ser capaces estos vehículos remotos de integrarse cooperativamente en los procesos de adquisición, procesamiento y recopilación de información de inteligencia, dado su elevado número y pequeño tamaño. Este tipo de misiones presenta el inconveniente de que se desarrollan en teatros de operaciones “calientes” con alto riesgo de derribo, lo que hace inevitable para estas aeronaves cumplir la misión efectuando constantes maniobras evasivas a los efectos de decepcionar y desconcertar a las defensas aéreas enemigas, sin perder de vista que estos enjambres actúan

dentro de una estructura sincronizada, inteligente y colaborativa con el fin de incrementar la probabilidad de supervivencia del conjunto de RPAs, viéndose así el paralelismo físico matemático entre las misiones ISR y de saturación de defensas.

Ciertamente, los RPAs se encuentran en vías de alcanzar su madurez tecnológica; si bien, en el campo de los enjambres dicha situación dista bastante de alcanzar dicho estadio, situándose más bien este caso en el germen inicial. La razón fundamental de ello es que es necesario el desarrollo de avanzados algoritmos de inteligencia artificial que articulen una red epistemológica sinérgica fruto de la adición de capacidades entre los RPAs del enjambre, de tal modo que estos aparatos exhiban una conducta de cooperación y coordinación, inteligente, sincronizada y organizada.

En este punto, ha sido necesario fundamentar el concepto de inteligencia artificial como una serie de algoritmos organizados en programas computacionales, que para uno de los fundadores de la inteligencia artificial, Nils John Nilsson, debe basarse, básicamente, en cuatro apoyos: búsqueda de estados, metodologías de optimización evolutivas, sistemáticas de aprendizaje automático y razonamiento lógico formal.

Yendo por partes, parece evidente que el enjambre aumentará el efecto degradante, saturador, y neutralizante sobre el sistema defensivo enemigo, potenciando así la confusión, decepción, desconcierto y colapso, si cada RPA del enjambre se desenvuelve de manera diferente respecto al resto de vehículos remotos, siendo la velocidad de cada RPA diferente de la del resto. Así, resulta patente que una gran cantidad de objetos volando en diferentes direcciones proporcionará una mayor carga de trabajo al sistema de defensa aérea enemigo. Ello ha motivado la necesidad de definir en este trabajo una función de entropía del enjambre, concebida como el número de estados posible, que se ha materializado como una medida del número de RPAs con distintas velocidades.

Tras definir la entropía, queda patente que el mayor efecto saturador, degradativo y neutralizante sobre el sistema de defensa enemigo se alcanza maximizando la entropía como función de factdeca, factdecv y factMmmm. En este trabajo, se ha podido comprobar como esta función es bastante

discontinua, variable con el tiempo, estridente, y en definitiva con numerosos óptimos locales, tras lo cual en el presente trabajo se ha implementado y programado un algoritmo EPSO (Evolutionary Particle Swarm Optimization) donde las ya pujantes características del algoritmo PSO se ven reforzadas agregando elementos evolutivos (mutación, selección y cruce). De esta manera, el algoritmo EPSO parte de un trío factdeca, factdecv y factMmmm, y entropía asociada, y es capaz de alcanzar paulatinamente sucesivos tríos con entropías superiores, desechando los que presentan entropías más bajas, estableciendo un pseudorazonamiento de tal forma que partiendo de un grupo de tríos se alcanza un trio diferente de los primeros con entropía aumentada.

El enjambre al que se ha orientado este trabajo ha sido un enjambre de RPAs de pequeño tamaño, aunque técnicamente podría aplicarse a otros tamaños, si bien los RPAs diminutos presentan importantes ventajas como su maniobrabilidad, baja firma radar, furtividad, a lo que se une su bajo valor económico frente a los grandes. Sin embargo, este pequeño tamaño no puede transportar grandes microprocesadores lo que conduce a una baja capacidad computacional y obviamente a una lentitud en los cálculos. De cualquier forma, resulta notorio que las misiones operativas demandan celeridad en los procesos de toma de decisiones por lo que en este trabajo se ha confeccionado un procedimiento consistente en realizar una serie de simulaciones, previamente al despliegue del enjambre, y aplicando un algoritmo de aprendizaje automático tipo SVM (Support Vector Machine) utilizar estas simulaciones como elementos de aprendizaje para entrenar al SVM y construir un modelo de predicción que proporcione una respuesta para una muestra nueva, de tal manera que el enjambre pueda tomar ciertas decisiones por sí mismo. Para lo cual, se ha empleado la función tune del paquete e1071 de r-project que permite hallar los parámetros óptimos a través de técnicas de validación cruzada con búsqueda aleatoria.

Por consiguiente, en este trabajo se ha elaborado una serie de algoritmos que abarcan un EPSO de optimización evolutiva y búsqueda de estados junto a una SVM de aprendizaje automático, que parte de valores de entropía más baja para encontrar un valor de entropía más alta, lo cual delimita una técnica del tipo de inteligencia artificial capaz de tomar hasta cierto punto la mejor decisión

entre varias posibilidades en cuanto a que valores proporcionan la entropía más alta en un enjambre de RPAs, lo que en última instancia genera un algoritmo ideado en suma como un sistema de inteligencia de enjambres.

14.1. Líneas futuras

En una tecnología tan emergente como esta y en un estadio tan germinal, múltiples líneas futuras de investigación podrían abrirse. En el campo de las colisiones, sigue apostándose por las tecnologías “Sense & Avoid”, aún en desarrollo, que de cualquier forma no están previstas para ser de aplicación a RPAs de pequeño tamaño y aun así requerirían un ingente trabajo de investigación para que pudieran ser miniaturizadas. En definitiva, cualquier tecnología que permita detectar la presencia de obstáculos y proponer una maniobra elusiva será bienvenida aunque requerirá un proceso de investigación exhaustivo (como por ejemplo LIDAR, radar, acústica, IR, EO, DIC,…).

A este respecto, los análisis y estudios llevados a cabo por el KAIST en el campo de las colisiones resultan prometedores pero deberían ser complementados con estudios más profundos que contemplen más variables y opciones de cálculo. De hecho, el presente trabajo no deja de ser un análisis que surge a partir de las conclusiones establecidas por el KAIST.

Por otra parte, es necesario estudiar la aplicación de los enjambres de RPAs a mayor cantidad de misiones ya sean operativas o no, como búsqueda y rescate de personas, control de fronteras, control de áreas incendiadas, en análisis de datos de AWACs, etc.

De cualquier forma, y teniendo en cuenta los esfuerzos dedicados por diferentes centros de investigación en todo el mundo, parece inevitable la aplicación de algoritmos metaheurísticos a diferentes facetas de los enjambres de RPAs, como es el caso de la determinación de algoritmos de caminos óptimos y búsqueda de rutas autónomas hacia el blanco minimizando el tiempo en el teatro de operaciones, lo cual no deja de ser baladí a tenor de la gran cantidad de algoritmos metaheurísticos existentes en la actualidad que podrían ser de aplicación.

Otro problema que se ha anticipado a lo largo de este trabajo es la necesidad de estimular las comunicaciones seguras de alto nivel a la par que veloces en zona de conflicto entre una gran cantidad de RPAs, lo cual incrementa la complejidad computacional con el consiguiente inconveniente de lentitud en los cálculos. Esto intenta paliarse recurriendo a los microprocesadores del resto de agentes del enjambre (ultraswarm) que se combinan para realizar cálculos complejos dentro de una red, o bien incluso en un centro de computación en tierra si las comunicaciones así lo permitieran.

También se ha aseverado a lo largo del trabajo que el mayor efecto neutralizante, saturador y de degradación sobre un sistema defensivo enemigo se conseguirá maximizando la función de entropía tal como se ha definido; no obstante, este concepto debería ser confirmado mediante simulaciones en ejercicios auténticos con el fin de evaluar el grado real de decepción, confusión desconcierto y colapso, de tal modo que la definición de entropía sea revisada y ajustada.

Finalmente, parafraseando a Nils John Nilsson, de los cuatro pilares, mencionados previamente, en los que debe basarse la inteligencia artificial, quizás el más importante sea el aprendizaje automático. En este sentido, no se tiene claro si la SVM es el más eficiente o no en este tipo de problemas, por lo que una nueva línea de investigación que debería abrirse es si otras técnicas de aprendizaje automático (redes neuronales artificiales, redes bayesianas, deep learning,…) representan alguna ventaja frente al SVM.

Apéndice A. Navegación proporcional en el vuelo en formación de un enjambre de RPAs

La idea básica en el empleo de la navegación proporcional al vuelo en formación de un enjambre de RPAs es simple; un RPA que no sea el líder del enjambre sigue a otro RPA sin colisionar con él ni con elementos ajenos al enjambre.

En un enjambre, los RPAs varían su posición en todo momento con lo cual no adoptan posiciones fijas sino temporales. Así, en el vuelo en formación la posición de cada RPA varía aunque el enjambre conserve su aspecto habitual. Es común imaginar al enjambre adoptando figuras geométricas aunque no hay razón concreta para ello.

Para el vuelo en formación, un RPA será el líder. Cada formación volará como una única aeronave en la parte que corresponde a la notificación de posición y a la navegación. Es vital que la separación entre los RPAs que intervienen en el vuelo sea tal que en ninguna circunstancia se pueda producir colisión alguna, manteniendo la cohesión del enjambre al mismo tiempo. El enjambre también debe evitar colisionar con elementos no pertenecientes al enjambre.

Dos o más RPAs se desplazan en formación cuando realizan maniobras como si fueran uno solo desde un punto de vista planificado, siguiendo las órdenes del líder, el cual es único en cada formación y es el encargado de tomar decisiones. No obstante, debe existir un protocolo de cambio de liderazgo por

si líder sufriera algún percance existiendo en todo momento líderes de reemplazo. Si hubiera dos líderes es porque hay dos formaciones.

Guiado es el conjunto de operaciones que tiene por objeto conducir a cada RPA desde el punto de lanzamiento al objetivo que se le marque. Esto requiere realizar una serie de tareas:

1. Determinar la posición relativa RPA seguidor – RPA seguido o bien un obstáculo a evitar si se acercará a él demasiado.
2. Elaborar las órdenes correctoras de la posición del RPA seguidor.
3. Interpretar las órdenes y accionar los dispositivos mecánicos que permiten variar la posición del RPA seguidor.
4. Verificar la exactitud de las acciones emprendidas y corregir en caso necesario.
5. Evitar las colisiones con otros RPAs del enjambre así como elementos ajenos al enjambre.

El sistema que interpreta las órdenes y acciona los elementos mecánicos se localiza en cada RPA. No ocurre lo mismo con los sistemas detector de la posición relativa RPA seguidor – RPA seguido (obstáculo en lugar de RPA seguido) y el elaborador de órdenes, que pueden estar dentro del RPA seguidor o ser externos a él, y situarse en la GCS.

La idea planteada en este trabajo es que cada agente del enjambre reciba información del resto del enjambre conteniendo, entre otros parámetros, la posición espacial del resto de RPAs del enjambre, con el objetivo de que la distancia de seguridad se conserve dentro de los límites indicados previamente y se puedan evitar las colisiones entre los RPAs del enjambre, a la par que se mantenga la cohesión del enjambre. No obstante, la navegación proporcional proporciona información adicional de posición de los RPAs del propio enjambre e información de posición de entes ajenos al enjambre como otras aeronaves y obstáculos.

En cuanto a las colisiones con entes ajenos al enjambre, cada RPA podría emitir algún tipo de energía de radiación hacia los obstáculos que encuentre en su camino recibiendo un eco de rebote. Estos obstáculos son entes ajenos al enjambre ya sean otras aeronaves o incluso otro tipo de elementos. El guiado

podría ser semiactivo si la radiación es emitida por algún sistema exterior al RPA y el propio RPA detectará el eco. Podría emplearse un láser con designador, radar, etc. El guiado sería pasivo si el RPA se limita a recoger la radiación propia de los entes entrometidos, bien de forma natural o bien como eco por iluminación exterior. Diversos sistemas podrían emplearse como IR, I2R (Imaging Infrared), TV (Televisión), UV (Ultravioleta), MMW (Millimeter Wave), sonido, etc.

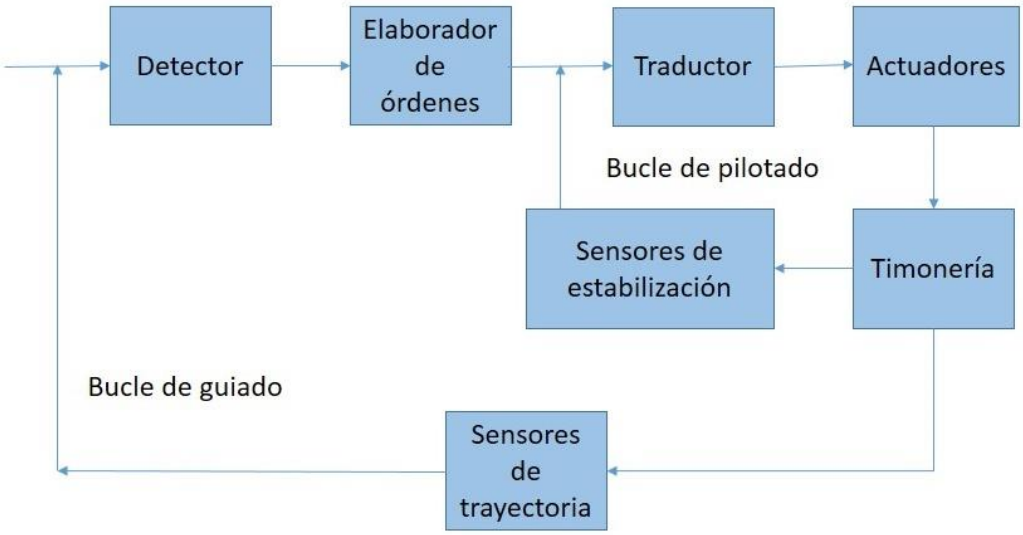

Figura A.1. Diagrama de guiado.

A.1. Guiado

El guiado consistiría en que el detector captaría la posición relativa RPA seguidor – RPA seguido (o bien un obstáculo a evitar) transmitiéndola al elaborador de órdenes. En función de la ley de guiado empleada, este calcula las correcciones necesarias para mantener al RPA en la trayectoria apropiada, ordenándole una aceleración normal comandada, que es la aceleración teórica necesaria que debe adoptar el RPA para corregir la desviación. La señal de aceleración ordenada le llega al bucle de pilotado donde el traductor, interpreta las órdenes y las convierte en señales eléctricas, que activan los actuadores que a su vez mueven los dispositivos de gobierno del RPA y lograr así su maniobrabilidad. De este modo, la mecánica de vuelo intenta convertir la aceleración comandada en aceleración real y el resultado es la aceleración normal ejecutada por el RPA. Un sistema de sensores, entre los que pueden

formar parte acelerómetros y calculadores de estimación miden la desviación que la aceleración ejecutada produce y la realimentan negativamente para producir la señal de error que cierra el bucle de guiado. Los actuadores pueden ser motores eléctricos, hidráulicos, neumáticos, electroimanes, etc. La timonería puede ser aletas, aunque también puede ser algún tipo de propulsor.

Garantizar el vuelo supone controlar la eficacia de las acciones ejecutadas. El bucle de pilotado valora la eficacia de la timonería de gobierno y mediante los sensores de estabilización realimenta al traductor. El bucle de pilotado tiene que controlar la maniobrabilidad del RPA evitando acciones fuertemente exageradas que hagan al RPA ingobernable perdiendo su estabilidad.

Mediante los sensores de trayectoria se constituye el bucle de guiado cuya misión es variar la trayectoria de manera que el RPA seguidor siga al RPA seguido, conforme a la ley de guiado aplicada. Los sensores calculan la posición del RPA seguidor y por comparación efectúan la estabilización y el guiado. Los sensores pueden ser inerciales, acelerómetros, giróscopos, girómetros, altímetros, etc.

Con la exposición hecha hasta aquí se tiene el guiado en lazo abierto. Para controlar el vuelo es necesario fiscalizar la eficacia de las acciones realizadas. Este control se realiza en dos bucles: el bucle de pilotado y el de guiado. Para ello es necesario medir la acción de la timonería de gobierno y se realimenta a través de los sensores de estabilización, para formar el bucle de pilotado, y de los sensores de trayectoria para formar el bucle de guiado.

El bucle de pilotado tiene por objeto controlar la maniobrabilidad del agente, evitando acciones de gobierno tan violentas que provoquen la pérdida de estabilidad aerodinámica del mismo, es decir, que lo hagan ingobernable.

El detector puede ser activo o pasivo, haciendo uso de uno o varios de los siguientes sistemas de radiación:

- Infrarroja (cámara térmica o laser).
- Electromagnética (radar).
- Espectro visible (cámara de TV).

Los sensores tienen por objeto determinar la posición del RPA, de forma que sirvan de comparación para la estabilización y guiado. Típicamente:

- Centrales inerciales
- Giróscopos
- Girómetros
- Acelerómetros
- Altímetros

El detector es fundamental en la cadena de guiado. Este módulo es el encargado de determinar la posición relativa del RPA seguido (o bien de un obstáculo) en relación a una dirección de referencia. Los detectores pueden ser electromagnéticos (radar) y ópticos de luz infrarroja o visible.

El principio del radar detector consiste en que la antena apunta hacia el RPA seguido/obstáculo actuando alternativamente como receptor y emisor de señales Cada cierto tiempo (microsegundos) el transmisor envía un pulso constituido por un tren de ondas, cuya frecuencia es del orden de GHz. El duplexor es un conmutador electrónico encargado de conectar la antena al transmisor o receptor según corresponda. En el instante de la transmisión, el duplexor conecta el transmisor con la antena y el pulso se envía al espacio. Inmediatamente el duplexor conmuta con el receptor y se queda esperando recibir el eco del RPA seguido o bien el obstáculo. Este proceso se repite con una cierta frecuencia. Si se recibe eco, el receptor evalúa la posición del RPA seguido/obstáculo en relación al eje de referencia y al eje de antena. La posición del RPA seguido/obstáculo es enviada a dos electrónicas: la de seguimiento y la de guiado. La electrónica de seguimiento traduce la posición del RPA seguido/obstáculo en órdenes para los servomotores de antena, los cuales la mueven orientándola hacia el RPA seguido/obstáculo. La electrónica de guiado usa la posición del RPA seguido/obstáculo para producir la orden de pilotado del RPA.

El tipo de radar descrito se denomina radar mononopulso. Cuando se necesite detectar la distancia a que se encuentra el RPA seguido/obstáculo se mide el tiempo transcurrido entre la emisión del pulso y la recepción del eco y, dado que la velocidad de la onda es conocida (velocidad de la luz), es fácil determinar la distancia. Si se usa el efecto doppler se puede obtener la velocidad del RPA seguido/obstáculo directamente. Si el radar es de onda continua; es decir, en lugar de pulsos periódicos se envía una señal

permanente, entonces se requieren antenas separadas para la emisión y recepción.

Los detectores ópticos están basados en recopilar la señal visible o infrarroja que emite el RPA seguido u obstáculo, bien de forma natural o bien como eco por iluminación exterior. El detector de infrarrojos recopila las señales del espectro infrarrojo tras pasar por una óptica apropiada.

Bucle de pilotado es la forma en que se actúa sobre los dispositivos del RPA para lograr su maniobrabilidad. El bucle de pilotado está formado por todas aquellas acciones que controlan la maniobrabilidad del RPA. El elaborador de órdenes del bucle de guiado genera, en función del error de posición que se procura corregir, una aceleración normal, que es el comando que se considera que corrige el error. Esta aceleración se transforma en el ángulo de timonería de gobierno del RPA seguidor, que será aquel que debe provocar la aceleración deseada. Los dispositivos de gobierno (bucle de pilotado) originan como respuesta una aceleración ejecutada. Hasta aquí se tiene un pilotado en bucle abierto, donde se supone que la aceleración ejecutada concuerda con la aceleración comandada. Si el error no se corrige, el bucle de guiado ordenará una nueva aceleración comandada. En situaciones muy simples esto puede ser suficiente; sin embargo, la aceleración ejecutada puede diferir bastante de la aceleración comandada debido a las inercias del propio RPA y sus dispositivos de control. En general, la aceleración ejecutada no alcanza el valor comandado después de transcurrir un cierto tiempo, con un régimen transitorio que podría ser inaceptable. Esto exige a introducir un bucle de pilotado, cuyo objetivo es optimizar la respuesta del RPA a la aceleración comandada. El bucle de pilotado puede realizarse por control aerodinámico o bien por empuje de algún impulsor.

A.2. Leyes de guiado. Autoguiado

Una vez localizado el RPA seguido por el RPA seguidor hay que ver la forma en la que el seguidor persigue al seguido. Esto se efectúa mediante alguna ley de guiado que es la conducta que sigue el RPA seguidor para aproximarse al RPA seguido o bien a un obstáculo a evitar. Las leyes de guiado típicas son persecución (pura o diferida), colisión, navegación proporcional y alineamiento.

En la persecución pura el vector velocidad del RPA apunta permanentemente al RPA seguido/obstáculo. Se le conoce ordinariamente como “carrera de perro”. Exige una gran maniobrabilidad lo que a su vez requiere una elevada aceleración normal.

En la colisión, la recta RPA seguidor – RPA seguido se mantiene invariable desplazándose paralela a sí misma.

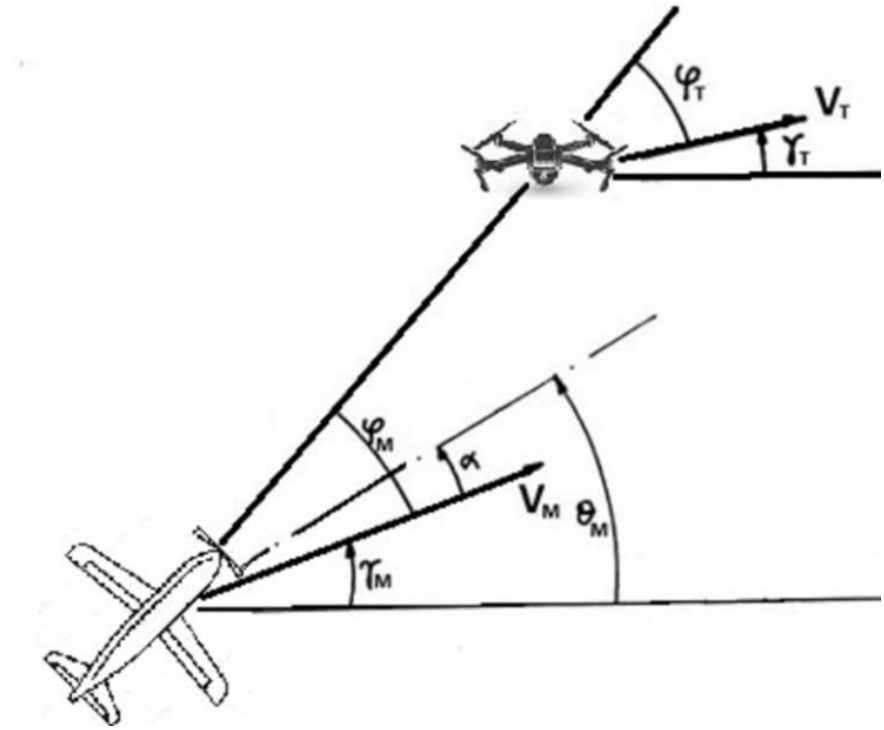

Figura A.2. Guiado con navegación proporcional.

No obstante, la más utilizada es la navegación proporcional que es aquella en la que la velocidad de rotación del vector velocidad del RPA seguidor es proporcional a la velocidad de rotación de la recta RPA seguidor – RPA seguido:

$$\eta_m = \gamma_m + \varphi_m \tag{A.1}$$

$$\dot{\gamma}_m = A\dot{\eta}_m \tag{A.2}$$

A es el coeficiente de navegación proporcional. Obsérvese que si A=1 se tiene el caso de persecución. Una A infinito se corresponde con el caso de colisión. Un guiado con A>l tiene mejor repartido el esfuerzo de la aceleración normal de forma que no requiere una maniobrabilidad tan violenta como la ley de

persecución. Es frecuente usar el denominado coeficiente de navegación proporcional verdadero, definido como:

$$a = A\frac{v_M cos\varphi_m}{V_r} \tag{A.3}$$

Donde V_m es la velocidad del seguidor y v_r la velocidad relativa entre ambos RPAs.

En el alineamiento, se tienen que mantener alineados la GCS, RPA seguidor y RPA seguido. Cuando intervienen un gran número de RPAs, se desecha este tipo de alineamiento en beneficio de la navegación proporcional.

La NP requiere de un detector que mida la variación del ángulo respecto a una dirección fija, mientras que la persecución solo implica detectar el desvío angular respecto al eje longitudinal seguidor. La NP requiere, pues, un equipo detector más complicado que la persecución.

Se dice que se aplica el autoguiado cuando se sigue la ley de navegación proporcional con un coeficiente fijo o adaptativo (ley óptima). En el autoguiado se necesita un dispositivo denominado autodirector, cuya misión es proporcionar al bucle de guiado la velocidad de rotación de la línea RPA seguidor – RPA seguido, pero lo hace a través de dispositivos mecánicos, como girómetros y antenas, que tienen un efecto dinámico y, por tanto, introducen un retraso temporal. Por otro lado, los ruidos también afectan a la medida de la posición del agente seguido. Así, las funciones de transferencia y los ruidos son las causas principales de la distancia entre el RPA seguidor – RPA seguido/obstáculo. El autodirector puede ser electromagnético u óptico en la banda infrarroja.

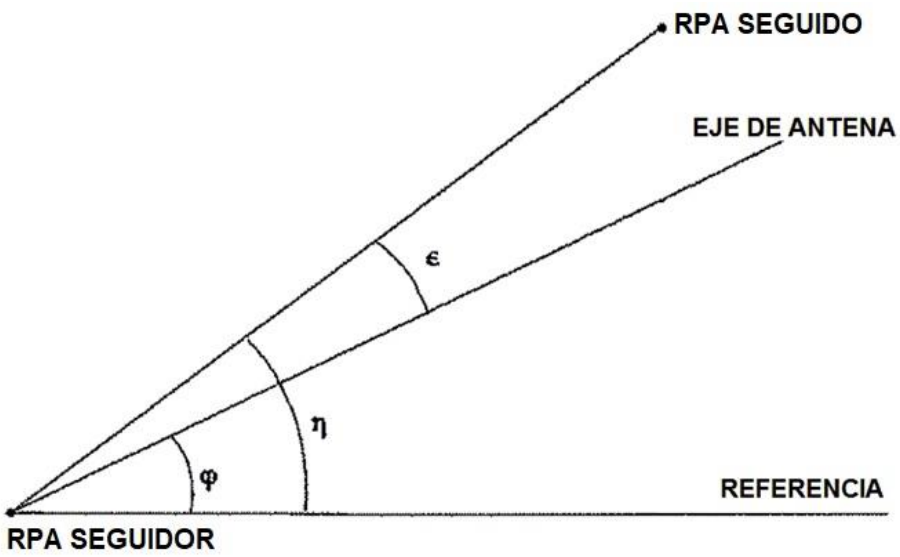

Figura A.3. Detección electromagnética.

Considérese un autodirector electromagnético. En la figura adjunta se muestran los ángulos que intervienen en el proceso de detección. La relación teórica que liga a los ángulos de desviación, ϵ, de la línea RPA seguidor – RPA seguido/obstáculo, η, y del eje de antena, ϕ, es

$$\epsilon = \eta - \varphi \tag{A.4}$$

Sin embargo, la dinámica del detector hace que la relación no sea tan directa, ya que en realidad está afectada de una función de transferencia del tipo

$$\frac{\epsilon}{\eta - \varphi} = \frac{1}{1 + T_{ad}s} \tag{A.5}$$

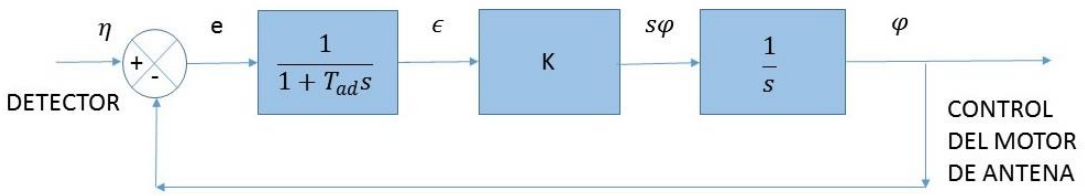

Figura A.4. Sistema autodirector.

La señal ϵ se emplea para el control del movimiento de la antena. La señal del detector, η, es comparada con la señal de control de la antena, ϕ, apareciendo el error, e. Esta es convertida en la desviación ϵ tras ser afectada por la dinámica del sistema. Un bloque de ganancia K transforma la desviación en la

velocidad de rotación del ángulo de antena y una integral proporciona el ángulo, ϕ, del eje de antena. Así, se puede escribir

$$\epsilon = \frac{1}{1 + T_{ad}s}\left[\eta - \frac{K}{s}\epsilon\right] \tag{A.6}$$

Con

$$\frac{\epsilon}{\eta} = \frac{s}{K + s + T_{ad}s^2} \tag{A.7}$$

Y

$$\epsilon = \frac{\frac{1}{K}s\eta}{1 + \frac{1}{K}s + \frac{T_{ad}}{K}s^2} \tag{A.8}$$

La cual es la función de transferencia que proporciona la velocidad de rotación de la línea RPA seguidor – RPA seguido, sη. Aunque, en general, la función de transferencia del autodirector es de segundo orden, como se puede comprobar en la expresión, sin embargo, la ganancia K suele ser grande y la constante de tiempo T_{ad} pequeña, lo que permite escribir de forma simplificada la aproximación

$$\epsilon = \frac{\frac{1}{K}s\eta}{1 + \frac{1}{K}s} \tag{A.9}$$

A continuación se plantea el siguiente problema académico ya que será de utilidad posteriormente, bajo la hipótesis de condiciones iniciales próximas a la colisión sin aceleraciones longitudinales, con lo que la velocidad relativa RPA seguidor – RPA seguido, v_r, es prácticamente constante y el problema se puede linealizar.

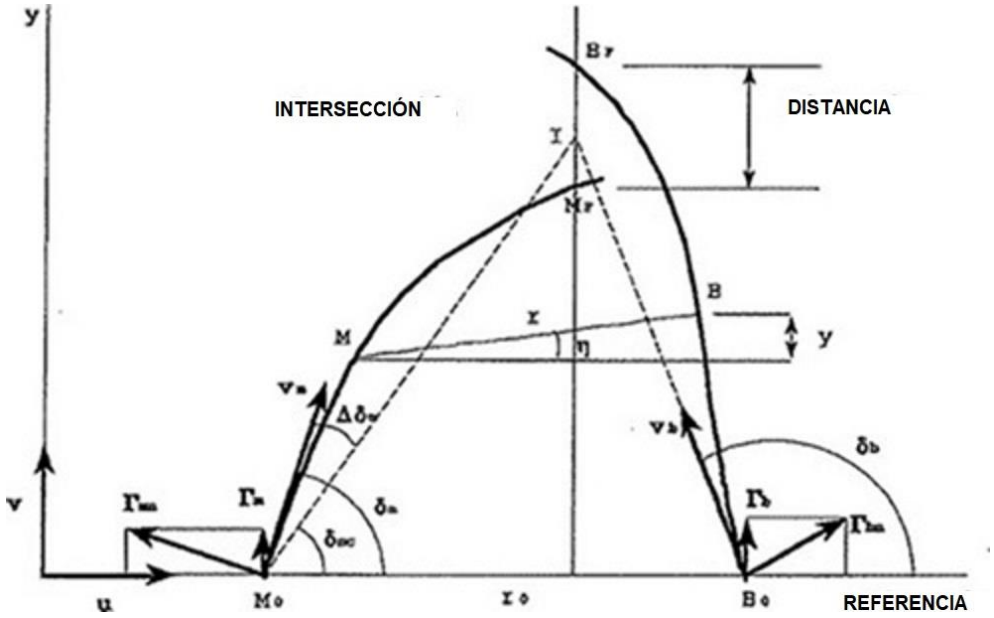

Figura A.5. Autoguiado linealizado.

Tomando como referencia la línea RPA seguidor – RPA seguido inicial, M_0B_0. La distancia a considerar entre ambos RPAs se toma en la dirección del eje y, perpendicular a la línea de referencia. Esta es la distancia transversal, distinta de la distancia geométrica, r. Si se sigue la ley de colisión acabarían en último término encontrándose en el punto de intersección I, verificándose

$$v_T sen\delta_{mc} = V_b sen\varphi_b \tag{A.10}$$

El alejamiento de la ley de colisión se produce, fundamentalmente, por dos causas: un error en la puntería, $\Delta\delta_m$, con relación al ángulo, δ_{mc}, o una maniobra del RPA seguido.

Considerando como tiempo final, o tiempo de vuelo, t_F, el correspondiente a la trayectoria en colisión. Por otra parte, t es el tiempo transcurrido desde el inicio y, dado que v_r, es prácticamente constante, se obtiene la siguiente relación lineal

$$r = v_r(t_F - t) \tag{A.11}$$

η es pequeño por las condiciones impuestas, por lo que

$$\eta = \frac{y}{r} = \frac{y}{v_r(t_F - t)} \tag{A.12}$$

La distancia transversal, y, en el instante final coincide con la distancia geométrica, por lo que es válido el planteamiento del bucle de guiado. Las

aceleraciones de ambos RPAs son las correspondientes a la dirección del eje y (aceleraciones verdaderas). Para el seguidor se tiene

$$\Gamma_m = \Gamma_{mn} cos\delta_m = A v_m cos\delta_m \dot{\eta} = a v_r \dot{\eta} \quad \text{(A.13)}$$

Con Γ_m y Γ_{mn} la aceleración del RPA seguidor y la aceleración normal del RPA seguidor respectivamente.

El error de puntería, $\Delta\delta_m$, produce una velocidad transversal

$$v_m se\square(\delta_{mc} + \Delta\delta_m) \quad \text{(A.14)}$$

Desarrollando el seno y teniendo en cuenta que $\Delta\delta_m$ es pequeño

$$v_m sen\delta_{mc} + v_m cos\delta_{mc}\Delta\delta_m \quad \text{(A.15)}$$

La velocidad transversal relativa es

$$\Delta v = v_b sen\delta_b - v_m sen\delta_{mc} - v_m cos\delta_{mc}\Delta\delta_m \quad \text{(A.16)}$$

Considerando la condición de colisión, queda

$$\Delta v = -v_m cos\delta_{mc}\Delta\delta_m \quad \text{(A.17)}$$

Y cuya derivada equivale a una aceleración añadida a la aceleración del RPA seguido.

El bucle de guiado toma la forma que muestra la figura siguiente. La función de transferencia $1/Z_a(s)$ introduce la dinámica del autodirector y como se ha visto anteriormente suelen ser de segundo o de primer orden. La distancia transversal, y, es dividida por la distancia geométrica, r, para obtener el ángulo de la línea RPA seguidor – RPA seguido. El autodirector y el elaborador de órdenes proporcionan la aceleración comandada, Γ_{co}, y la timonería del bucle de pilotado la traduce en una aceleración ejecutada, Γ_{ej}. Esta es realimentada negativamente a la aceleración del RPA seguido y a la producida por el error de puntería. La aceleración resultante, Γ, es integrada doblemente para obtener la distancia transversal, y.

Puesto que el sistema es lineal, el efecto conjunto de la maniobra del RPA seguido y el error de puntería sobre la distancia transversal es igual a la suma de los efectos separados de cada una de las causas.

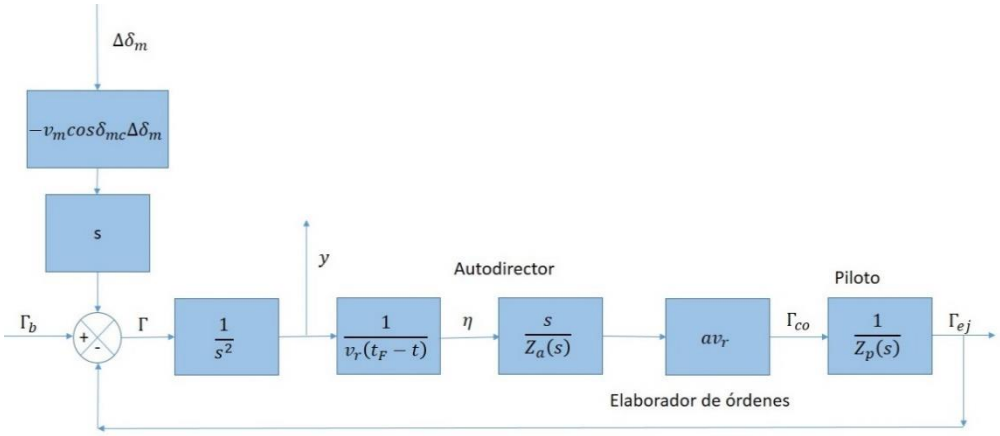

Figura A.6. Bucle de guiado con NP.

Considerando una maniobra del RPA seguido sin error de puntería. Por la linealidad del sistema se puede agrupar las funciones de transferencia del autodirector y del bucle de pilotado en una sola

$$\mathrm{Z}(s) = \mathrm{Z}_a(s)\mathrm{Z}_p(s) \tag{A.18}$$

La figura queda entonces:

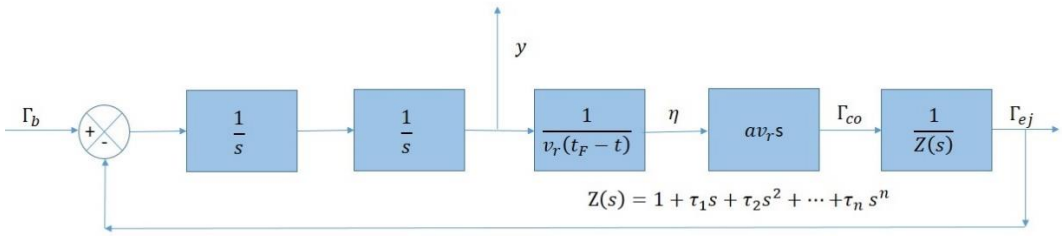

Figura A.7. Navegación proporcional con maniobra del RPA seguido.

La aceleración comandada es

$$\Gamma_{co} = \frac{d}{dt}\left(\frac{ay}{t_F - t}\right) = a\left(\frac{\dot{y}}{t_F - t} + \frac{y}{(t_F - t)^2}\right) \tag{A.19}$$

Puesto que 1/Z(s) es de orden n, este se traduce en un sistema de n ecuaciones diferenciales. Incluyendo las variables x_i y considerando las dos expresiones anteriores, el sistema a integrar es:

$$\begin{aligned} \dot{x}_1 &= \mathrm{x}_2 \\ \dot{x}_2 &= \mathrm{x}_3 \end{aligned} \tag{A.20}$$

$$\dots$$

$$\dot{x}_{n-1} = \mathrm{x}_n$$

$$\dot{x}_n = \left[a\left(\frac{x_{n+2}}{t_F - t} + \frac{x_{n+1}}{(t_F - t)^2}\right) - \mathrm{x}_1 - \tau_1 \mathrm{x}_2 - \dots - \tau_{n-1}\mathrm{x}_n\right]\frac{1}{\tau_n}$$

$$\dot{x}_{n+1} = \mathrm{x}_{n+2}$$

$$\dot{x}_{n+2} = \Gamma_b - \mathrm{x}_1$$

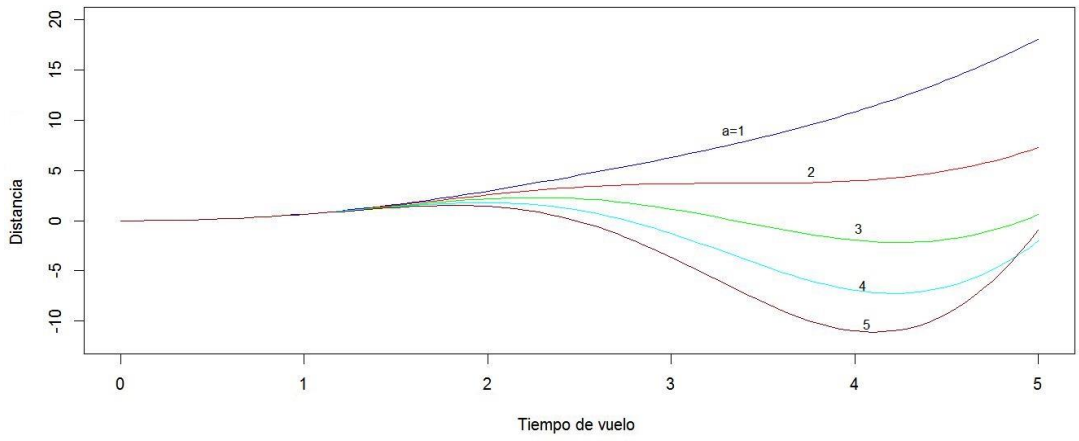

Figura A.8. Distancia entre RPAs. Maniobra RPA seguido en escalón. Bucle de orden 5.

Resulta de interés académico realizar la integración de este sistema para una maniobra del RPA seguido en escalón unitario (1 m/s^2), obteniendo la distancia entre RPAs para un tiempo de vuelo de 5s, bucle de pilotado de orden 5 y diferentes coeficientes de NP. Esto permitirá aplicar el método del sistema adjunto de Laning y Battin a continuación y obtener conclusiones de interés. Obsérvese como la distancia entre RPAs mejora al aumentar el coeficiente de NP. Si se desea construir una gráfica que contenga esta distancia en función del tiempo de vuelo para diferentes tiempos y no solo 5s, puede hacerse construyendo múltiples figuras para varios tiempos de vuelo aunque es un trabajo bastante engorroso por lo que es común recurrir al método del adjunto de Laning y Battin.

Este método del sistema adjunto ya se usaba antiguamente en diferentes problemas, no fue hasta los años 1950 cuando lo popularizaron Laning y Battin. El método se basa en el uso de la respuesta al impulso de los sistemas lineales. El problema del autoguiado se ha linealizado para simplificar los

cálculos, empero se ha demostrado mediante simulaciones por contraste de los resultados lineales con los no lineales, alcanzando a la conclusión de que las diferencias son tan pequeñas que no existe ningún reparo en la utilización del sistema lineal, con las ventajas que ello supone.

Considerando el diagrama de bloques del autoguiado, ya expuesto anteriormente con la siguiente función de transferencia del bucle de pilotado:

$$\frac{\Gamma_{ej}}{\Gamma_{co}} = \frac{1}{Z_p(s)} \tag{A.21}$$

Que en el caso más sencillo es de primer orden.

$$\frac{\Gamma_{ej}}{\Gamma_{co}} = \frac{1}{1 + \tau_1 s} \tag{A.22}$$

La constante de tiempo τ_1 es esencial. Si las constantes de tiempo fueran nulas la respuesta sería inmediata y la distancia en cuestión nula. Sin embargo, la existencia de las constantes de tiempo introduce un retraso en la ejecución de las órdenes y esto termina traduciéndose en una distancia no nula que está en relación con el tiempo de vuelo; es decir, cuanto mayor sea el tiempo de vuelo, en relación con la constante de tiempo, menor será dicha a distancia.

Las constantes de tiempo se obtienen a partir de la mecánica de vuelo mediante funciones no triviales complejas que dependen de los coeficientes aerodinámicos, características másicas, momentos de inercia, empuje, geometría, características de la atmósfera, etc.

Partiendo de un sistema lineal original y construyendo el sistema adjunto, también lineal, como se indica en la figura

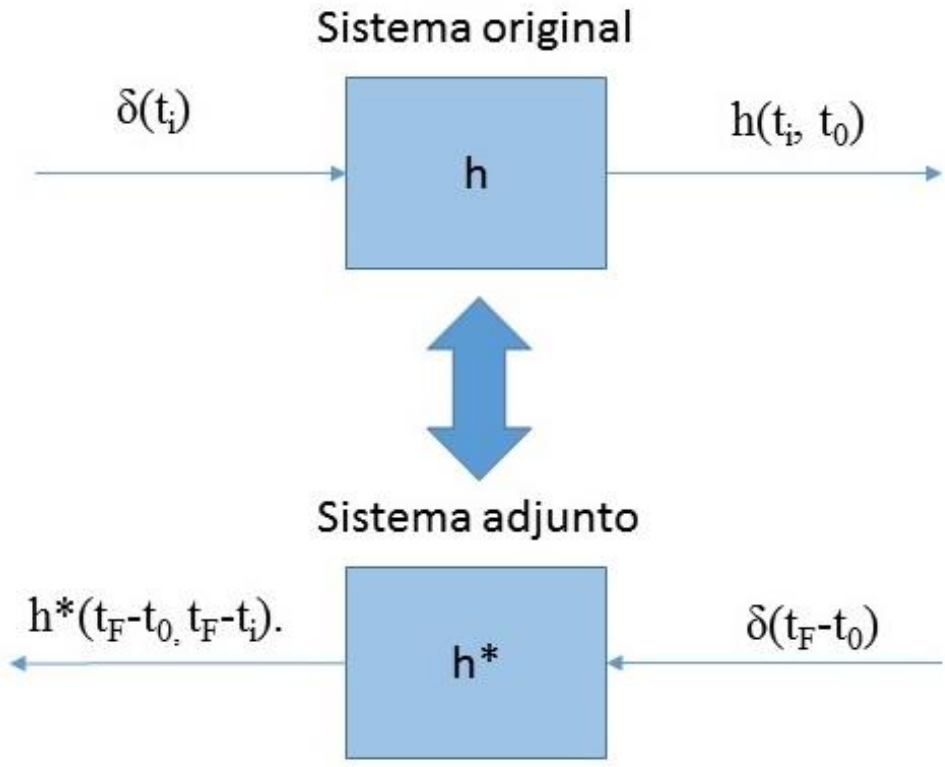

Figura A.9. Sistemas original y adjunto.

El sistema adjunto se forma invirtiendo el flujo de la señal en el sistema original y midiendo los tiempos en referencia a un tiempo t_F, de forma que su tiempo es el complemento al tiempo final, $\theta = t_F - t$. En el sistema original se aplica un impulso, $\delta(t_i)$, en el instante t_i. La respuesta que se observa en el instante t_0 es $h(t_i, t_0)$. En el sistema adjunto el impulso equivalente a su entrada es $\delta(t_F-t_0)$, aplicado en el instante t_F-t_0, y la salida se observa en el instante t_F-t_i, $h^*(t_F-t_0, t_F-t_i)$. Por la reciprocidad de los sistemas lineales se verifica

$$h(t_i, t_0) = h^*(t_F-t_0, t_F-t_i) \qquad \text{(A.23)}$$

Que expresa que la curva $h^*(0, t_F-t_i)$ recoge todos los puntos finales de las respuestas $h(t_i, t_F)$ cuando t_i varía de cero a t_F.

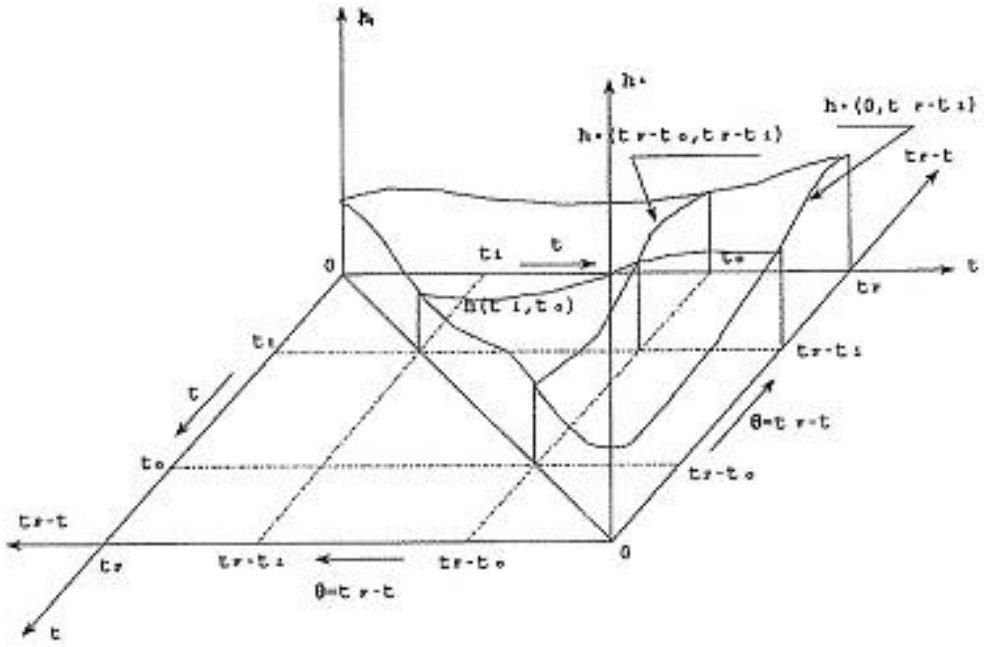

Figura A.10. Respuestas al impulso en los sistemas original y adjunto.

Así, la ventaja del sistema adjunto está en que en una sola curva aparecen todos los puntos finales de una infinidad de curvas del sistema original. A continuación, se verá cómo puede aplicarse el sistema adjunto al caso en que la entrada al sistema original es un escalón, en lugar de un impulso.

Se sabe que, para una entrada cualquiera, x(t), la respuesta del sistema original viene dada por la integral de convolución.

$$y(t) = \int_{-\infty}^{t} x(\tau)\,\mathrm{h}(t-\tau)\mathrm{d}\tau \tag{A.24}$$

En donde t es el instante de observación. Para una entrada escalón de amplitud k la respuesta es

$$y(t) = k\int_{0}^{t} \mathrm{h}(t-\tau)\mathrm{d}\tau \tag{A.25}$$

Puesto que t es el instante de observación y τ el de aplicación del impulso, el valor de la respuesta al impulso, h(t-τ), debe calcularse previamente por múltiples integraciones del sistema original con entrada impulso, una para cada instante diferente de aplicación del impulso. Una vez conocida la función h(t-τ), se efectúa la integración para obtener la respuesta al escalón. Sin embargo, si

se utiliza el sistema adjunto, se puede sustituir en la relación anterior $h(t_i, t_0) = h^*(t_F-t_0, t_F-t_i)$

$$y(t) = k\int_0^t \mathrm{h}^*[(t_F-\tau)-(t_F-t)]\mathrm{d}\tau \tag{A.26}$$

Con el cambio de variable

$$\begin{aligned}\theta &= t_F-\tau\\ d\theta &= -d\tau\end{aligned} \tag{A.27}$$

Obteniendo

$$y(t) = k\int_{t_F-t}^{t_F} \mathrm{h}^*[\theta-(t_F-t)]\mathrm{d}\theta \tag{A.28}$$

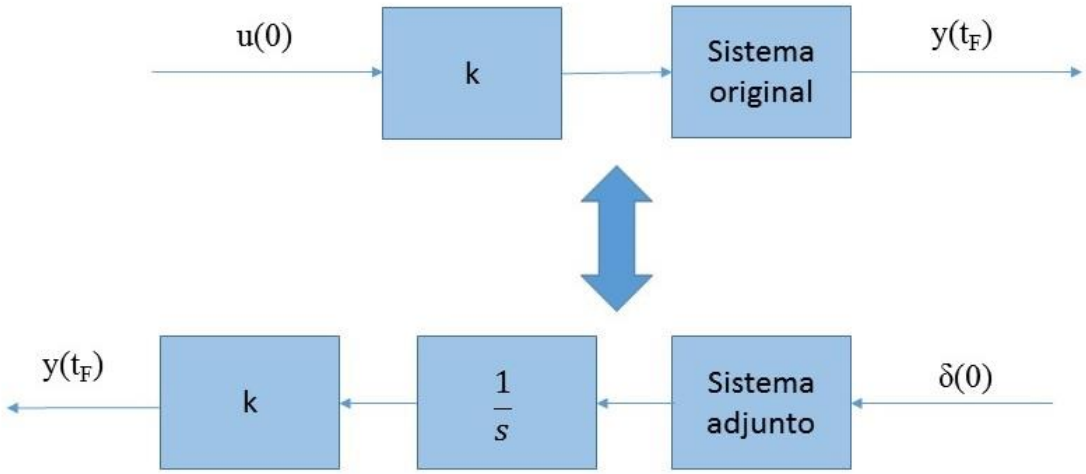

Figura A.11. Equivalencia de los sistemas original y adjunto.

Si se toma como instante de observación el instante final, $t=t_F$, queda

$$y(t) = k\int_0^{t_F} \mathrm{h}^*[\theta]\mathrm{d}\theta \tag{A.29}$$

Como el valor $\theta=t_F-\tau$ en el sistema adjunto constituye el instante de observación, en lugar del instante de aplicación del impulso, la integral anterior quiere decir que la respuesta al escalón del sistema original en el instante t_F, se puede obtener aplicando un impulso en el instante cero a la entrada del sistema adjunto e integrando su salida.

Si hay múltiples entradas escalón en el sistema original, el teorema de superposición de los sistemas lineales dice que la salida $y(t_F)$ es la suma de las aportaciones individuales de cada una de las entradas. Entonces, el sistema adjunto, que también es lineal, proporciona cada una de las contribuciones por separado:

$$y(t_F)=y_1(t_F)+\ldots+y_n(t_F) \qquad \text{(A.30)}$$

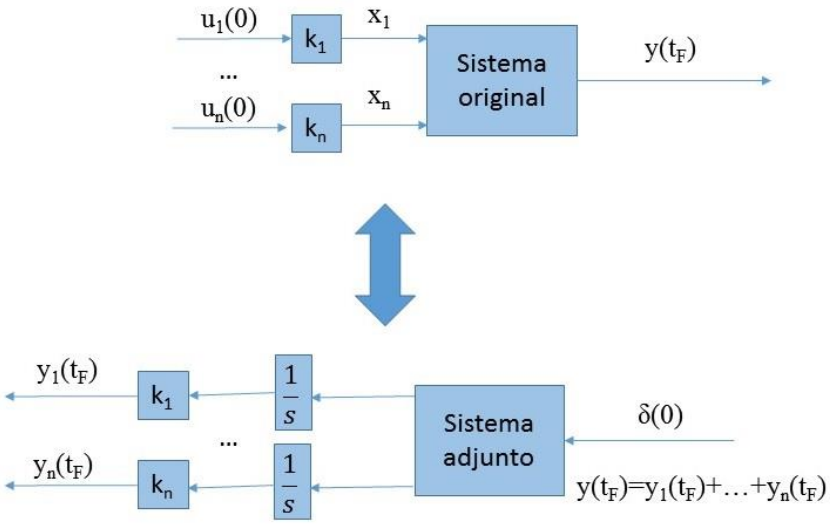

Figura A.12. Respuesta separada para las diferentes entradas.

Para la formación del adjunto correspondiente a un sistema lineal original con entradas en escalón se siguen los siguientes pasos, ilustrados en la figura adjunta:

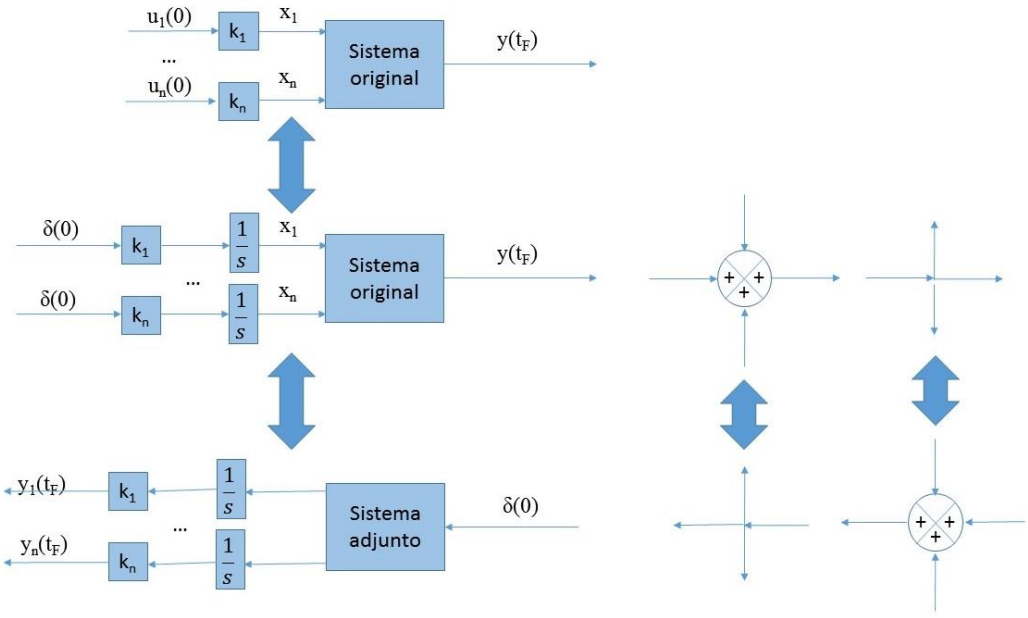

Figura A.13. Formación del sistema adjunto.

1. Se transforman las entradas escalón en entradas impulso usando una integración, dado que la transformada de Laplace de un impulso es un escalón.
2. Se hace el cambio de variable $\theta=t_F-t$, en donde t es el tiempo en el sistema original y θ en el adjunto.
3. Se invierte el sentido del flujo del sistema. Así, la salida única y entradas múltiples del sistema original se transforman en entrada única y salidas múltiples en el sistema adjunto. En cada nudo las flechas entrantes se transforman en salientes y viceversa.
4. Se aplica en la entrada del sistema adjunto un impulso en el instante cero.

Se aplica ahora el método del sistema adjunto para calcular la distancia entre RPAs. Tomando el sistema original de autoguiado con NP.

En el sistema original se han considerado tres causas de perturbación con entrada escalón:

1. Una maniobra del RPA seguido, con amplitud Γ_b.
2. Un error en la puntería inicial, $\Delta\delta mc$.
3. Un escalón en la distancia transversal, Δy_b, producido por un cambio en la velocidad del RPA seguido.

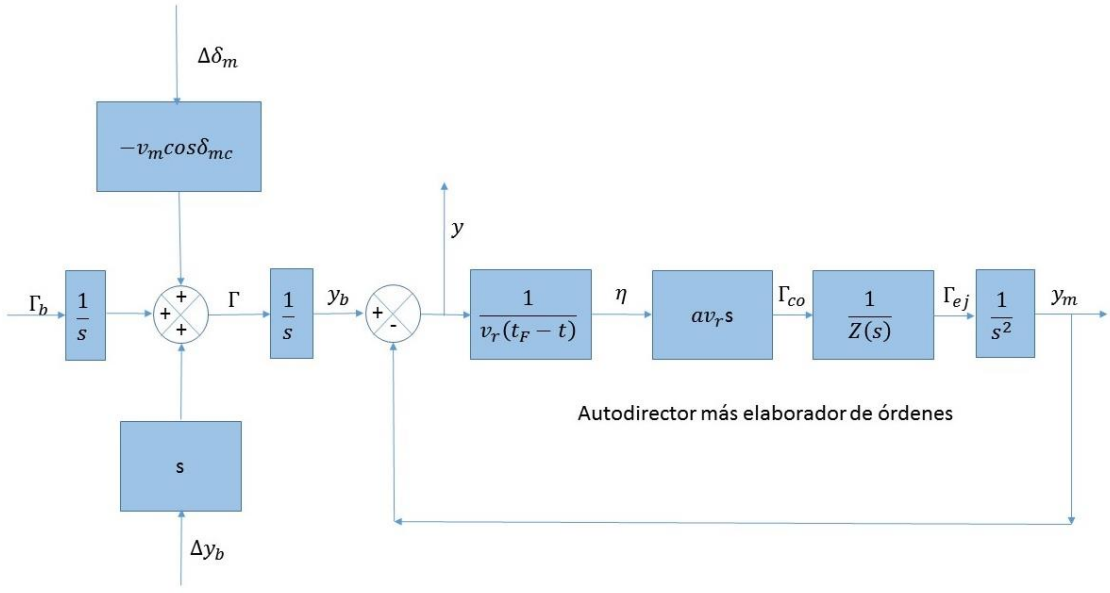

Figura A.14. Sistema original de autoguiado.

Antes de sumar las tres entradas hay que uniformarlas. Para ello se ha integrado una vez la aceleración del RPA seguido y se ha derivado el escalón en distancia. La velocidad resultante de la suma se integra para obtener la distancia transversal y_b, a la que se le resta la distancia transversal corregida por el RPA seguidor, y_m, para obtener la distancia transversal relativa entre ambos RPAs, y. Dividiendo esta distancia por la distancia geométrica entre RPAs se obtiene el ángulo η. El elaborador de órdenes genera la aceleración comandada, Γ_{co}, y el bucle de pilotado produce la aceleración ejecutada, Γ_{ej}, que integrada dos veces proporciona la distancia transversal corregida y_m. Esta se realimenta para cerrar el bucle de guiado.

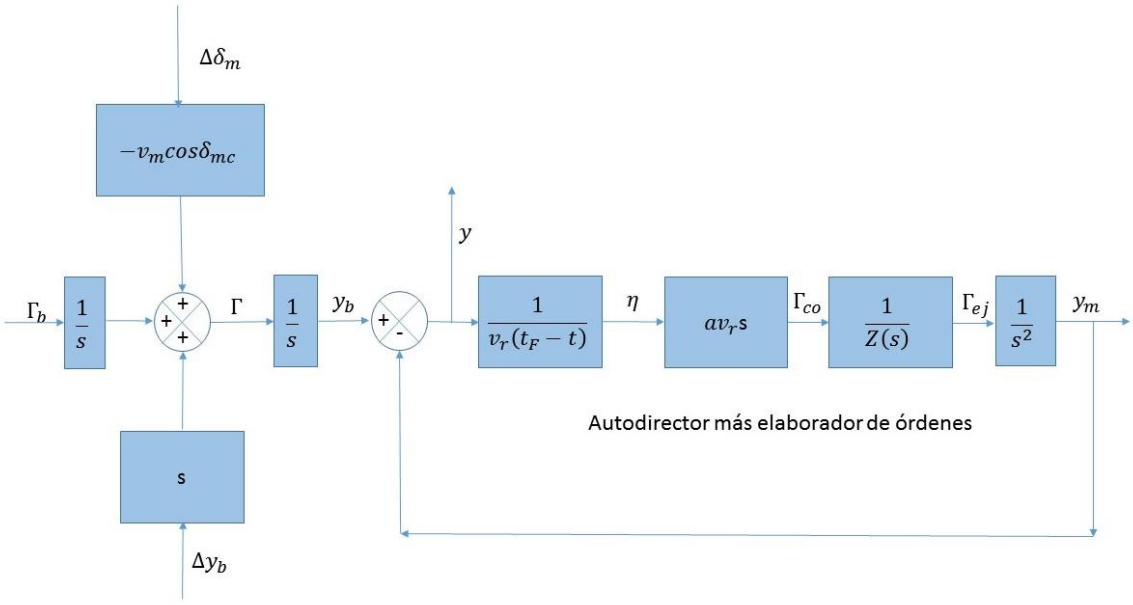

Figura A.15. Sistema original de autoguiado.

El primer paso para construir el adjunto es sustituir las entradas escalón por entradas impulsos. Además se han agrupado los tres últimos bloques en uno solo tomando

$$W(s) = as\frac{1}{Z(s)}\frac{1}{s^2} = \frac{a}{sZ(s)} \tag{A.31}$$

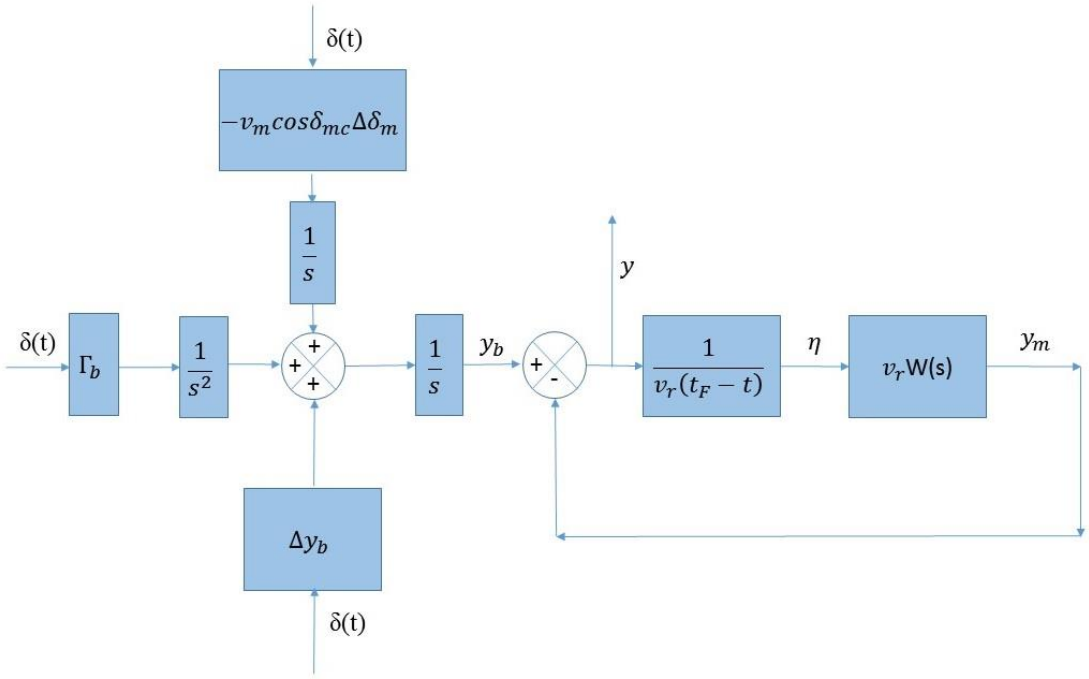

Figura A.16. Sustitución de las entradas escalón por impulsos.

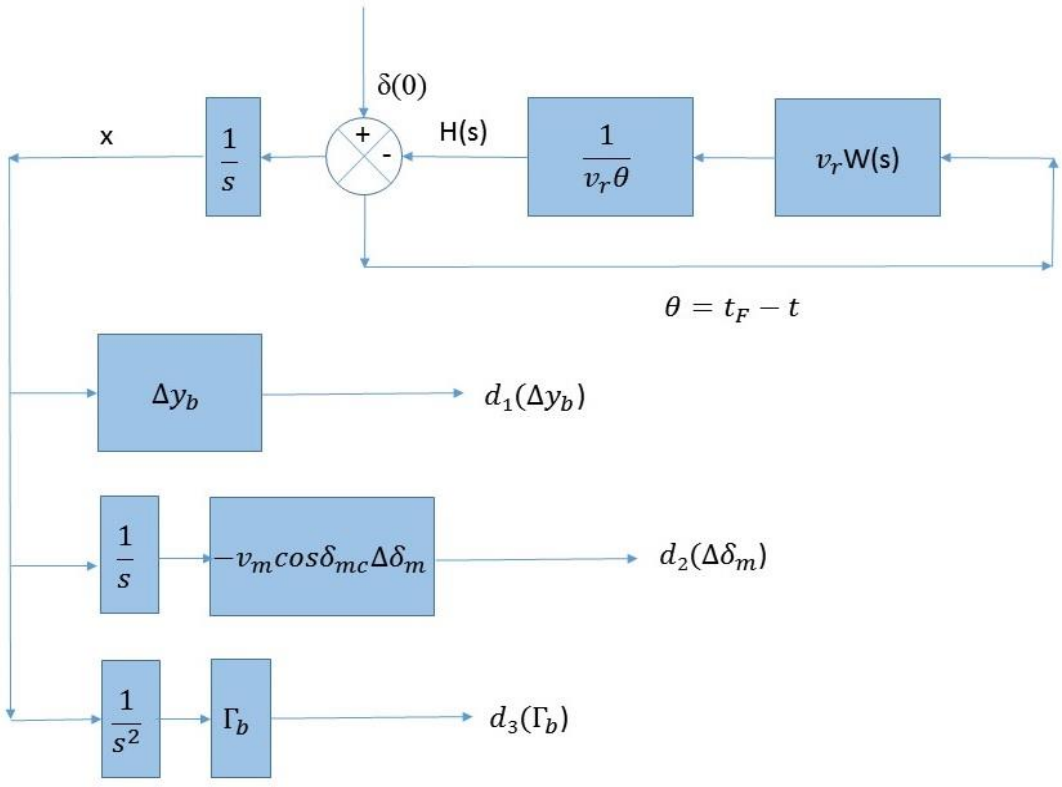

Figura A.17. Sistema adjunto del autoguiado.

Los siguientes pasos realizan el cambio de variable, $\theta=t_F-t$, la inversión del sentido del flujo y la aplicación de un impulso a la entrada del sistema adjunto. Puesto que la salida, y, del sistema original es la distancia entre RPAs en el

instante final, t_F, las salidas en el sistema adjunto representan las distancias entre RPAs individuales con que contribuye cada perturbación inicial. Se puede apreciar en el sistema adjunto que la señal H(s) se realimenta con signo negativo. Esto se debe a que al invertir el flujo la señal proviene del signo negativo de la realimentación y_m, en el sistema original.

Para la integración numérica el impulso inicial se introduce haciendo la señal x=1 en el instante inicial. Esto se debe a que la señal x es inicialmente la integral del impulso unitario de entrada y, por tanto, su valor es un escalón unidad.

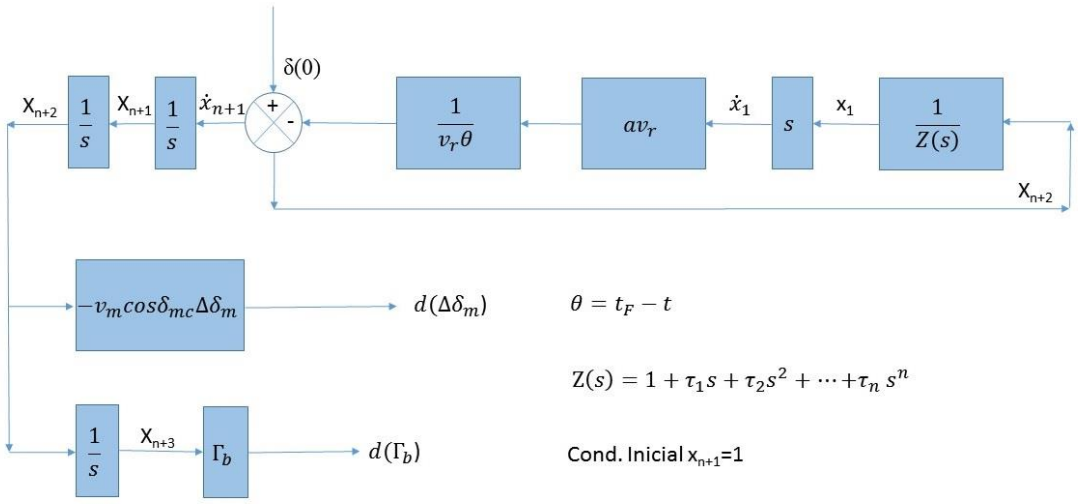

Figura A.18. Adjunto para cálculo numérico.

Para obtener la solución numérica es necesario expresar el sistema adjunto en términos de las ecuaciones diferenciales que lo describen. Para ello, se ha representado el adjunto para cálculo numérico con algunas ligeras modificaciones que ponen de manifiesta las variables que se usaran para la integración. Obsérvese que la velocidad relativa, v_r, no entrará en los cálculos por cancelarse al aparecer recíprocamente en dos bloques. La dinámica de vuelo está expresada por la función de transferencia

$$\frac{1}{Z(s)} = \frac{1}{1 + \tau_1 s + \tau_2 s^2 + \cdots + \tau_n s^n} \tag{A.32}$$

El sistema de ecuaciones diferenciales a integrar se deduce de la figura

$$\dot{x}_1 = \mathrm{x}_2 \tag{A.33}$$

$$\dot{x}_2 = \mathrm{x}_3$$

$$\dots$$

$$\dot{x}_{n-1} = \mathrm{x}_n$$

$$\dot{x}_n = [\mathrm{x}_{n+2} - \mathrm{x}_1 - \tau_1 \mathrm{x}_2 - \cdots - \tau_{n-1}\mathrm{x}_n]\frac{1}{\tau_n}$$

$$\dot{x}_{n+1} = -\frac{a}{\theta}\mathrm{x}_2$$

$$\dot{x}_{n+2} = \mathrm{x}_{n+1}$$

$$\dot{x}_{n+3} = \mathrm{x}_{n+2}$$

El impulso inicial de la entrada se traduce en la condición inicial x_{n+1}=1. Las variables de salida son: x_{n+2} para la distancia por error de puntería, y x_{n+3} para la distancia por maniobra del RPA seguido.

La resolución del problema se ha planteado en función del tiempo de vuelo, θ, pero es más práctico expresarlo en tiempo normalizado. Para la normalización se considera la función de transferencia de la dinámica de guiado (bucle de pilotado más autodirector más filtro de ruido):

$$Z(s) = 1 + \tau_1 s + \tau_2 s^2 + \cdots + \tau_n s^n \qquad \text{(A.34)}$$

El coeficiente más significativo es τ_1, ya que sería la constante de tiempo que correspondería a la aproximación de primer orden.

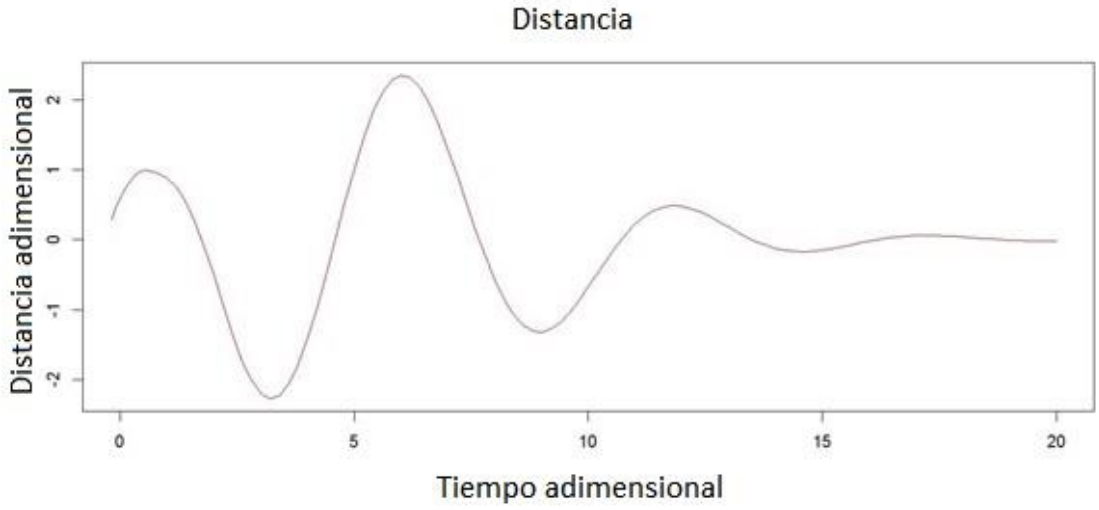

Figura A.19. Bucle de pilotado de orden 5. Distancia por error de puntería.

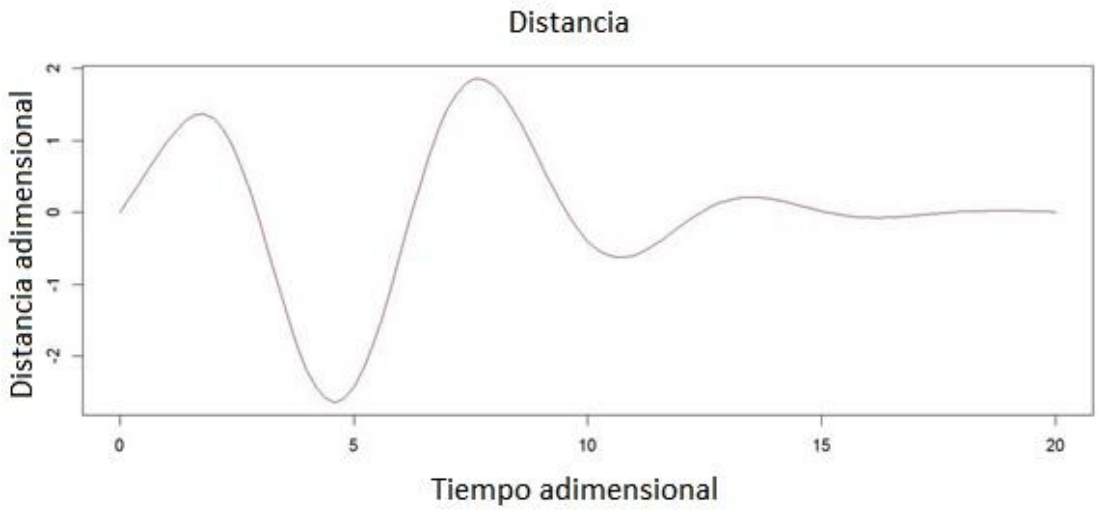

Figura A.20. Bucle pilotado orden 5. Distancia por maniobra RPA seguido con escalón unidad.

Obsérvese que la distancia por error de puntería se encuentra adimensionalizada con $-v_m cos\delta_{mc}\Delta\delta_m \tau_1$ y la distancia por maniobra del RPA seguido con escalón unidad con $\Gamma_b \tau_1^2$, respectivamente, mientras que el tiempo de vuelo se encuentra adimensionalizado con τ_1.

Una vuelta de tuerca adicional es introducir la optimización. Los métodos de optimización están basados, bien en la programación dinámica, bien en el principio del máximo. En general, estos métodos proporcionan condiciones de optimización, pero no una solución completa del problema. Además, en los casos reales conducen a un enorme volumen de cálculos. También requieren el conocimiento de ciertos parámetros, que deben estimarse a través del filtrado de las medidas, ya que la relación señal/ruido es baja.

El problema de optimización se expresará en términos de un vector de estado, x, de dimensión nx1, que está formado por las variables que determinan el sistema. El control se realiza a través de un vector comando, u, de dimensión mx1 constituido por las variables controladas y cuya acción determina el comportamiento del sistema. La variable independiente es el tiempo, t. Las condiciones del problema se traducen en ecuaciones diferenciales que se expresan en la ecuación vectorial de estado:

$$\dot{x} = f(x, u, t) \tag{A.35}$$

Si se linealiza el problema, esta ecuación toma la forma

$$\dot{x} = Ax + Bu \tag{A.36}$$

Donde, A y B son matrices que describen el comportamiento del sistema.

El problema en sí mismo resulta muy complejo y en este apartado solo se darán pequeños esbozos, para mayor información pueden consultarse las referencias.

En el problema del guiado la optimización consiste en minimizar una de las siguientes funciones:

1. Duración de la trayectoria.
2. Distancia entre RPAs.
3. Energía consumida por el RPA seguidor.

La distancia entre RPAs es la que corresponde a tiempo de vuelo establecido, t_F, y está definida por el valor.

$$\lim_{t \to t_F} y = \lim_{t \to t_F} (Y_b - Y_m) \tag{A.37}$$

Donde $y=Y_b-Y_m$ es la situación relativa entre RPAs que está determinada por la posición. El pilotado aerodinámico conlleva una disminución de la velocidad, causada por la resistencia del aire. Optimizar la energía consumida equivale a minimizar la función

$$\int_{t_0}^{t_f} (\Gamma_{mn} cos\delta_m)^2 dt = \int_{t_0}^{t_f} U^2 dt \tag{A.38}$$

En aquellos problemas en los que se considera a constante y se fija el tiempo de vuelo, t_F, el alcance está limitado y la optimización se realizará fijando, o no, el estado final y limitando, o no, el vector de comando, u.

La resolución de los casos propuestos se hará optimizando por el principio del máximo de Pontriaguin, que se expone a continuación sin analizar detalles demostrativos.

Se empieza por expresar el problema a optimizar en forma de ecuación vectorial de estado

$$\dot{x} = f(x, u, t) \tag{A.39}$$

En donde x es el vector de estado y u el vector comando de control. Se supone un problema lineal y, por tanto, la ecuación vectorial de estado toma la forma

$$\dot{x} = Ax + Bu \tag{A.40}$$

Se fijará el criterio a minimizar como un funcional de la forma

$$Q = g(x_F, t_F) + \int_{t_0}^{t_f} \square(x, u, t)dt \tag{A.41}$$

Este funcional consta de dos sumandos, alguno de los cuales puede ser nulo. El primer sumando depende del estado final y aparecerá siempre que éste no esté fijado. El segundo sumando depende del vector de estado, x, y del vector de control, u.

Las condiciones iniciales estarán fijadas. Para las condiciones finales

Se construye el hamiltoniano

$$\mathrm{H} = \Psi^t \mathrm{f}(x, u, t) - r(x, u, t) \tag{A.42}$$

En donde, f es la función vectorial que define la ecuación vectorial de estado, r es la función integrando del criterio a minimizar, y Ψ^t la traspuesta del vector adjunto, cuyas componentes son funciones a determinar durante el proceso de optimización.

El principio del máximo permite determinar el vector de control óptimo en función del vector de estado y del vector adjunto. La condición de óptimo expresa que a un mínimo del funcional Q corresponde un máximo del hamiltoniano, H. El máximo de H se determina de las condiciones

$$\frac{\partial H}{\partial U_i} = 0 \tag{A.43}$$

Donde U_i son las componentes del vector de estado, u, i=1,…,m.

Se escribe el sistema adjunto, formado por las ecuaciones de Pontriaguin, cuya resolución determina al vector adjunto:

$$\frac{\partial \Psi_i}{\partial t} = -\frac{\partial H}{\partial U_i} \tag{A.44}$$

Para i=1,…,n

Se integra, analítica o numéricamente el sistema formado por las ecuaciones de estado, y las ecuaciones adjuntas, obteniendo así una solución con 2n constantes de integración. El conocimiento del estado inicial proporciona n

condiciones, las restantes n condiciones se obtienen bien de las condiciones de contorno, si todas las variables están fijadas, o bien de las condiciones de transversalidad que pueden adoptar las siguientes formas según los casos.

Si el tiempo final, t_F, es libre y el estado final x_F, está fijado, las ecuaciones de transversalidad incluyen la ecuación

$$\widehat{H}_F = \frac{\partial g}{\partial t_F} \tag{A.45}$$

En donde, $\widehat{H}_F$ representa el máximo del hamiltoniano en el instante final.

Si el tiempo final está fijado y el estado final es libre, las condiciones de transversalidad incluyen una ecuación de la forma siguiente para cada variable de estado, x_i, que sea libre:

$$\Psi_{iF} = \frac{\partial g}{\partial x_{iF}} \tag{A.46}$$

Con i=1,…, n y Ψ_{iF} la componente i del vector adjunto en el instante final.

El análisis se efectuará con la hipótesis de condiciones próximas a la colisión, velocidad relativa constante y tiempo de vuelo fijo. Se trata de minimizar la distancia entre RPAs y la energía cuando no hay limitación en el comando de control, U. Es decir, el comando de control puede suministrar toda la intensidad que se le exija.

El criterio a minimizar es

$$Q = \frac{\mu}{2} y_F^2 + \frac{\lambda}{2} \int_{t_0}^{t_f} U^2 dt \tag{A.47}$$

En donde, U es la aceleración normal verdadera del RPA seguidor (comando a optimizar). El primer sumando de Q evalúa la distancia entre RPAs y la integral la energía. Si λ=0 solamente se optimiza la distancia entre RPAs.

El subsiguiente desarrollo se omite por su complejidad y extensión, para mayor información puede consultarse la referencia, resultando finalmente

$$\Gamma_{\square o} = \left(\mathrm{N} v_r \dot{\eta} - \mathrm{k}_m \Gamma_{ej} + \mathrm{k}_b \Gamma_b\right) \tag{A.48}$$

$$N = \frac{6\xi^2\left(\xi - 1 + e^{-\xi}\right)}{\frac{6\lambda}{\mu\tau_1^3} + 2\xi^3 - 6\xi^2 + 6\xi + 3 - 12\xi e^{-\xi} - 3e^{-2\xi}}$$

$$k_m = \frac{1}{\xi^2}\left(\xi - 1 + e^{-\xi}\right)$$

$$k_b = \frac{1}{\xi_b^2}\left(\xi_b - 1 + e^{-\xi_b}\right)$$

$$\xi = \frac{\theta}{\tau_1} = \frac{(t_F - t)}{\tau_1}$$

$$\xi_b = \frac{\theta}{\tau_b} = \frac{(t_F - t)}{\tau_b}$$

Al inicio de la trayectoria θ es grande, por tanto, también es grande ξ y se obtiene:

$$N_0 = \lim_{\xi \to \infty} N = 3$$

$$k_{m0} = \lim_{\xi \to \infty} k_m = 0$$

Al final de la trayectoria θ=ξ=0 y se tiene:

a) Para la optimización de la distancia entre RPAs y la energía, $\lambda \neq 0$:

$$N_F = \lim_{\xi \to 0} N = 0$$

$$k_{mF} = \lim_{\xi \to 0} k_m = \frac{1}{2}$$

b) Para la optimización solo de la energía, λ=0:

$$N_F = \lim_{\xi \to 0} N = \infty$$

$$k_{mF} = \lim_{\xi \to 0} k_m = \frac{1}{2}$$

Si λ≠0, N va de 0 a 3 pasando por un máximo.

Si λ=0, N va de infinito a 3 decreciendo continuamente.

k_m y k_b decrecen continuamente desde 0,5 a 0

A continuación, se va a aplicar el método del sistema adjunto para el estudio de un autoguiado con ley optimal. Considerando la ley optimal calculada anteriormente:

$$\Gamma_{co} = \left(\mathrm{N v}_r \dot{\eta} - \mathrm{k}_m \Gamma_{ej} + \mathrm{k}_b \Gamma_b\right)$$

$$N = \frac{6\xi^2\left(\xi - 1 + e^{-\xi}\right)}{\frac{6\lambda}{\mu\tau_1^3} + 2\xi^3 - 6\xi^2 + 6\xi + 3 - 12\xi e^{-\xi} - 3e^{-2\xi}}$$

$$k_m = \frac{1}{\xi^2}\left(\xi - 1 + e^{-\xi}\right)$$

$$k_b = \frac{1}{\xi_b^2}\left(\xi_b - 1 + e^{-\xi_b}\right) \tag{A.49}$$

$$\xi = \frac{\theta}{\tau_1} = \frac{(t_F - t)}{\tau_1}$$

$$\xi_b = \frac{\theta}{\tau_b} = \frac{(t_F - t)}{\tau_b}$$

Las constantes τ_1 y τ_b se corresponden, respectivamente, a las aproximaciones de primer orden de las funciones de transferencia de RPAs.

Como

$$\eta = \frac{y}{r} = \frac{y}{v_r(t_F - t)} \tag{A.12}$$

Y

$$v_r \dot{\eta} = \frac{\dot{y}}{t_F - t} + \frac{y}{(t_F - t)^2} \tag{A.50}$$

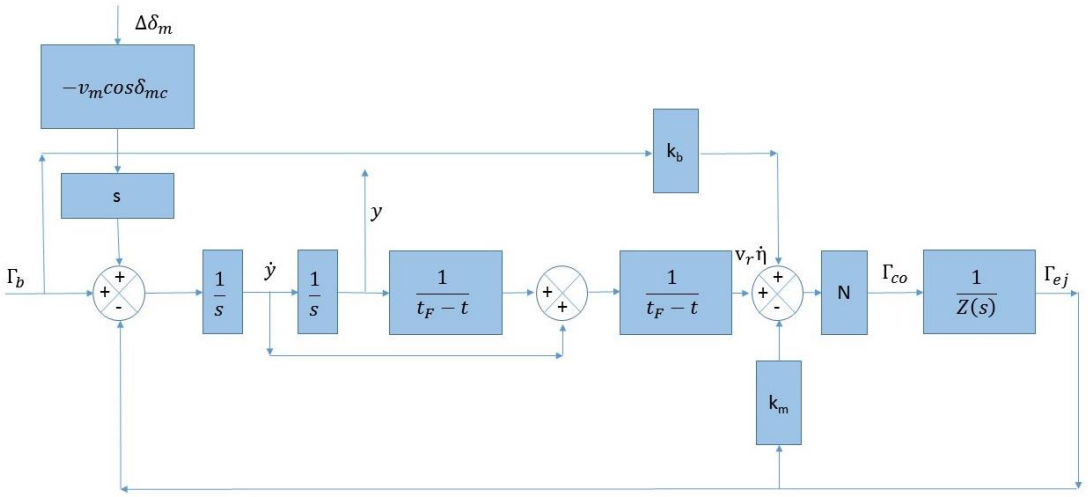

Figura A.21. Guiado con ley optimal.

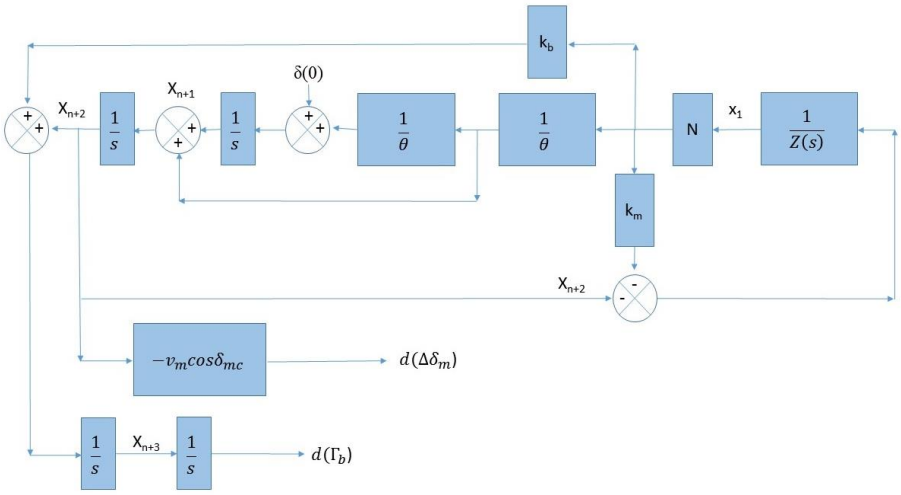

Figura A.22. Adjunto del guiado con ley optimal.

$$\dot{x}_1 = \mathrm{x}_2$$

$$\dot{x}_2 = \mathrm{x}_3$$

$$\dots$$

$$\dot{x}_{n-1} = \mathrm{x}_n$$

$$\dot{x}_n = [-\mathrm{x}_{n+2} - k_m \mathrm{N} \mathrm{x}_1 - \mathrm{x}_1 - \tau_1 \mathrm{x}_2 - \cdots - \tau_{n-1} \mathrm{x}_n] \frac{1}{\tau_n} \qquad \text{(A.51)}$$

$$\dot{x}_{n+1} = \frac{N}{\theta^2} \mathrm{x}_1$$

$$\dot{x}_{n+2} = \mathrm{x}_{n+1} + \frac{N}{\theta} \mathrm{x}_1$$

$$\dot{x}_{n+3} = \mathrm{x}_{n+2} + k_b \mathrm{N} \mathrm{x}_1$$

Las condiciones iniciales son x_i=0 si i≠n+l, x_{n+1}=1.

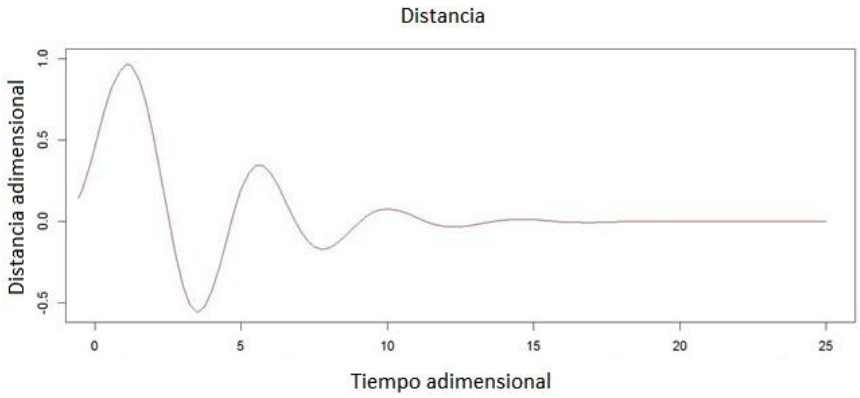

Figura A.23. Autoguiado ley optimal. Bucle pilotado orden 5. Distancia por error de puntería.

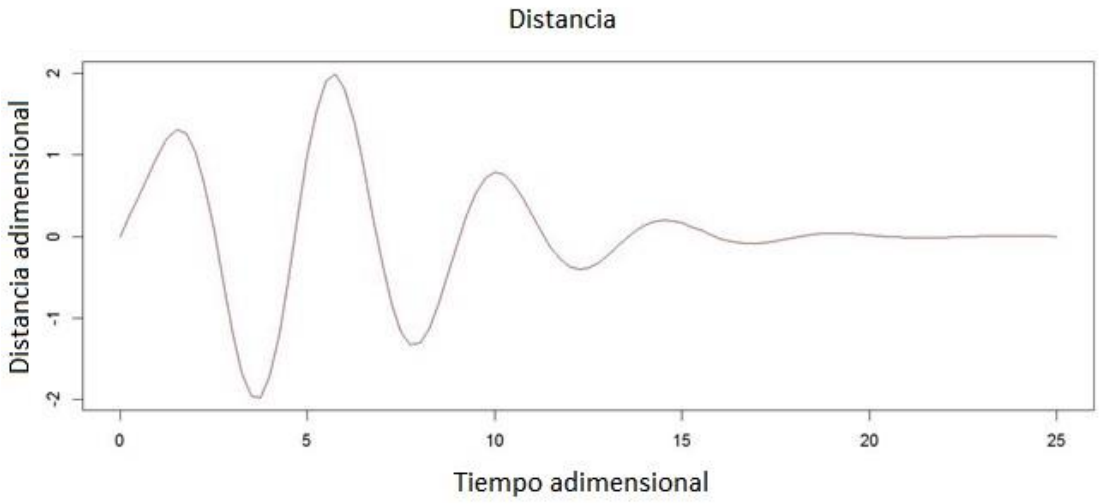

Figura A.24. Autoguiado ley optimal. Distancia de maniobra del RPA seguido con escalón unidad.

Obsérvese que la distancia entre RPAs por error de puntería se encuentra adimensionalizada con $-v_m cos\delta_{mc}\Delta\delta_m$ y la distancia entre RPAs por maniobra del RPA seguido con escalón unidad con Γ_b, respectivamente, mientras que el tiempo de vuelo se encuentra adimensionalizado con τ_1.

Obsérvese que el guiado con ley optimal mostrado aquí se corresponde con el RPA seguidor real con maniobra del RPA seguido, surgiendo un nuevo factor en las ecuaciones que es la constante de tiempo del RPA seguido τ_b, donde se supone como hipótesis que el RPA seguido tiene una función de transferencia de primer orden. Este hecho contrasta con los cálculos efectuados en la determinación de la distancia entre RPAs, convencional, sin guiado con ley optimal, donde la función de transferencia del RPA seguido es la unidad.

Las figuras han sido construidas considerando el RPA seguido con constantes τ_b=2s y τ_1=1s.

A.3. Análisis de resultados

La utilización de la navegación proporcional es una técnica que puede aplicarse de forma satisfactoria a los enjambres de RPAs, especialmente al vuelo en formación como se ha visto a lo largo de este trabajo. Mediante esta técnica se puede evitar la colisión del RPA seguidor tanto con el RPA al que sigue del propio enjambre como de otras aeronaves así como de obstáculos que se encuentren en el espacio aéreo.

Aunque la idea inicial planteada en este trabajo es que cada agente del enjambre reciba información del resto del enjambre conteniendo, entre otro tipo de datos, la posición espacial del resto de RPAs del enjambre, con la misión de que la distancia de seguridad se conserve dentro de los límites indicados anteriormente y se puedan evitar las colisiones entre los RPAs del enjambre, a la par que se el enjambre se encuentre cohesionado. Sin embargo, la navegación proporcional proporciona información añadida de posición de los RPAs del propio enjambre e información de posición de entes foráneos al enjambre como obstáculos y otras aeronaves.

Para la implementación práctica de este trabajo es necesario disponer de funciones de transferencia representativas similares a las reales. Estas se han obtenido aplicando en teoría de control el lugar de las raíces que es aquel lugar geométrico de los polos y ceros de una función de transferencia, cuando la ganancia del sistema se altera. El método del lugar de las raíces calcula la posición de los polos de la función de transferencia en lazo cerrado para un valor determinado de la ganancia partiendo de la función de transferencia a lazo abierto. De esta forma, el lugar de las raíces analiza el efecto de la ganancia en bucle abierto en la respuesta dinámica de un sistema realimentado. Es una herramienta para el análisis dinámico de sistemas realimentados, que muestra características del sistema como estabilidad, rapidez del sistema en cadena cerrada y oscilaciones.

El lugar de las raíces es una herramienta extraordinariamente útil para analizar sistemas dinámicos lineales. Retrotrayendo a la teoría de control se tiene que un sistema es estable si todos sus polos se encuentran en el semiplano izquierdo del plano complejo, para el caso de sistemas continuos. Sin entrar en grandes complejidades que requerirían entrar en conceptos avanzados de la teoría de control, el análisis de un sistema lineal tiene una respuesta que se puede modelar, a grandes rasgos, como un conjunto de exponenciales. Si estas exponenciales tienen unas constantes de tiempo de tal forma que a medida que el tiempo aumenta tiendan a cero, el sistema es estable.

Así, las constantes de tiempo aquí planteadas han sido desarrolladas partiendo de una función de transferencia representativa, de manera que las constantes

de tiempo de dicha función hagan que dicha función muestre un comportamiento estable del sistema.

$$Z(s) = 1 + \tau_1 s + \tau_2 s^2 + \cdots + \tau_n s^n \tag{A.52}$$

Bucle de pilotado de orden 5:

$$Z(s) = 1 + s + 0{,}9s^2 + 0{,}4s^3 + 0{,}1s^4 + 0{,}03s^5 \tag{A.53}$$

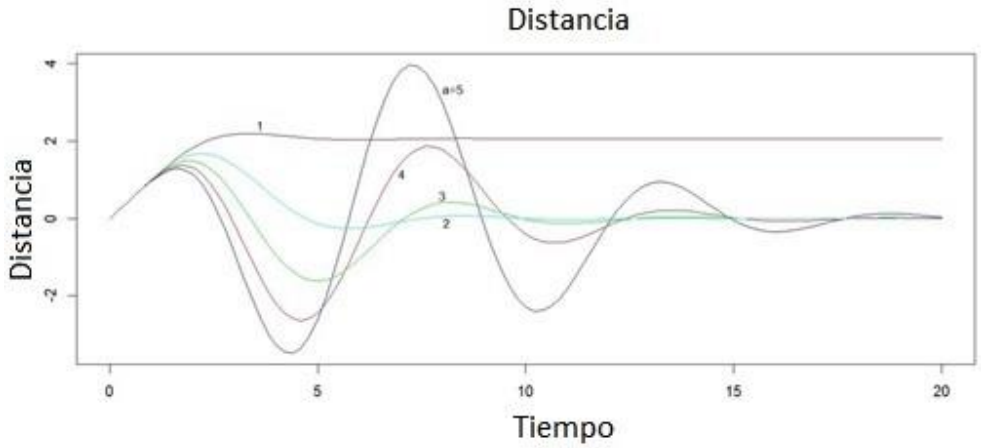

Figura A.25. Bucle pilotado orden 5. Distancia de maniobra del RPA seguido con escalón unidad.

Se puede aproximar el movimiento del RPA seguido como una combinación de escalones unitarios (1 m/s^2) debido a que de esta forma el método del sistema adjunto de Laning y Battin, basado en el uso de la respuesta al impulso de los sistemas lineales, transformando las entradas escalón en entradas impulso mediante una integración. Así, mediante una única curva se obtiene la distancia entre el RPA seguidor y el RPA seguido en función del tiempo.

Para el vuelo en formación de un enjambre se pueden seguir las siguientes reglas adicionalmente a las establecidas por Reynolds sin prejuicio de las teorías de Olfati-Saber.

1. El líder de un enjambre debe tener líderes de reemplazo por si sufriera algún percance o accidente. Existen dos opciones, o todos los RPAs del enjambre son iguales con lo que todo RPA puede ser un líder, o bien la función líder es un atributo propio de una clase diferenciada de RPAs dentro del enjambre, con lo que la transferencia de la función líder solo puede realizarse a otro RPA con capacidad para ello.

2. Este liderazgo se manifiesta como un valor lógico L que adquiere cada RPA de forma que en todo momento se conoce que RPA actúa como líder y cual no.
3. De acuerdo a la investigación del KAIST, para evitar las colisiones es necesario que cada RPA guarde una distancia de seguridad con los otros RPAs, promedio entre 5 y 15 veces el tamaño promedio de un RPA.
4. Así, cada cierto tiempo (paso de tiempo δt), cada RPA comprueba si delante de él hay otros RPAs del enjambre original a una distancia definida d, superior a 15 veces el tamaño promedio de un RPA. Si los hay no es el líder.
5. El concepto de delante del RPA es “delante” del RPA teniendo en cuenta la dirección del vector velocidad del RPA propiamente dicho.
6. Para ello, se traza un semiángulo α/2 cada lado del vector velocidad. Este valor dependerá del problema en cuestión aunque en primera aproximación se puede considerar α=120º.

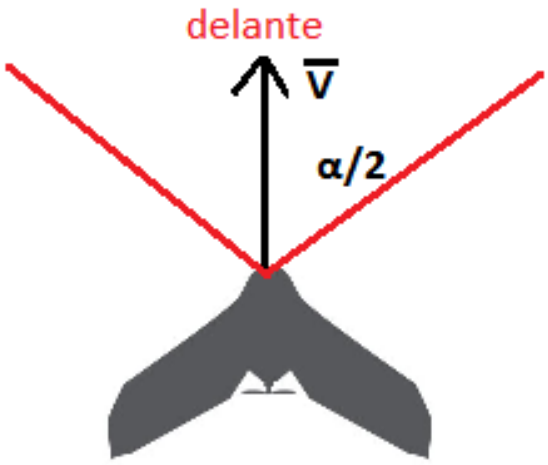

Figura A.26. Determinación de la función líder.

7. Si delante del RPA, a una distancia inferior a d, hay varios RPAs elegirá al próximo al que seguir manteniendo la distancias definidas por el KAIST para no colisionar con él.
8. Si delante del RPA a una distancia definida d no hay líder, este RPA es el líder y cambia su valor lógico L. Así, si todos los RPAs delanteros desaparecen (son derribados), el RPA pasa a ser un líder cambiando su valor lógico.

9. En un enjambre puede haber varios líderes debido a que se han formado varios subenjambres. Ello puede ocurrir si parte del enjambre se separa del enjambre original por la razón que sea y no es capaz de volver a él.
10. Cada cierto tiempo Δt (Δt> δt), el RPA líder busca a otro líder y si el segundo líder está a menos de la distancia d del primero inducirá pequeños movimientos, dentro de la capacidad del RPA; es decir, induciendo la velocidad máxima angular de viraje del RPA y la aceleración del RPA sin sobrepasar la velocidad de crucero (datos técnicos que proporciona la mecánica de vuelo del RPA) con la intención de que el primer enjambre alcance al segundo y se una a él en un solo enjambre.

Apéndice B. Elementos evolutivos computacionales

La genética numérica permite comprobar muchos esquemas a la vez, lo cual permite atacar problemas no lineales cuyo espacio de soluciones potenciales es muy extenso como para hacer una búsqueda profunda en un tiempo razonable. En los problemas lineales, la aptitud de cada componente es independiente del resto, por lo que las mejoras en alguna parte proporcionarán una mejora en el sistema completo. La no linealidad es lo común en la vida real, donde cambiar un componente afecta a todo el sistema, y donde cambios múltiples que aisladamente perjudiciales, todos juntos pueden conducir hacia mejoras.

El cruce es un factor crucial en tecnologías evolutivas. Sin el cruce, cada solución individual explora el espacio de búsqueda sin información completa de lo que el resto de individuos puedan haber encontrado. El cruce transmite información entre los mejores individuos para producir una descendencia que tenga las virtudes de sus progenitores eliminando sus debilidades.

Para representar el problema, lo más común es definir a los individuos como cadenas de números binarios, enteros o reales; donde cada número en cuestión representa algún aspecto determinado de la solución. Si los individuos son cadenas binarias, un 0 o 1 podría significar la falta o la presencia de una cierta característica. La mutación por otra parte implica cambiar estos números, cambiar bits restar o sumar valores aleatorios.

En este caso el concepto de esquema o patrón describe un conjunto cromosomas. Un esquema es una cadena compuesta de ceros, unos y

asteriscos. Cada esquema representa al conjunto de cromosomas que contienen ceros y unos en la misma posición en la que se encuentran en el esquema y cualquier valor en la posición de los asteriscos. Considérense todas las cadenas binarias (ceros y unos) de n dígitos que forman un espacio de búsqueda, representados como ******** (donde * significa 0 o 1). La cadena 01101010 es un miembro de este espacio, pero también es un elemento del espacio 0*******, del espacio 01******, del espacio 0******0, del espacio 0*1*1*1*, del espacio 01*01**0, etc. Evaluando cada cadena concreta, se estaría explorando cada uno de los espacios a los que pertenece. Tras muchas evaluaciones, iría obteniendo una mejora más precisa de la aptitud media de cada uno de estos espacios, cada uno de los cuales contiene muchos elementos. Este concepto se conoce como teorema del esquema en el campo de los algoritmos evolutivos.

Los individuos (potenciales soluciones del problema) pueden representarse como un grupo de parámetros (denominados genes), los cuales en conjunto forman un conjunto de valores (referido como cromosoma). El conjunto completo de material genético se llama genoma. La evolución opera a nivel de cromosomas y no a nivel de individuos. Cada individuo se codifica como una serie de cromosomas por lo que se necesita un método para codificar las posibles soluciones del problema para que un ordenador pueda procesarlos. Un enfoque habitual es codificar los candidatos como cadenas binarias tipo 1s y 0s, donde el dígito de cada posición representa el esquema de algún rasgo de la solución. Otro método diferente pero con analogías consiste en codificar las soluciones del problema como cadenas de números decimales o incluso enteros, donde cada posición representa algún rasgo concreto de la solución. Este sistema es más configurable y preciso que el método binario, y está intuitivamente mucho más cerca del espacio de soluciones del problemas.

En biología, el conjunto de datos representando un cromosoma se denomina fenotipo. El fenotipo contiene la información necesaria para construir un organismo, el cual se identifica como genotipo. La adaptación al problema concreto de un individuo depende de la evaluación del genotipo que puede extraerse a partir del fenotipo.

Otros parámetros deben ser seleccionados como el ritmo de mutación, el cruce y el tipo de selección. Si los ritmos de mutación son muy altos la especie podría extinguirse.

La selección es un proceso que en la fase reproductiva escoge los individuos de la población para cruzarlos y producir descendientes, que una vez mutados, constituirán la siguiente generación de individuos, análogamente a como se hace con el ganado. Una vez seleccionados los progenitores (usualmente dos), sus cromosomas se combinan mediante el cruce y la mutación.

Muchas técnicas pueden utilizarse para seleccionar a los individuos que deben combinarse en la siguiente generación, como puede verse más abajo. Algunos de estos métodos son excluyentes pero otros pueden utilizarse en combinación. Solucionar el problema asociado a la convergencia prematura es clave no proporcionando tanta ventaja a los individuos excesivamente aptos. La selección escalada, por rango o por torneo, son tres de los métodos principales diseñados para conseguir esto.

- Selección elitista: se seleccionan los miembros más adecuados de cada generación (no es común emplear el elitismo puro para intentar evitar problemas derivados de la convergencia prematura). El problema principal es que no permite que el algoritmo abandone el óptimo local para buscar el óptimo global.
- Selección proporcional a la aptitud: los individuos mejores tienen más probabilidad de ser seleccionados.
- Selección por rueda de ruleta: selección proporcional a la destreza de cada individuo en la que la probabilidad de que este sea elegido es proporcional a la diferencia entre su destreza y la del resto de la población. Se imagina como una ruleta donde cada individuo posee una sección de la ruleta pero los más aptos tienen mayores secciones.
- Selección escalada: al aumentar la competencia de la población, la presión selectiva también se incrementa y la función objetivo segrega más. Esta técnica propuesta por Goldberg traza una recta entre el menor valor de aptitud y el mayor valor de la población, y prolonga o trunca los valores de aptitud de cada individuo al valor de la recta que le corresponda. Este método es muy útil cuando todos los individuos

tengan una aptitud relativamente alta con pequeñas diferencias entre ellos.

- Selección por torneo: se seleccionan varios subgrupos de individuos de la población madre aleatoriamente y dentro de cada subgrupo los individuos compiten entre ellos. Se toma un individuo de cada subgrupo para la reproducción.
- Selección por rango: a cada elemento de la población se le concede un valor numérico basado en su destreza y la selección se basa en este valor. Este método evita que individuos muy capaces dominen al resto reduciendo consecuentemente la denominada diversidad genética.
- Selección generacional: la descendencia de los individuos seleccionados en cada generación se convierte en la siguiente generación completa. No se conservan individuos entre generaciones.
- Selección por estado estacionario: la descendencia que se produce en cada generación reingresa de nuevo en la población, reemplazando a ciertos individuos menos aptos.
- Selección jerárquica: los individuos sufren varias rondas de selección en cada generación de tal forma que las evaluaciones iniciales son más rápidas y menos arbitrarias mientras que los que sobreviven más tiempo son evaluados más severamente. Este método reduce el tiempo total de cálculo evaluando someramente a los miembros menos capacitados y sometiendo a una evaluación más rigurosa a los supervivientes que llegan al final del juego.

La dificultad principal en este tipo de problemas es la existencia de gran cantidad de óptimos locales con un óptimo global aislado y poco diferenciado. Es crucial la velocidad de convergencia. Cuando la convergencia es muy rápida (convergencia prematura), el algoritmo converge centrándose en óptimos locales siendo necesario desarrollar estrategias para paliar este inconveniente, como por ejemplo evitar en los procesos de selección quedarse con los individuos más aptos que terminarían dominando a la población.

Para intentar evitar este problema, existen varias soluciones como emplear una función de selección proporcional a la función objetivo, en la cual cada individuo tiene una probabilidad de ser seleccionado como progenitor

proporcional al valor de su función objetivo. Es esencial salir del óptimo local para encontrar otros óptimos que faciliten la localización del óptimo global. Denominando E a la función objetivo y λ el número de individuos que constituyen la población, la probabilidad de que el individuo j sea seleccionado como progenitor es:

$$p_j = \frac{E_j}{\sum_{j=1}^{\lambda} E_j} \tag{B.1}$$

Esta función es invariante ante cambios de escala, pero no ante traslaciones.

Otra forma de superar la rápida convergencia proveniente de superindividuos muy aptos es efectuar la selección proporcional al rango del individuo, produciéndose una repartición más uniforme de la probabilidad de selección. Los individuos se ordenan de menor a mayor asignando al peor individuo el rango uno mientras que el individuo con mejor función objetivo tiene rango el número de individuos de la población, y la probabilidad de que un individuo sea seleccionado como progenitor se define como

$$p_j = \frac{rango(E_j)}{\lambda\,(\lambda+1)/2} \tag{B.2}$$

La suma de los rangos λ(λ+1)⁄2 constituye la constante de normalización. Esta función de selección es invariante frente a translaciones y cambios de escala.

Otra vuelta de tuerca es el modelo de selección del valor esperado, el cual introduce un contador c_j inicializado en $E_j/\hat{E}(t)$ siendo $\hat{E}(t)$ la media de la función objetivo en la generación t. Cada vez que cada individuo es seleccionado para el cruce, dicho contador disminuye una cantidad Δc, con $\Delta c \in [0.1, 0.5]$. El individuo dejará de poder ser seleccionado cuando su contador sea negativo. La dificultad de este método proviene de la elección del valor del contador que además podría variar de unos individuos a otros.

Una vez que el proceso de selección ha encontrado a los miembros más aptos, éstos deberían alterarse aleatoriamente para mejorar sus destrezas en la siguiente generación. Existen dos estrategias básicas para llevar esto a cabo.

La primera es la mutación. En la vida real, una mutación es un suceso bastante poco común (sucede alrededor de una de cada mil replicaciones). En la

mayoría de las situaciones, las mutaciones son letales, pero en promedio, contribuyen a la diversidad genética de la especie. Al igual que una mutación en los seres vivos, se realiza un cambio en uno de los genes de un cromosoma elegido al azar causando pequeñas alteraciones en puntos concretos del código de un individuo. En el caso de una representación binaria, un bit se sustituye por su complemento (un cero cambia en uno y al revés). Este operador permite introducir nuevo material cromosómico en la población tal y como sucede en la biología. La mutación se aplica individualmente a cada miembro de la población alterando al azar (con baja probabilidad) cada gen del cromosoma. La mutación ocasiona que los cromosomas de los hijos puedan ser diferentes a los de los progenitores.

La mutación es un mecanismo que genera diversidad, y por tanto proporciona una posible salida cuando el algoritmo genético está estancado, pero no se debe abusar de él.

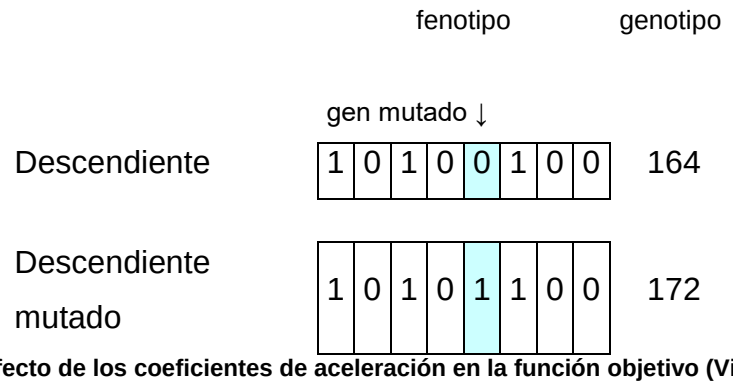

Figura B.1. Efecto de los coeficientes de aceleración en la función objetivo (Vishal A. Rane).

Análogamente al cruce, la mutación se opera como un porcentaje que indica con qué frecuencia se producirá, aunque a diferencia de la anterior sucede de forma más esporádica (el porcentaje de cruce es habitualmente de más del 60% mientras que el de mutación raramente suele superar el 5%).

Diferentes técnicas de mutación pueden considerarse dependiendo de si la mutación afecta a un bit, un grupo de bits, a un gen o a un grupo de genes:

- Mutación de bit: existe una probabilidad única de que se produzca una mutación de un bit concreto. Si se produce, el algoritmo toma aleatoriamente un bit y lo invierte.
- Mutación multibit: cada bit tiene una probabilidad de mutarse que se calcula por el operador de mutación multibit.

- Mutación de gen: equivalente a la mutación de bit, solo que en vez de cambiar un bit, cambia un gen completo. Puede sumar un valor constante, un elemento aleatorio o bien introducir un gen al azar.
- Mutación multigen: semejante a la mutación de multibit, solo que en vez de cambiar un conjunto de bits, cambia un conjunto de genes. Puede sumar un valor constante, un elemento aleatorio o bien introducir un gen al azar.
- Mutación de intercambio: existe una probabilidad de que se produzca una mutación. Si se produce, toma dos bits/genes aleatoriamente y los intercambia.
- Mutación de barajado: existe una probabilidad de que se produzca una mutación. Si se produce, toma dos bits o dos genes aleatoriamente y los baraja de forma aleatoria.

El segundo método de variación aleatoria se llama cruce e involucra elegir a dos individuos para que intercambien fragmentos de su código, produciendo una descendencia cuyos individuos son combinación de sus progenitores. Este proceso simula la recombinación que se da en los cromosomas durante la reproducción sexual. Las formas habituales de cruzamiento involucran el cruzamiento de un punto, en el que se establece un punto de intercambio en un punto aleatorio del genoma de los dos individuos y uno de los individuos contribuye con todo su código anterior a ese punto y el otro individuo contribuye con su código a partir de ese punto para generar una descendencia. En el cruzamiento uniforme, el valor de una posición dada en el genoma de descendencia corresponde al valor en esa posición del genoma de uno de los progenitores o al valor en esa posición del genoma del otro progenitor, elegidos con una cierta probabilidad.

Si se emparejan dos descendientes de los mismos progenitores, ello garantiza la perpetuación de un individuo con buenas características (es una práctica utilizada en la cría de animales y destinada a potenciar unas ciertas características frente a otras). No obstante, si esto sucede demasiado frecuentemente puede crear problemas, toda la población puede ser dominada por los descendientes de algún gen que además puede tener caracteres no deseados. Esto sería un estancamiento en un mínimo local.

Considerando el cruce en un punto, se escogen dos progenitores y se extraen sendas ristras de cromosomas en una posición elegida al azar para generar dos subristras iniciales y dos subristras finales. Posteriormente se intercambian las subristras finales generando dos nuevos cromosomas completos. Ambos descendientes heredan genes de ambos progenitores.

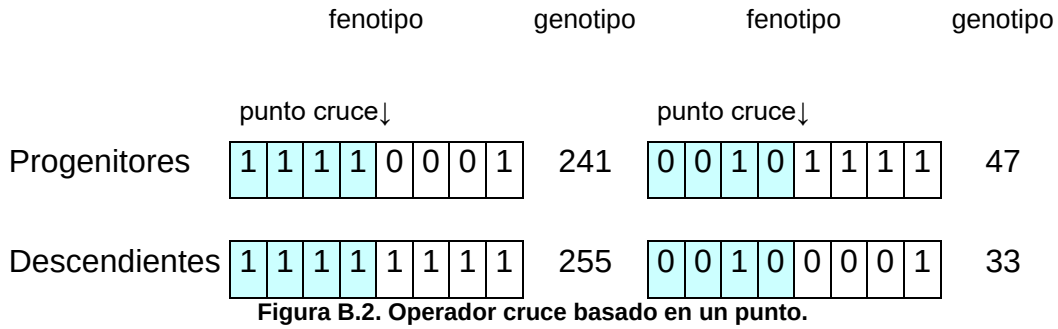

Figura B.2. Operador cruce basado en un punto.

El ejemplo anterior puede ser visto desde el punto de vista del genotipo, un descendiente dados dos progenitores:

$$255=Rnd1\cdot 241+Rnd2\cdot 47 \qquad (B.3)$$

Siendo Rnd1 y Rnd2 números generados aleatoriamente comprendidos entre 0 y 1. Obviamente es posible encontrar dos valores Rnd1 y Rnd2 que cumplan la relación anterior. Más exclusivo sería

$$255=Rnd\cdot 241+(1-Rnd)\cdot 47 \qquad (B.4)$$

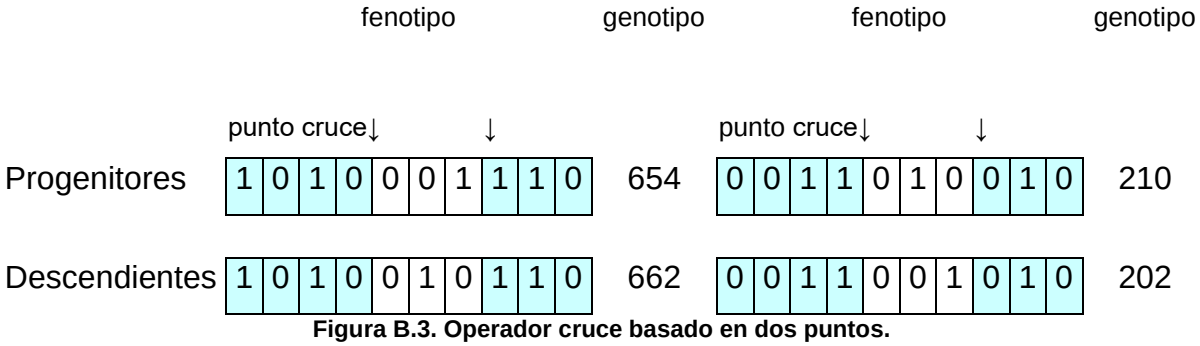

Figura B.3. Operador cruce basado en dos puntos.

Otros operadores de cruce han sido investigados teniendo en cuenta más de un punto de cruce. De Jong centró sus investigaciones en el cruce basado en múltiples puntos concluyendo que el cruce en dos puntos mejora frente a añadir más puntos de cruce. La ventaja de tener más de un punto de cruce reside en que el espacio de búsqueda puede ser explorado más fácilmente aunque también se pueden romper buenas estructuras.

Desde el punto de vista del genotipo; se tiene un descendiente dados dos progenitores:

$$662=Rnd\cdot 654+(1-Rnd)\cdot 210 \qquad (B.5)$$

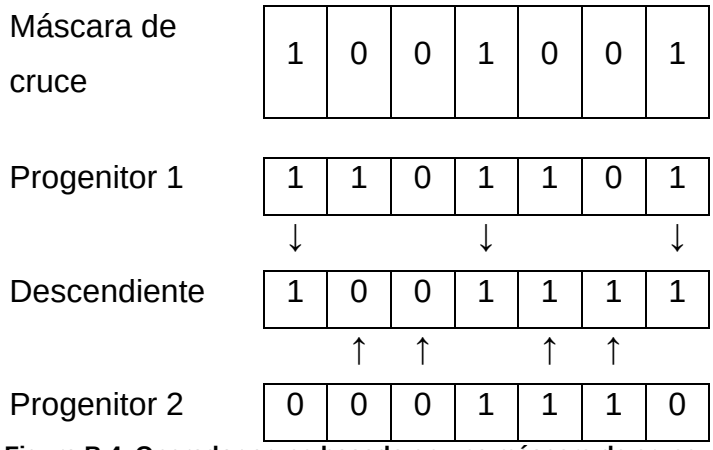

Figura B.4. Operador cruce basado en una máscara de cruce.

El operador de cruce uniforme de Syswerda se configura de forma que cada gen, en la descendencia se crea copiando el correspondiente gen de uno de los dos progenitores, seleccionado de acuerdo a una máscara de cruce aleatoria. Cuando se tiene un 1 en la máscara de cruce, el gen es copiado del primer progenitor, mientras que cuando exista un 0, el gen se extrae del segundo progenitor.

Existen diversas variantes de máscara de cruce aplicando el concepto de probabilidad de herencia. Si esta probabilidad es alta se heredan mayores características de los progenitores. Un avance ocurre cuando esta probabilidad depende de una gran variedad de parámetros.

Aunque podría pensarse que el operador cruce es más importante que el operador mutación, éste último se considera un operador cardinal que proporciona un elemento de aleatoriedad en el entorno de los individuos de la población. Es reconocido que el operador de cruce es el responsable de efectuar la búsqueda a lo largo del espacio de soluciones; empero, varios investigadores piensan que el operador mutación es más importante a medida que la población va convergiendo. Mediante la evolución primitiva, el proceso evolutivo consta tan sólo de selección y mutación, donde Schaffer encontró que dicha evolución primitiva supera ampliamente a una evolución basada exclusivamente en selección y cruce.

La definición de convergencia introducida por De Jong resulta muy interesante. Así, la población evolucionará a lo largo de generaciones de tal forma que la adaptación media de todos los individuos de la población, así como la adaptación del mejor individuo se traducirán en un progreso hacia el óptimo global. El concepto de convergencia está relacionado con el avance hacia la uniformidad, un gen converge cuando un alto porcentaje de individuos de la población comparten el mismo valor de dicho gen. Se dice que la población converge cuando todos los genes han convergido.

Generalmente el cruce se emplea dentro de la implementación del algoritmo genético como un cierto porcentaje que indica con qué frecuencia se efectuará. Esto significa que no todas las parejas de cromosomas se cruzarán, sino que habrá algunas que permanezcan intactas en la siguiente generación. De hecho, existe una corriente denominada elitismo, en la que el individuo más apto a lo largo de las distintas generaciones no se cruza con nadie y se mantiene indemne hasta que surge otro individuo mejor que él, que lo desalojará.

Una vez conseguidos los individuos descendientes de una determinada población, el proceso de reducción al tamaño original consiste en elegir la nueva generación de población entre los progenitores y los descendientes de los mismos.

Existen dos planteamientos esenciales a este respecto. En el primero, los individuos generados reemplazan a sus progenitores con lo que un individuo no convive nunca con sus antecesores (reducción simple). En el segundo bosquejo se sustituyen los individuos peor adaptados de toda la población (política elitista) con lo que existirá convivencia entre un individuo y sus ancestros. Sin embargo, se han bosquejado otras consideraciones substitutas a fin de evitar la aparición de elitismo como suplantar una tasa de la población.

El concepto de reducción está relacionado con el de tasa de reemplazamiento generacional; o sea, el porcentaje de descendientes generados con respecto del tamaño de la población.

Apéndice C. Tutorial SVM

Las SVM parten de una función Φ implícita que transforma los datos de entrada en un espacio de características de alta dimensionalidad definido por alguna función tipo kernel, como por ejemplo una función que devuelve el producto interno ⟨Φ(x),Φ(x´)⟩ relativa a la operación entre las imágenes de dos puntos de datos x, x´ en el espacio de características donde se realiza el aprendizaje. Mediante la aplicación de transformación no lineal Φ:X→H, el producto escalar ⟨Φ(x),Φ(x´)⟩ se puede representar mediante una función kernel k:

$$\mathrm{k}(\mathrm{x}, \mathrm{x}') = \langle \Phi(\mathrm{x}), \Phi(\mathrm{x}') \rangle \qquad \text{(C.1)}$$

Una propiedad interesante de las máquinas de vectores soporte o cualquier otro sistema basado en kernels es que, una vez que se ha seleccionado una función kernel válida, se puede trabajar en espacios de cualquier dimensión con poco costo computacional.

Otra ventaja añadida de las SVMs y los métodos kernel es que se puede plantear un kernel para un problema concreto que podría aplicarse directamente a los datos sin la necesidad de un proceso de extracción. Esto es sustancial en los problemas donde el proceso de extracción pierde mucha estructura de datos.

En clasificación, las SVMs separan las diferentes clases de datos mediante un hiperplano

$$\langle \mathrm{w}, \Phi(\mathrm{x}) \rangle + \mathrm{b} = 0 \qquad \text{(C.2)}$$

Correspondiente a la función de decisión

$$f(x) = signo(\langle \mathrm{w}, \Phi(\mathrm{x}) \rangle + b) \qquad \text{(C.3)}$$

Se puede demostrar matemáticamente que el óptimo es el que tiene el margen máximo de separación entre las dos clases (Vapnik 1998), cuya solución se obtiene resolviendo un problema de optimización cuadrático con restricciones y toma la forma $w = \sum_i \alpha_i \Phi(\mathrm{x}_i)$ función de un subconjunto de patrones de entrenamiento que se sitúan en el propio margen. Estos patrones de entrenamiento, denominados vectores soporte, contienen toda la información esencial del problema de clasificación.

En el caso de clasificación con márgenes débiles, el problema de optimización toma la forma:

$$\begin{gathered} minimizar\ t(w,\xi) = \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i \\ y_i(\langle \Phi(x_i), w\rangle + b) \geq 1 - \xi_i\ (i = 1 \ldots m) \\ \xi_i \geq 0\ (i = 1 \ldots m) \end{gathered} \tag{C.4}$$

Con $\|\ \|$ el operador norma L_2, m el número de patrones de entrenamiento y $\xi_i \geq 0$ y $y_i = \pm 1$. Generalmente, se puede demostrar que la solución SVM w tiene una solución del tipo

$$w = \sum_{i=1}^{m} \alpha_i \mathrm{y}_i \Phi(\mathrm{x}_i) \tag{C.5}$$

Donde los coeficientes de vectores soporte son no nulos cuando un punto (x_i, y_i) se posicionan en la restricción. Por otra parte, los coeficientes α_i se encuentran resolviendo el siguiente problema de programación cuadrática dual:

$$\begin{gathered} maximizar\ W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j\, k(x_i, x_j) \\ 0 \leq \alpha_i \leq \frac{C}{m}\ (i = 1 \ldots m) \\ \sum_{i,j=1}^{m}\alpha_i y_i = 0 \end{gathered} \tag{C.6}$$

Que es un problema cuadrático típico de la forma:

$$minimizar\ c^T x + \frac{1}{2}x^T H \tag{C.7}$$

$$b \leq Ax \leq b + r$$

$$l \leq x \leq u$$

Con $H \in \mathbb{R}^{mxn}$, $H_{ij}=y_i y_j\ k(xi,xj)$, $c=(1,\ldots,1) \in \mathbb{R}^m$, $u=(C,\ldots,C) \in \mathbb{R}^m$, $l=(0,\ldots,0) \in \mathbb{R}^m$, $A=(y_1,\ldots,y_m) \in \mathbb{R}^m$, b=0, r=0. Este problema puede ser resuelto con técnicas clásicas de programación cuadrática. El parámetro de costo c controla lo que paga la SVM al clasificar erróneamente un punto de entrenamiento y evidentemente la función de predicción. Un valor c alto forzará a la SVM a crear una función de predicción lo suficientemente compleja como para clasificar lo menos posible los puntos de entrenamiento, mientras que un parámetro c menor conducirá a una función de predicción más simple. Así, esta SVM se llama C-SVM. Otra formulación análoga de clasificación es emplear un parámetro similar al C, la v-SVM donde el parámetro v es un límite superior en el error de entrenamiento y un límite inferior para los vectores soporte controlando la complejidad de la función de clasificación.

En el caso de los problemas de detección (o clasificación de una clase), la SVM detecta valores atípicos en un conjunto de datos creando un límite de decisión esférico alrededor de un conjunto de puntos datos haciendo uso de un conjunto de vectores soporte que determinan el límite de la esfera.

$$\begin{gathered} minimizar\ t(w,\xi,\rho) = \frac{1}{2}\|w\|^2 - \rho + \frac{1}{mv}\sum_{i=1}^{m}\xi_i \\ \langle \Phi(x_i), w\rangle + b \geq \rho - \xi_i\ (i = 1 \ldots m) \\ \xi_i \geq 0\ (i = 1 \ldots m) \end{gathered} \tag{C.8}$$

El parámetro v se usa para controlar el volumen de la esfera y, en consecuencia, el número de valores atípicos encontrados. El valor de v establece un límite superior en la fracción de valores atípicos encontrados en los datos.

Las SVMs se pueden adaptar para resolver problemas de regresión que suelen llamarse SVR (Support Vector Regression). A partir de unos datos de entrenamiento, se asume que es posible cuasi-ajustarlos mediante una función encontrando unos parámetros w que permitan definir dicha función

$$f(x) = \langle \mathrm{w}, \Phi(\mathrm{x}) \rangle + \mathrm{b} = 0 \tag{C.9}$$

En el caso de regresión se define una función de pérdida de tal modo que esta función ignora los errores menores de un umbral Є>0 permitiendo cierta dispersión en la solución. Dado que es muy difícil que los datos de entrenamiento se ajusten al modelo con un error nulo, se emplea un modelo blando mediante dos variables de holgura, ξ_i^+ y ξ_i^-, que permitirán estimar de dicho error.

$$minimizar\ t(w, \xi) = \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m}(\xi_i^+ + \xi_i^-)$$

$$(\langle \Phi(x_i), w \rangle + b) - y_i - \epsilon - \xi_i^+ \leq 0 \tag{C.10}$$

$$y_i - (\langle \Phi(x_i), w \rangle + b) - \epsilon - \xi_i^- \leq 0$$

$$\xi_i^+, \xi_i^- \geq 0\ (i = 1 \ldots m)$$

Las funciones del kernel son funciones de producto interno entre dos puntos en un espacio de características, con un costo computacional inferior en espacios de dimensiones muy elevadas. Los kernels más comunes son

- kernel lineal

$$\mathrm{k}(\mathrm{x}, \mathrm{x}') = \langle \mathrm{x}, \mathrm{x}' \rangle \tag{C.11}$$

- función de base radial Gaussiana (RBF, Gaussian Radial Basis Function)

$$\mathrm{k}(\mathrm{x}, \mathrm{x}') = \exp(-\sigma\|\mathrm{x} - \mathrm{x}'\|^2)\ \sigma > 0 \tag{C.12}$$

- kernel polinomial

$$\mathrm{k}(\mathrm{x}, \mathrm{x}') = (\mathrm{escala} \cdot \langle \mathrm{x}, \mathrm{x}' \rangle + \mathrm{offset})^n \tag{C.13}$$

- kernel tangente hiperbólica o sigmoidal

$$\mathrm{k}(\mathrm{x}, \mathrm{x}') = \tanh(\mathrm{escala} \cdot \langle \mathrm{x}, \mathrm{x}' \rangle + \mathrm{offset}) \tag{C.14}$$

- función de Bessel de primera especie

$$\mathrm{k}(\mathrm{x}, \mathrm{x}') = \frac{Bessel_{(\nu+1)}^{n}(\sigma\|\mathrm{x} - \mathrm{x}'\|)}{(\|\mathrm{x} - \mathrm{x}'\|)^{-n(\nu+1)}} \tag{C.15}$$

- función de base radial de Laplace

$$\mathrm{k}(\mathrm{x}, \mathrm{x}') = \exp(-\sigma\|\mathrm{x} - \mathrm{x}'\|) \tag{C.16}$$

- kernel de base radial ANOVA

$$\mathrm{k}(\mathrm{x},\mathrm{x}') = \left(\sum_{k=1}^{n} \exp(-\sigma(x^k - x'^k)^2)\right)^d \tag{C.17}$$

- kernel de splines lineales en una dimensión

$$\mathrm{k}(\mathrm{x},\mathrm{x}') = 1 + \mathrm{x}\mathrm{x}'\min(\mathrm{x},\mathrm{x}') - \frac{\mathrm{x}+\mathrm{x}'}{2}\min(\mathrm{x},\mathrm{x}') + \frac{\min(\mathrm{x},\mathrm{x}')^3}{3} \tag{C.18}$$

$$\mathrm{k}(\mathrm{x},\mathrm{x}') = 1 + \mathrm{x}\mathrm{x}'\min(\mathrm{x},\mathrm{x}') - \frac{\mathrm{x}+\mathrm{x}'}{2}\min(\mathrm{x},\mathrm{x}') + \frac{\min(\mathrm{x},\mathrm{x}')^3}{3}$$

  - para el caso multidimensional

$$\mathrm{k}(\mathrm{x},\mathrm{x}') = \prod_{k=1}^{n} k(x^k, x'^k) \tag{C.19}$$

Los kernels Gaussianos, Laplacianos y Besselianos son generalistas utilizados cuando no se tiene más información. El kernel polinomial es muy empleado en el procesamiento de imágenes y los kernels ANOVA y de splines tienen su campo de aplicación en los análisis de regresión.

Asumiendo una postura constructiva de menor a mayor nivel de complejidad, se puede simplificar la ecuación $\langle \mathrm{w}, \Phi(\mathrm{x})\rangle + \mathrm{b} = 0$ en la forma perfectamente separable $\langle \mathrm{w}, \mathrm{x}\rangle + \mathrm{b} = 0$ con w y b coeficientes reales. El hiperplano de separación tiene que cumplir las siguientes restricciones para todo $x_i=(x_1,\ldots,x_m)\in\mathbb{R}^m$, sin error alguno

$$\langle \mathrm{w}, \mathrm{x}\rangle + \mathrm{b} = 0 \tag{C.20}$$

$$\langle \mathrm{w}, \mathrm{x}_i\rangle + \mathrm{b} \geq 0 \text{ si } \mathrm{y}_i = +1 \tag{C.21}$$

$$\langle \mathrm{w}, \mathrm{x}_i\rangle + \mathrm{b} \leq 0 \text{ si } \mathrm{y}_i = -1 \ (i = 1 \ldots m) \tag{C.22}$$

También se puede escribir

$$y_i(\langle w, \mathrm{x}_i\rangle + b) \geq 0 \ (i = 1 \ldots m) \tag{C.23}$$

Llegando a una forma atenuada

$$y_i D(x_i) \geq 0 \ (i = 1 \ldots m) \tag{C.24}$$

Evidentemente el hiperplano separador no es único sino que existen infinitos hiperplanos que cumplan las restricciones. La siguiente cuestión es si de entre este conjunto infinito de hiperplanos se pude seleccionar uno que sea óptimo.

Esta cuestión se dilucida definiendo el margen de un hiperplano de separación, que se denota por τ, como la mínima distancia entre dicho hiperplano y el dato más cercano de cualquiera de las dos clases. A partir de esta definición, un hiperplano de separación se denomina como óptimo cuando su margen es de tamaño máximo.

Una propiedad clara de la definición del hiperplano de separación óptimo es que equidista del dato más cercano de cada clase. Esto se puede ver mediante una sencilla demostración por reducción al absurdo. Supongamos que la distancia del hiperplano óptimo al dato más próximo de la clase +1 fuese inferior que el correspondiente al dato más cercano de la clase −1. Esto significaría que el hiperplano se puede alejar del dato de la clase +1 una distancia tal que la distancia del hiperplano a dicho dato sea superior a la anterior y, a su vez, siga siendo inferior que la distancia al dato más cercano de la clase −1. Así, se llega al absurdo de que se puede aumentar el tamaño del margen cuando inicialmente se había partido de que este era máximo (hiperplano óptimo). Se aplica un razonamiento análogo en el caso de suponer que la distancia del hiperplano óptimo al dato más próximo de la clase −1 fuese inferior que el correspondiente al dato más cercano de la clase +1.

Por geometría, se infiere que la distancia entre un hiperplano de separación D(x) y un dato x´ viene dada por

$$\frac{D(x')}{\|w\|} \tag{C.25}$$

El vector w y el parámetro b definen el hiperplano D(x) y este vector es perpendicular al hiperplano. Como $y_i D(x_i) \geq 0$ todos los datos de entrenamiento cumplirán que

$$\frac{y_i D(x_i)}{\|w\|} \geq \tau \; (i = 1 \ldots m) \tag{C.26}$$

De la ecuación anterior, se infiere que encontrar el hiperplano óptimo equivale a encontrar el valor de w que maximiza el margen. No obstante, existen infinitas soluciones que únicamente se diferencian en w. Obviamente, todas las funciones $\lambda(\langle w, x\rangle + b)$ con λ∈ℝ constituyen el mismo hiperplano. Para limitar todo este conjunto de soluciones a una sola

$$y_i D(x_i) \geq \tau \|w\| \ (i = 1 \ldots m) \tag{C.27}$$

Si $\tau\|w\| = 1$, se aprecia que al aumentar el margen es equivalente a disminuir la norma de w

$$\tau = \frac{1}{\|w\|} \tag{C.28}$$

Por definición, un hiperplano de separación óptimo posee un margen máximo y, por tanto, un valor mínimo de $\|w\|$ y tiene que cumplir $y_i D(x_i) \geq \tau\|w\|$ y $\tau = 1/\|w\|$

$$y_i D(x_i) > 1 \ (i = 1 \ldots m) \tag{C.29}$$

Que es equivalente a

$$y_i(\langle w, \mathrm{x}_i \rangle + b) \geq 1 \ (i = 1 \ldots m) \tag{C.30}$$

Cuanto más grande es el margen, mayor será la distancia de separación existirá entre ambas clases. Aquellos datos tales que $y_i(\langle w, \mathrm{x}_i \rangle + b) = 1$ son los conocidos vectores soporte. De este modo, el hiperplano óptimo puede definirse como un problema equivalente a encontrar los valores de *w* y *b* que minimizan el funcional $f(w) = \|w\|$ y cumplen las ligaduras $y_i(\langle w, \mathrm{x}_i \rangle + b) \geq 1$ quedando

$$\begin{gathered} minimizar\ f(w) = \frac{1}{2}\|w\|^2 = \frac{1}{2}\langle \mathrm{w}, w \rangle \\ y_i(\langle w, \mathrm{x}_i \rangle + b) - 1 \geq 0 \ (i = 1 \ldots m) \end{gathered} \tag{C.31}$$

Este sistema se corresponde con un problema de programación cuadrático que puede resolverse mediante la teoría clásica de optimización. Sin descender a la teoría propiamente dicha, esta teoría define un problema primal, con una forma dual si la función a optimizar y las ligaduras son funciones estrictamente convexas. Resolver el problema dual resuelve el problema primal.

El problema descrito anteriormente cumple el criterio de convexidad y a tenor de lo dicho antes, tiene un dual. Para ello es necesario construir un problema de optimización sin restricciones haciendo uso de la Lagrangiana:

$$L(w, b, \alpha) = \frac{1}{2}\langle \mathrm{w}, w \rangle - \sum_{i=1}^{m} \alpha_i [y_i(\langle w, \mathrm{x}_i \rangle + b) - 1] \tag{C.32}$$

Con α_i los multiplicadores de Lagrange. Aplicando la primera condición de KKT (Karush-Kuhn-Tucker)

$$\frac{\partial L(w^*, b^*, \alpha)}{\partial w} = w^* - \sum_{i=1}^{m} \alpha_i y_i \mathrm{x}_i = 0 \ (i = 1 \dots m) \tag{C.33}$$

$$\frac{\partial L(w^*, b^*, \alpha)}{\partial b} = \sum_{i=1}^{m} \alpha_i y_i = 0 \ (i = 1 \dots m) \tag{C.34}$$

La segunda condición KKT:

$$\alpha_i [1 - y_i (\langle w^*, \mathrm{x}_i \rangle + b^*)] = 0 \ (i = 1 \dots m) \tag{C.35}$$

Quedando

$$w^* = \sum_{i=1}^{m} \alpha_i y_i \mathrm{x}_i = 0 \ (i = 1 \dots m) \tag{C.36}$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \ (i = 1 \dots m) \tag{C.37}$$

Estas ecuaciones permiten determinar el problema dual

$$L(w, b, \alpha) = \frac{1}{2} \langle \mathrm{w}, w \rangle - \sum_{i=1}^{m} \alpha_i y_i \langle w, \mathrm{x}_i \rangle - b \sum_{i=1}^{m} \alpha_i y_i + \sum_{i=1}^{m} \alpha_i \tag{C.38}$$

Arreglando la ecuación

$$L(\alpha) = -\frac{1}{2} \left(\sum_{i=1}^{m} \alpha_i y_i \mathrm{x}_i \right) \left(\sum_{j=1}^{m} \alpha_j y_j \mathrm{x}_j \right) + \sum_{i=1}^{m} \alpha_i \tag{C.39}$$

$$L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i \, y_j \langle \mathrm{x}_i, \mathrm{x}_j \rangle \tag{C.40}$$

Se ha convertido el el problema de minimización primal en su dual maximizante sujeto a las ligaduras de los multiplicadores de Lagrange:

$$\begin{gathered} maximizar \ L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i \, y_j \langle \mathrm{x}_i, \mathrm{x}_j \rangle \\ \sum_{i=1}^{m} \alpha_i y_i = 0 \ (i = 1 \dots m) \end{gathered} \tag{C.41}$$

$$\alpha_i \geq 0 \ (i = 1 \dots m)$$

Este problema es abordable mediante técnicas convencionales de programación cuadrática igual que el primal, aunque el coste computacional de resolver este problema es inferior al primal.

De la segunda condición KKT, se infiere que si los α_i>0 entonces

$$y_i(\langle w^*, \mathrm{x}_i \rangle + b^*) = 1 \tag{C.42}$$

Cuando la restricción toma el caso igual que, se tienen los vectores soporte y por tanto solo los datos con α_i>0 son los vectores soporte. Así, el hiperplano de separación se construirá con una combinación lineal de los vectores soporte y el resto de los datos tendrán asociado un α_i=0.

El parámetro b^* se calcula a partir del conjunto de vectores soporte

$$b^* = \frac{1}{M_{vs}} \sum_{vs=1}^{M_{vs}} (\mathrm{y}_{vs} - \langle w^*, \mathrm{x}_{vs} \rangle) \tag{C.43}$$

Generalmente los problemas reales están sometidos a ruido distorsionador y por tanto no son linealmente separables. En estas circunstancias, hay errores de clasificación en algunos de los datos de partida que constituyen el conjunto de entrenamiento.

Así, se tienen dos tipos de datos. El primero está formado por aquellos que caen dentro del margen de la clase correcta de acuerdo al hiperplano de separación. El segundo son los datos que caen al otro lado de dicho hiperplano. Estos dos casos son no separables [no cumplen la condición $y_i(\langle w, \mathrm{x}_i \rangle + b) \geq 1$], pero los primeros están clasificados de forma correcta y los segundos no.

No obstante, para encontrar un hiperplano óptimo para los datos que si son separables, se definen un grupo de variables reales positivas, llamadas variables de holgura ξ_i (i=1,...,m)

$$y_i(\langle w, \mathrm{x}_i \rangle + b) \geq 1 - \xi_i \ (i = 1 \dots m) \tag{C.44}$$

Para cada par (x_i,y_i), su variable de holgura ξ_i, representa una desviación del caso separable con lo que variables de holgura nulas corresponden a datos separables, mayores que cero se corresponden con datos no separables y

superiores a uno con datos no separables y mal clasificados. Sumando las variables de holgura, se tiene una medida del número de datos no separables.

Para no clasificar erróneamente muchos datos, la función a optimizar debe incluir los errores de clasificación del hiperplano de separación

$$f(w,\xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \tag{C.45}$$

C es una constante elegida por el operador que permite controlar el grado de sobreajuste del clasificador y el número de datos no separables. C muy grande compensa con ξ_i pequeños y los datos son más separables. C muy pequeño se contrarresta con ξ_i grandes y un gran número de datos están mal clasificados.

Con lo que el nuevo problema de optimización se concreta como

$$\begin{gathered} minimizar\ \frac{1}{2}\langle \mathrm{w}, w\rangle + C\sum_{i=1}^{m}\xi_i \\ y_i(\langle w, \mathrm{x}_i\rangle + b) + \xi_i - 1\ \geq 0 \\ \xi_i \geq 0\ (i = 1 \ldots m) \end{gathered} \tag{C.46}$$

Este hiperplano recibe el nombre de hiperplano de separación de margen blando como contrapartida al perfectamente separable conocido como hiperplano de margen duro. Análogamente a como se hizo antes, cuando el problema de optimización se corresponde a un espacio de características de muy alta dimensionalidad, se pude simplificar transformándolo en su dual. Similarmente a como se efectuó antes se construye la lagrangiana

$$\begin{aligned} L(w,b,\xi,\alpha,\beta) = \frac{1}{2}\langle \mathrm{w}, w\rangle + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i[y_i(\langle w, \mathrm{x}_i\rangle + b) + \xi_i - 1] \\ - \sum_{i=1}^{m}\beta_i\xi_i \end{aligned} \tag{C.47}$$

Aplicando las condiciones de KKT

$$\frac{\partial L}{\partial w} = w^* - \sum_{i=1}^{m}\alpha_i y_i \mathrm{x}_i = 0 \tag{C.48}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{C.49}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \tag{C.50}$$

$$\alpha_i[1 - y_i(\langle w^*, \mathrm{x}_i\rangle + b^*) - \xi_i] = 0 \ (i = 1 \dots m) \tag{C.51}$$

$$\beta_i \cdot \xi_i = 0 \ (i = 1 \dots m) \tag{C.52}$$

Resolviendo se relacionan las variables del problema primal (w,b,ξ) con las del dual (α,β)

$$L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i \, y_j \langle \mathrm{x}_i, \mathrm{x}_j \rangle \tag{C.53}$$

El problema dual maximizante queda

$$maximizar \ L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i \, y_j \langle \mathrm{x}_i, \mathrm{x}_j \rangle$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{C.54}$$

$$0 \le \alpha_i \le C \ (i = 1 \dots m)$$

De la ecuación $C=\alpha_i+\beta_i$ se infiere que si $\alpha_i=0$ entonces $C=\beta_i$ y de $\beta_i \cdot \xi_i=0$ se deduce que $\xi_i=0$ (datos separables). Para el caso no separable, $\xi_i>0$ y de $\beta_i \cdot \xi_i=0$ se tiene que $\beta_i=0$, y de $C=\alpha_i+\beta_i$ se llega a $\alpha_i=C$ y a

$$[1 - y_i(\langle w^*, \mathrm{x}_i\rangle + b^*) - \xi_i] = 0 \tag{C.55}$$

Esta ecuación tiene dos alternativas. En la primera, x_i es no separable y está bien clasificado ($y_i(\langle w^*, \mathrm{x}_i\rangle + b^*) - \xi_i \ge 0$) con lo que $\xi_i = 1 - |y_i(\langle w^*, \mathrm{x}_i\rangle + b^*)|$. En la segunda x_i es no separable y está mal clasificado ($y_i(\langle w^*, \mathrm{x}_i\rangle + b^*) - \xi_i < 0$) con lo que $\xi_i = 1 + |y_i(\langle w^*, \mathrm{x}_i\rangle + b^*)|$.

Por otra parte, para el caso $0<\alpha_i<C$, considerando la ligadura $C=\alpha_i+\beta_i$ se deduce que β_i no puede ser nulo, y de $\beta_i \cdot \xi_i=0$ se extrae que $\xi_i=0$. De igual forma, para el caso $0<\alpha_i<C$, de $\alpha_i[1 - y_i(\langle w^*, \mathrm{x}_i\rangle + b^*) - \xi_i] = 0$ y de $\xi_i=0$ se llega a

$$1 - y_i(\langle w^*, \mathrm{x}_i\rangle + b^*) = 0 \tag{C.56}$$

Así, x_i es vector soporte si y solo si $0<\alpha_i<C$. El parámetro b^* se calcula a partir de

$$b^* = y_i - \sum_{j=1}^{m} \alpha_i^* \mathrm{y}_i \langle \mathrm{x}_j, \mathrm{x}_i\rangle \ (0 < \alpha_i < C) \tag{C.57}$$

Con $\alpha_i^* = 1 \ldots m$ correspondientes al problema dual. Para datos cuasi-separables, ($\alpha_i^* \neq 0$) hay dos tipos de datos, los que $0 < \alpha_i^* < C$ y los que $\alpha_i^* = C$ (datos no separables que corresponden a vectores soporte acotados.

Finalmente se llega al caso en que los datos no sean separables/cuasi-separables y es necesario evolucionar de los hiperplanos lineales hacia otro tipo de estructura tipo funciones no lineales que definen espacios transformados de alta dimensionalidad donde se buscan los hiperplanos de separación óptimos en dichos espacios transformados (espacio de características).

Así, la aplicación de transformación no lineal $\Phi: X \rightarrow H$ hace corresponder a cada dato x con un punto en el espacio de características. Construyendo un hiperplano de separación lineal en el nuevo espacio, la frontera lineal de decisión obtenida en el espacio de características será transformada en una frontera no lineal en el espacio original. La función de decisión en el espacio de características vendrá dada por $\langle \mathrm{w}, \Phi(\mathrm{x})\rangle$ y en dual, la función de decisión se obtiene de $w^* = \sum_{i=1}^{m} \alpha_i y_i \mathrm{x}_i$ quedando

$$\langle \mathrm{w}, \Phi(\mathrm{x})\rangle = \sum_{i=1}^{m} \alpha_i^* y_i K(x, x_i) \tag{C.58}$$

Siendo la función kernel $\mathrm{k}(\mathrm{x}, \mathrm{x}') = \langle \Phi(\mathrm{x}), \Phi(\mathrm{x}')\rangle$ un producto escalar. El problema dual maximizante queda

$$\begin{gathered} maximizar \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i \, y_j \langle \mathrm{x}_i, \mathrm{x}_j\rangle \\ \sum_{i=1}^{m} \alpha_i y_i = 0 \end{gathered} \tag{C.59}$$

$$0 \leq \alpha_i \leq C \ (i = 1 \dots m)$$

En el caso de regresión se parte de unos datos de entrenamiento y se asume que es posible cuasi-ajustarlos mediante una función lineal encontrando unos parámetros w

$$f(x) = \langle \mathrm{w}, \mathrm{x} \rangle + \mathrm{b} = 0 \tag{C.60}$$

Tal como se ha comentado anteriormente se emplea un modelo blando mediante dos variables de holgura, ξ_i^+ y ξ_i^-, similar al empleado en el problema de clasificación. La suma de todas las variables de holgura estiman el número de datos con error no nulo.

$$\begin{gathered} minimizar \ \frac{1}{2}\langle \mathrm{w}, w \rangle + C \sum_{i=1}^{m} (\xi_i^+ + \xi_i^-) \\ (\langle \mathrm{w}, x_i \rangle + b) - y_i - \epsilon - \xi_i^+ \leq 0 \\ y_i - (\langle \mathrm{w}, x_i \rangle + b) - \epsilon - \xi_i^- \leq 0 \\ \xi_i^+, \xi_i^- \geq 0 \ (i = 1 \dots m) \end{gathered} \tag{C.61}$$

Para la transformación al problema dual se necesita obtener la función Lagrangiana

$$\begin{aligned} & L(w, b, \xi^+, \xi^-, \alpha^+, \alpha^-, \beta^+, \beta^-) \\ & = \frac{1}{2}\langle \mathrm{w}, w \rangle + C \sum_{i=1}^{m} \sum_{i=1}^{m} (\xi_i^+ + \xi_i^-) \\ & + \sum_{i=1}^{m} \alpha_i^+ [(\langle \mathrm{w}, x_i \rangle + b) - y_i - \epsilon - \xi_i^+] \\ & + \sum_{i=1}^{m} \alpha_i^- [y_i - (\langle \mathrm{w}, x_i \rangle + b) - \epsilon - \xi_i^-] + \sum_{i=1}^{m} \beta_i^+ \xi_i^+ - \sum_{i=1}^{m} \beta_i^- \xi_i^- \end{aligned} \tag{C.62}$$

Con las condiciones KKT

$$\frac{\partial L}{\partial w} = w + \sum_{i=1}^{m} \alpha_i^+ \mathrm{x}_i - \sum_{i=1}^{m} \alpha_i^- \mathrm{x}_i = 0 \tag{C.63}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} \alpha_i^+ - \sum_{i=1}^{m} \alpha_i^- = 0 \tag{C.64}$$

$$\frac{\partial L}{\partial \xi_i^+} = C - \alpha_i^+ - \beta_i^+ = 0 \tag{C.65}$$

$$\frac{\partial L}{\partial \xi_i^-} = C - \alpha_i^- - \beta_i^- = 0 \tag{C.66}$$

$$\alpha_i^+ [(\langle \mathrm{w}, x_i \rangle + b) - y_i - \epsilon - \xi_i^+] = 0 \tag{C.67}$$

$$\alpha_i^- [y_i - (\langle \mathrm{w}, x_i \rangle + b) - \epsilon - \xi_i^-] = 0 \tag{C.68}$$

$$\beta_i^+ \xi_i^+ = 0 \tag{C.69}$$

$$\beta_i^- \xi_i^- = 0 \tag{C.70}$$

Resolviendo se obtiene la formulación del problema dual

$$\begin{aligned} L(\alpha^+, \alpha^-) = & \sum_{i=1}^{m} (\alpha_i^- - \alpha_i^+)\, \mathrm{y}_i - \epsilon \sum_{i=1}^{m} (\alpha_i^- + \alpha_i^+) \\ & - \frac{1}{2} \sum_{i,j=1}^{m} (\alpha_i^- - \alpha_i^+)(\alpha_j^- - \alpha_j^+) \langle \mathrm{x}_i, \mathrm{x}_j \rangle \end{aligned} \tag{C.71}$$

El problema dual maximizante queda

$$\begin{aligned} maximizar & \sum_{i=1}^{m} (\alpha_i^- - \alpha_i^+)\, \mathrm{y}_i - \epsilon \sum_{i=1}^{m} (\alpha_i^- + \alpha_i^+) \\ & - \frac{1}{2} \sum_{i,j=1}^{m} (\alpha_i^- - \alpha_i^+)(\alpha_j^- - \alpha_j^+) \langle \mathrm{x}_i, \mathrm{x}_j \rangle \\ & \sum_{i=1}^{m} (\alpha_i^+ - \alpha_i^-) = 0 \end{aligned} \tag{C.72}$$

$$0 \leq \alpha_i^+, \alpha_i^- \leq C \ (i = 1 \ldots m)$$

El parámetro b* se calcula a partir de

$$b^* = y_i - \langle w^*, \mathrm{x}_i \rangle + \epsilon \ (0 < \alpha_i^+ < C) \tag{C.73}$$

$$b^* = y_i - \langle w^*, \mathrm{x}_i \rangle + \epsilon \ (0 < \alpha_i^- < C) \tag{C.74}$$

El valor de b* siempre es único ya que $\alpha_i^+ \cdot \alpha_i^- = 0$. Si se cumple $0 < \alpha_i^+ < C$ entonces $\alpha_i^- = 0$ y viceversa.

En el caso de que no sea posible emplear una función lineal, se puede usar un método similar al del problema de clasificación no separable linealmente. Los

datos del espacio origen se transforman en el espacio de características donde se puede efectuar un análisis lineal. Esta transformación va a depender del kernel en cuestión. También es necesario seleccionar Є y C. Considerando la función lineal

$$f(x) = \sum_{i=1}^{m} (\alpha_i^+ - \alpha_i^-)\, K(x, x_i) \tag{C.75}$$

α_i^+ y α_i^- se obtienen resolviendo el problema dual

$$\begin{aligned} maximizar \sum_{i=1}^{m} (\alpha_i^- - \alpha_i^+)\, \mathrm{y}_i - \epsilon \sum_{i=1}^{m} (\alpha_i^- + \alpha_i^+) \\ -\frac{1}{2} \sum_{i,j=1}^{m} (\alpha_i^- - \alpha_i^+)(\alpha_j^- - \alpha_j^+) K\langle \mathrm{x}_i, \mathrm{x}_j \rangle \\ \sum_{i=1}^{m} (\alpha_i^+ - \alpha_i^-) = 0 \\ 0 \le \alpha_i^+, \alpha_i^- \le C \ (i = 1 \dots m) \end{aligned} \tag{C.76}$$

Apéndice D. Tratamiento estadístico de imágenes

Como se ha visto previamente, la gestión y coordinación de enjambres requiere volar de forma autónoma, intercomunicándose entre RPAs para evitar colisiones, procesando información del entorno y detectando amenazas. Esto implica un elevado esfuerzo computacional, lo cual debe hacerse desde unos aparatos que generalmente disponen de un espacio limitado a bordo para albergar CPUs. Diversas tecnologías intentan aliviar este trabajo computacional, siendo uno de ellos el tratamiento estadístico de las imágenes captadas por la cámara de cada RPA del enjambre. Tradicionalmente las tecnologías de reconocimiento de patrones se basan en la comparación de imágenes, o partes de la misma, con patrones predefinidos, pixel a pixel, con un coste computacional considerable. No obstante, una alternativa viable sería usar en lugar de cada imagen, una función estadística representativa de dicha imagen digital, como el histograma.

Así, mediante técnicas de análisis no-paramétricas se define un estadístico de contraste, que aplicado a una secuencia de histogramas de las imágenes captadas por la cámara de un RPA, permite establecer un parámetro de medida, de tal modo, que mientras este parámetro se encuentre dentro de rango, el agente no necesitará comunicarse con el enjambre, reduciendo estas transmisiones a aquellas situaciones donde este parámetro se salga de margen.

En estas condiciones, cualquier operación que se tome para atenuar la carga de trabajo de CPU de cada agente del enjambre representa una importante

ventaja. Para ello, un método posible sería analizar las imágenes captadas por la cámara de cada RPA del enjambre, a través de sus histogramas, definiendo un estadístico de contraste, de tal modo que mientras un cierto parámetro se encuentre dentro de margen, el agente no solicitará comunicarse con el resto del enjambre, recurriendo a la transmisión cuando este parámetro se salga fuera de rango.

El reconocimiento de patrones está fundamentado en comparar imágenes, o elementos de la misma, con patrones predefinidos, evaluando el grado de igualdad o desigualdad, lo cual se suele realizar pixel a pixel, con el consiguiente coste de procesado. Matemáticamente, el problema consiste en evaluar la similitud entre dos funciones $f(x,y)$ y $g(x,y)$ empleando funcionales del tipo $\int \Psi(f,g)d\sigma$.

El histograma de una imagen f es una representación gráfica de una variable en forma de barras, de manera que la superficie de cada barra es proporcional a la frecuencia de la distribución tonal de intensidad de color de una imagen digital f; o sea, esta gráfica representa el número de píxeles presentes en f por cada valor tonal. Píxel es la menor unidad homogénea de color que forma parte de una imagen digital. Aplicando el formalismo matemático, una imagen de dimensión NxM se define como una aplicación $f:NxM \rightarrow QxQ$ donde $Q=\{0,\ldots,q-1\}$ para una imagen con q niveles de cuantización. Un histograma h de una imagen f se define como una aplicación $h:Q \rightarrow NxM$.

Si bien el histograma no caracteriza todas las propiedades de una imagen, como las relaciones espaciales entre sus pixeles, es una función estadística representativa en múltiples aplicaciones. Se trata de una herramienta muy útil, disponible en programas de tratamiento de imágenes y mucho más asequible en términos de coste de procesado. Como propiedades básicas de un histograma se puede citar que la imagen f no se puede deducir a partir de h, dos imágenes diferentes pueden tener asociado el mismo histograma y los histogramas no contienen información espacial sobre la imagen. No obstante, posee propiedades interesantes que debería poseer cualquier método de medida de imágenes digitales, como identidad, simetría, invariancia frente a desplazamientos, escalado y rotaciones, propiedades que no poseen otros funcionales.

Dada una imagen *f* y su histograma *h*, considerado como una función de densidad de cada nivel de gris y (variable aleatoria continua). Se normaliza dicho histograma *H(y)=h(y)/(NxM)* con *NxM* el número total de píxeles. El histograma acumulado normalizado o función de distribución es $F(y) = \int_{-\infty}^{y} H(\lambda)d\lambda$ y el histograma acumulado $G(y) = \int_{0}^{y} h(\lambda)d\lambda$, con lo que *F(y)=G(y)/(NxM)* dado que $\int_{-\infty}^{0} h(\alpha)d\alpha = 0$. Con imágenes digitales estas funciones son discretas, si bien se ha supuesto la hipótesis de continuidad por facilidad en el desarrollo. Por simplicidad, se van a considerar histogramas de nivel de gris de imágenes monocromas. En el caso de imágenes a color, en lugar de un único valor de intensidad de nivel de gris, los píxeles se cuantifican usando tres componentes con distinto significado según el modelo de color utilizado.

El contraste de hipótesis es un procedimiento que permite, partiendo de una muestra significativa y elegida al azar, formular conclusiones que rechazan o no una hipótesis de partida acerca de un argumento ignorado de una población. Sean dos imágenes digitales f_1 y f_2, con histogramas acumulados G_1 y G_2, y funciones de distribución F_1 y F_2. Se puede medir la similitud entre dos imágenes, evaluando la diferencia entre sus funciones de distribución. Para ello, se va a emplear el test Z de Kolmogorov-Smirnov (KS), una prueba no-paramétrica de bondad de ajuste, que permite medir el grado de concordancia existente entre la distribución de un conjunto de datos (muestra) y una distribución teórica específica. El test KS se basa en el estadístico de contraste $D = max|F_1(x) - F_2(x)|$.

La hipótesis nula H_0 es la que provisionalmente se acepta como cierta y se somete a contrastación empírica. En el test KS la hipótesis nula es que los histogramas de las dos imágenes pertenezcan a la misma población (o sean iguales) frente a la hipótesis alternativa H_1, complementaria a H_0. Así, se puede demostrar que la probabilidad de que se verifique la hipótesis nula P_{H0} viene dada, cuando el tamaño de muestra tiende a infinito, como

$$P_{H_0}\left[D > \lambda\left(\frac{1}{n}+\frac{1}{m}\right)^{\frac{1}{2}}\right] = 2\sum_{k=1}^{\infty}(-1)^{k-1}\exp(-2k^2\lambda^2) \qquad \text{(D.1)}$$

$$\lambda = \left[-½\left(\frac{1}{n}+\frac{1}{m}\right)\right]\ln\left(\frac{\alpha}{2}\right)$$

Siendo n y m el número de píxeles de cada una de las imágenes, $\alpha \in [0,1]$ el nivel de significación y (1-α) el nivel de confianza.

El nivel de confianza del contraste (1-α) es la probabilidad de no rechazar la hipótesis nula H_0. De este moco, definido un valor α, la región de no rechazo se define como aquella formada por el conjunto de datos del estadístico de contraste que lleva a no rechazar la hipótesis nula H_0; es decir, cuando $D < [-½(1/n + 1/m)]\ln(\alpha/2)$. Otra forma de verlo, definido un valor característico del estadístico D, se puede calcular el mayor nivel de significación α_m tal que no se rechaza la hipótesis nula H_0; esto es, $\alpha_m = 2\exp[-2Dnm/(n+m)]$. Así, se tiene que cuanto más se parezcan los histogramas de las dos imágenes, menor será el valor del estadístico D y por consiguiente mayor será el nivel de significación α_m.

Así, se puede aplicar este procedimiento a una secuencia de imágenes obtenido de un video de youtube grabado por la cámara de un RPA que muestra la destrucción de Mosul por los efectos de la Guerra. A efectos académicos, este video se ha muestreado con una frecuencia de una imagen por segundo, si bien esta frecuencia debería ser mucho mayor en una situación real. Empleando el lenguaje de programación r-project, y concretamente el paquete imager, se ha confeccionado un programa computacional que calcula la medida de la diferencia entre cada dos imágenes consecutivas.

Figura D.1. Imagen t=14 segundos.

A continuación, se va a exponer la diferencia entre dos imágenes consecutivas del video. Como la toma de tiempos entre ambas imágenes es muy pequeña, estas imágenes son muy similares y la segunda puede considerarse como una pequeña perturbación de la primera. Obtenido $F_1(x)$-$F_2(x)$, se calcula el estadístico de contraste D, obteniendo α_m. Así, mientras la segunda imagen no difiera mucho de la primera, el CPU del RPA interpreta que el entorno no ha cambiado mucho y no solicitará comunicarse con el resto del enjambre, hasta que dicha diferencia sea tal que el agente si solicite a los otros RPAs su posición, de modo que pueda mantener su distancia de seguridad con el resto de agentes, evitando las colisiones y conservando la topología del enjambre.

Figura D.2. Imagen t=15 segundos.

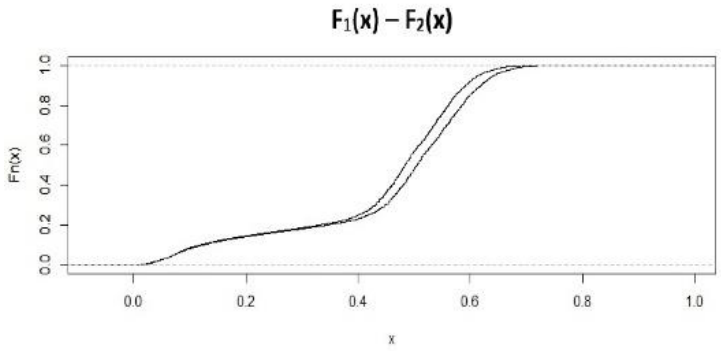

Figura D.3. $F_1(x)$-$F_2(x)$.

Un análisis a realizar es que ocurre cuando la segunda imagen se diferencia mucho de la primera, por la aparición de un obstáculo de dimensiones significativas en la segunda imagen. Este obstáculo pretende simular el efecto de un elemento inesperado que aparece en el campo de visión de la cámara del RPA y como se refleja este efecto en el correspondiente histograma de la imagen.

Figura D.4. Imagen t=15 segundos con obstáculo pequeño.

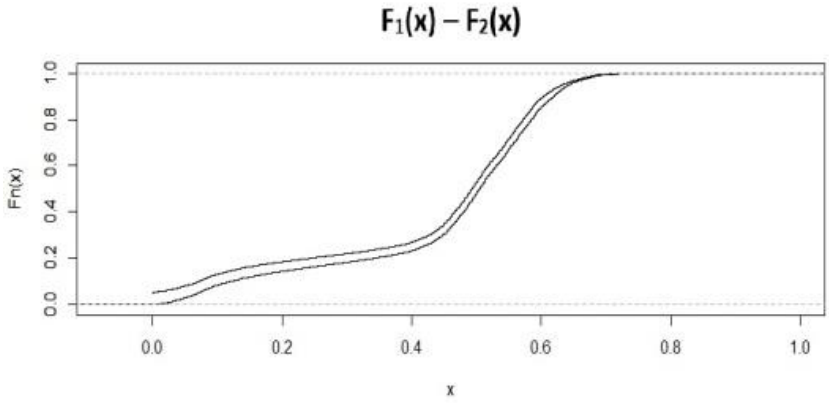

Figura D.5. $F_1(x)$-$F_2(x)$.

La siguiente cuestión es cómo afectaría al histograma si el obstáculo que aparece en la segunda imagen aumenta de tamaño; es decir, el obstáculo se acerca a la cámara del RPA. En este último caso, se aprecia como el estadístico de contraste D del test KS se incrementa respecto al anterior, como era de esperar.

Figura D.6. Imagen t=15 segundos con obstáculo grande.

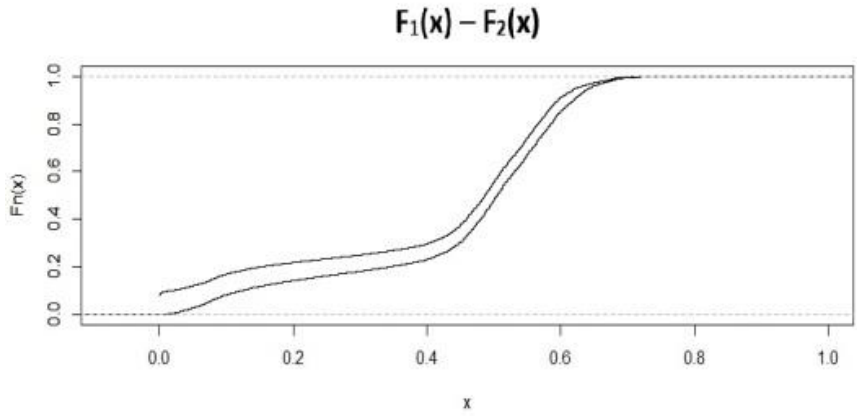

Figura D.7. $F_1(x)$-$F_2(x)$.

A continuación se obtienen los histogramas si el obstáculo fuera de un color distinto al negro, alcanzando un resultado inusual en cuanto al estadístico de contraste D.

Figura D.8. Imagen t=15 segundos con obstáculo de otro color.

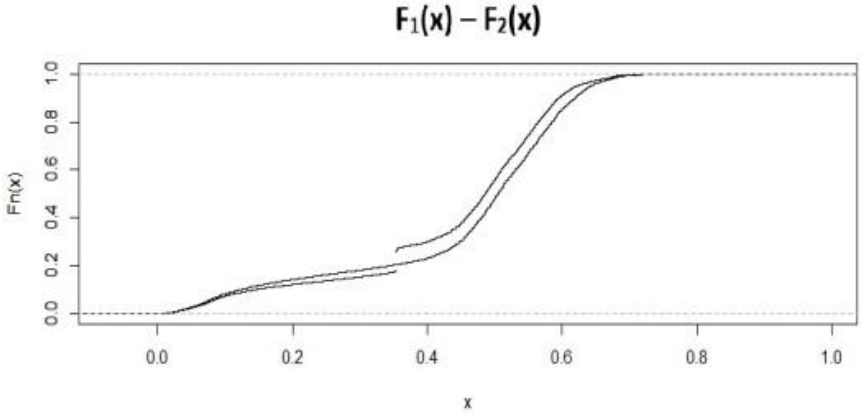

Figura D.9. $F_1(x)$-$F_2(x)$.

Continuando con la investigación, podría aplicarse otro tipo de test no-paramétrico, como el U de Mann-Whitney de sumas de rangos, que en r-project se pone a punto con la función wilcox.test. O bien el test no-paramétrico de Wald-Wolfowitz o de rachas (paquete randtests de r-project). De este modo, las tres pruebas hacen uso del procedimiento de contraste de hipótesis para comprobar si dos muestras proceden de una misma población. Si bien en general las tres pruebas dan resultados razonablemente similares, en situaciones extremas se pueden presentar discrepancias donde un test rechace una hipótesis y otro la acepte.

Una alternativa es utilizar un KS modificado $D = \sum_{i=1}^{i=j} max|F_1(x_i) - F_2(x_i)|$ para un conjunto de valores x_i, distribuidos a lo largo del eje x, con distintos niveles de decisión dependiendo del valor D. Para $D<D_1$ no es necesario tomar ninguna decisión. Para $D>D_1$ el RPA decisor solicitará mensajería de posición a aquellos agentes del enjambre cuya última distancia conocida al decisor es inferior a un valor d_1. Para $D>D_2$ el RPA decisor solicitará mensajería de posición a aquellos agentes del enjambre cuya última distancia conocida al decisor es inferior a un valor d_2, y así sucesivamente. $D_1<D_2<D_3<\ldots$ $d_1<d_2<d_3<\ldots$ Las parejas de valores (d_p,D_p) se obtendrán mediante técnicas empíricas y dependerá de diversos factores como el tipo de RPA del enjambre.

Apéndice E. Código del programa

APLICACION DE LOS ALGORITMOS EPSO Y SVM A LA INTELIGENCIA DE ENJAMBRES DE RPAS EN MISIONES DE SATURACION DE DEFENSAS E ISR

##

UPCT

##

DECLARACIONES INICIALES

velocidadcru<-17 ## velocidad de crucero del enjambre (m/s)

velocidadper<-6 ## velocidad de perdida del RPA del enjambre (m/s)

tam<-1 ## tamano promedio del RPA (m) (si hubiera varios el valor mas grande)

factmin<-5 ## factor mínimo que multiplica al tamano promedio para evitar el riesgo de colisiones (KAIST)

factmax<-15 ## factor maximo que multiplica al tamano promedio para evitar el riesgo de colisiones (KAIST)

heta<-18 ## angulo que puede girar el vector velocidad del RPA por segundo para evitar la colision (10º/segundo)

heta<-heta*pi/180 ## se pasa heta a radianes

deceleracion<-1.5 ## deceleracion del RPA para evitar la colision (m/s2)

aceleracion<-5 ## aceleracion del RPA para recuperar la velocidad nominal despues de decelerar (m/s2)

kfact<-1 ## factor k que afecta a factMmax y factMmin

nRPAs<-8 ## numero de RPAs que componen el enjambre

dimension<-2 ## numero de dimensiones del problema

distanciainicial<-factmax*tam ## distancia inicial que modela el enjambre en t=0

tiempo<-0 ## tiempo durante cada mision SEAD ISR (seg)

pasotiempo<-0.2 ## paso de tiempo de cada mision SEAD ISR (seg)

tiempoSEAD<-60 ## tiempo de cada mision SEAD ISR (seg)

tiempoadimen<-1 ## tiempo adimensional

factdecv<-10 ## factor de decision donde cada RPA cambia el vector velocidad para no colisionar cuando distancia<factdecv*tam

factdeca<-7 ## factor de decision donde cada RPA decelera para no colisionar cuando distancia<factdeca*tam (factdecv>=factdeca)

factMmmm<-15 ## distancia factMmmm*tam de cada RPA al centroide para que los RPAs no se alejen (factMmmm>=factdecv>=factdeca)

factMmax<-25 ## valor maximo de factMmmm de cada RPA al centroide del enjambre factMmax<-kfact*factmax*nRPAs^(1/dimension)/2

factMmin<-factmin ## valor minimo de factMmmm de cada RPA al centroide del enjambre factMmin<-kfact*factmin*nRPAs^(1/dimension)/2

DECLARACIONES INICIALES ENTROPÍA

entropiatiempo<-0 ## entropia sumatoria en cada mision SEAD ISR

velocidadminima<-numeric(length=dimension) ## vector velocidad minima del enjambre

velocidadmaxima<-numeric(length=dimension) ## vector velocidad maxima del enjambre

deltavelocidad<-numeric(length=dimension) ## deltavelocidad<-(velocidadmaxima-velocidadminima)/nRPAs

contadormicroestados<-matrix(0,(nRPAs+1),2) ## matriz de microestados en cada intervalo para calcular la entropia

indiceentropia<-matrix(0,nRPAs,2) ## matriz de entropia en cada intervalo

INICIALIZACIONES

velocidadminima[]<-1000000000000 ## inicializa a infinito

velocidadmaxima[]<--1000000000000 ## inicializa a -infinito

deltavelocidad[]<-0 ## inicializa a 0

contadormicroestados[]<-0 ## inicializa a 0

indiceentropia[]<-0 ## inicializa a 0

entropiatiempo<-0 ## inicializa a 0

DECLARACIONES INICIALES EPSO (EVOLUTIONARY PARTICLE SWARM OPTIMIZATION)

nparticulas<-20 ## numero de particulas de la nube de particulas EPSO

niteraciones<-200 ## numero de iteraciones del algoritmo EPSO

nfact<-3 ## numero de parámetros a optimizar con EPSO

wpso<-1 ## coeficiente de inercia del algoritmo EPSO

c1pso<-2 ## coeficientes de aceleracion del algoritmo EPSO

c2pso<-2 ## coeficientes de aceleracion del algoritmo EPSO

factorcontraccion<-0.95 ## factor de contraccion de los coeficientes de inercia y aceleracion

errormuta<-0.05 ## error de mutacion en el proceso evolutivo EPSO

jiteracionesmem1<-0 ## memoriza el numero de iteraciones para calcular el criterio de parada 1

jiteracionesmem2<-0 ## memoriza el numero de iteraciones para calcular el criterio de parada 2

nlop<-250 ## parametro para calcular el criterio de parada 1

nkop1<-nfact ## parametro para calcular el criterio de parada 1

entropiaoptnk<-0 ## entropia del vector de factores de decision utilizado por el criterio de parada 1

errorop1<-0.5 ## Error a considerar en el criterio de parada 1

nkop2<-nfact ## parametro para calcular el criterio de parada 2

DECLARACIONES DE MATRICES Y VECTORES DEL ENJAMBRE DE RPAS

matrizposicion<-matrix(0,nRPAs,2) ## matriz de posicion del enjambre de RPAs (xi,yi)

matrizvelocidad<-matrix(0,nRPAs,2) ## matriz de velocidad del enjambre de RPAs (vxi,vyi)

distanciaRPAmin<-numeric(length=nRPAs) ## vector de distancias minimas de cada RPA al RPA mas cercano

indiceRPAdistanciamin<-numeric(length=nRPAs) ## vector que identifica para cada RPA el RPA mas cercano

matrizRPAdistanciamin<-matrix(0,nRPAs,2) ## matriz de vectores que para cada RPA apunta al RPA mas cercano

dimensionmatrizglobal<-trunc(tiempoSEAD/pasotiempo) ## dimension global de la matriz de posicion

matrizposicionglobal<-matrix(0,dimensionmatrizglobal,(2*nRPAs)) ## matriz de posicion de todos los RPAs del enjambre

matrizvelocidadglobal<-matrix(0,dimensionmatrizglobal,(2*nRPAs)) ## matriz de posicion de todos los RPAs del enjambre

distanciaRPAcentroide<-numeric(length=nRPAs) ## vector de distancias de cada RPA al centroide del enjambre

matrizdistanciaRPAcentro<-matrix(0,nRPAs,2) ## matriz de vectores que para cada RPA apunta al centroide

matrizposicioncentroide<-numeric(length=dimension) ## vector de posicion del centroide (piloto humano/control autonomo)

matrizvelocidadcentroide<-numeric(length=dimension) ## vector velocidad del centroide (piloto humano/control autonomo)

INICIALIZACIONES

matrizposicion[]<-0 ## inicializa a 0

matrizvelocidad[]<-0 ## inicializa a 0

distanciaRPAmin[]<-0 ## inicializa a 0

indiceRPAdistanciamin[]<-0 ## inicializa a 0

matrizRPAdistanciamin[]<-0 ## inicializa a 0

matrizposicionglobal[]<-0 ## inicializa a 0

matrizvelocidadglobal[]<-0 ## inicializa a 0

distanciaRPAcentroide[]<-0 ## inicializa a 0

matrizdistanciaRPAcentro[]<-0 ## inicializa a 0

matrizposicioncentroide[]<-0 ## inicializa a 0

```
matrizvelocidadcentroide[]<-0 ## inicializa a 0

## INICIALIZACION DE LA POSICION INICIAL DEL ENJAMBRE
matrizposicion[1,1]<-0 ## posicion x del RPA 1
matrizposicion[1,2]<-0 ## posicion y del RPA 1
matrizposicion[2,1]<-distanciainicial ## posicion x del RPA 2
matrizposicion[2,2]<-0 ## posicion y del RPA 2
matrizposicion[3,1]<-2*distanciainicial ## posicion x del RPA 3
matrizposicion[3,2]<-0 ## posicion y del RPA 3
matrizposicion[4,1]<-0 ## posicion x del RPA 4
matrizposicion[4,2]<-distanciainicial ## posicion y del RPA 4
matrizposicion[5,1]<-2*distanciainicial ## posicion x del RPA 5
matrizposicion[5,2]<-distanciainicial ## posicion y del RPA 5
matrizposicion[6,1]<-0 ## posicion x del RPA 6
matrizposicion[6,2]<-2*distanciainicial ## posicion y del RPA 6
matrizposicion[7,1]<-distanciainicial ## posicion x del RPA 7
matrizposicion[7,2]<-2*distanciainicial ## posicion y del RPA 7
matrizposicion[8,1]<-2*distanciainicial ## posicion x del RPA 8
matrizposicion[8,2]<-2*distanciainicial ## posicion y del RPA 8

matrizposicioncentroide[1]<-distanciainicial ## posicion x del centroide
matrizposicioncentroide[2]<-distanciainicial ## posicion y del centroide

## INICIALIZACION DE LA VELOCIDAD INICIAL DEL ENJAMBRE
matrizvelocidad[1,1]<-0.894427191 ## velocidad x del RPA 1
matrizvelocidad[1,2]<-0.447213595 ## velocidad y del RPA 1
```

```
matrizvelocidad[2,1]<-0.447213595 ## velocidad x del RPA 2
matrizvelocidad[2,2]<-0.894427191 ## velocidad y del RPA 2
matrizvelocidad[3,1]<--0.447213595 ## velocidad x del RPA 3
matrizvelocidad[3,2]<-0.894427191 ## velocidad y del RPA 3
matrizvelocidad[4,1]<-0.894427191 ## velocidad x del RPA 4
matrizvelocidad[4,2]<--0.447213595 ## velocidad y del RPA 4
matrizvelocidad[5,1]<--0.894427191 ## velocidad x del RPA 5
matrizvelocidad[5,2]<-0.447213595 ## velocidad y del RPA 5
matrizvelocidad[6,1]<-0.447213595 ## velocidad x del RPA 6
matrizvelocidad[6,2]<--0.894427191 ## velocidad y del RPA 6
matrizvelocidad[7,1]<--0.447213595 ## velocidad x del RPA 7
matrizvelocidad[7,2]<--0.894427191 ## velocidad y del RPA 7
matrizvelocidad[8,1]<--0.894427191 ## velocidad x del RPA 8
matrizvelocidad[8,2]<--0.447213595 ## velocidad y del RPA 8


## INICIALIZACION DE MAGNITUDES GLOBALES
matrizposicionglobal[1,1]<-matrizposicion[1,1] ## posicion x del RPA 1
matrizposicionglobal[1,2]<-matrizposicion[1,2] ## posicion y del RPA 1
matrizposicionglobal[1,3]<-matrizposicion[2,1] ## posicion x del RPA 2
matrizposicionglobal[1,4]<-matrizposicion[2,2] ## posicion y del RPA 2
matrizposicionglobal[1,5]<-matrizposicion[3,1] ## posicion x del RPA 3
matrizposicionglobal[1,6]<-matrizposicion[3,2] ## posicion y del RPA 3
matrizposicionglobal[1,7]<-matrizposicion[4,1] ## posicion x del RPA 4
matrizposicionglobal[1,8]<-matrizposicion[4,2] ## posicion y del RPA 4
matrizposicionglobal[1,9]<-matrizposicion[5,1] ## posicion x del RPA 5
```

```
matrizposicionglobal[1,10]<-matrizposicion[5,2] ## posicion y del RPA 5
matrizposicionglobal[1,11]<-matrizposicion[6,1] ## posicion x del RPA 6
matrizposicionglobal[1,12]<-matrizposicion[6,2] ## posicion y del RPA 6
matrizposicionglobal[1,13]<-matrizposicion[7,1] ## posicion x del RPA 7
matrizposicionglobal[1,14]<-matrizposicion[7,2] ## posicion y del RPA 7
matrizposicionglobal[1,15]<-matrizposicion[8,1] ## posicion x del RPA 8
matrizposicionglobal[1,16]<-matrizposicion[8,2] ## posicion y del RPA 8

matrizvelocidadglobal[1,1]<-matrizvelocidad[1,1] ## velocidad x del RPA 1
matrizvelocidadglobal[1,2]<-matrizvelocidad[1,2] ## velocidad y del RPA 1
matrizvelocidadglobal[1,3]<-matrizvelocidad[2,1] ## velocidad x del RPA 2
matrizvelocidadglobal[1,4]<-matrizvelocidad[2,2] ## velocidad y del RPA 2
matrizvelocidadglobal[1,5]<-matrizvelocidad[3,1] ## velocidad x del RPA 3
matrizvelocidadglobal[1,6]<-matrizvelocidad[3,2] ## velocidad y del RPA 3
matrizvelocidadglobal[1,7]<-matrizvelocidad[4,1] ## velocidad x del RPA 4
matrizvelocidadglobal[1,8]<-matrizvelocidad[4,2] ## velocidad y del RPA 4
matrizvelocidadglobal[1,9]<-matrizvelocidad[5,1] ## velocidad x del RPA 5
matrizvelocidadglobal[1,10]<-matrizvelocidad[5,2] ## velocidad y del RPA 5
matrizvelocidadglobal[1,11]<-matrizvelocidad[6,1] ## velocidad x del RPA 6
matrizvelocidadglobal[1,12]<-matrizvelocidad[6,2] ## velocidad y del RPA 6
matrizvelocidadglobal[1,13]<-matrizvelocidad[7,1] ## velocidad x del RPA 7
matrizvelocidadglobal[1,14]<-matrizvelocidad[7,2] ## velocidad y del RPA 7
matrizvelocidadglobal[1,15]<-matrizvelocidad[8,1] ## velocidad x del RPA 8
matrizvelocidadglobal[1,16]<-matrizvelocidad[8,2] ## velocidad y del RPA 8
```

```
## ASIGNACION VELOCIDAD NOMINAL AL ENJAMBRE
for(jindice in 1:nRPAs)
{
	for(jdimension in 1:dimension)
	{
		matrizvelocidad[jindice,jdimension]<-
velocidadcru*matrizvelocidad[jindice,jdimension]
	} ## fin del for jdimension
} ## fin del for jindice

## DECLARACIONES DE MATRICES Y VECTORES DEL ALGORITMO EPSO
## factores de decision
(vectorfact[1]=factMmmm,vectorfact[2]=factdecv,vectorfact[3]=factdeca)
matrizposicionfact<-matrix(0,nparticulas,nfact) ## posicion de la nube de
particulas
vectorentropia<-numeric(length=nparticulas) ## vector de entropia de la nube
de particulas

matrizmejorposicionfact<-matrix(0,nparticulas,nfact) ## mejor posicion de la
nube de particulas
vectorentropiamejor<-numeric(length=nparticulas) ## vector de entropia de la
mejor posicion de la nube de particulas

matrizvelocidadfact<-matrix(0,nparticulas,nfact) ## velocidad de la nube de
particulas

vectorfactopt<-numeric(length=nfact) ## vector de factores de decision que
almacena el valor optimo
```

entropiaopt<-0 ## entropia correspondiente al vector de factores de decision que almacena el optimo

matrizposicionfactmem<-matrix(0,nparticulas,nfact) ## memoria de la posicion de la nube de particulas

matrizvelocidadfactmem<-matrix(0,nparticulas,nfact) ## memoria de la velocidad de la nube de particulas

vectorfactmax<-numeric(length=nfact) ## vector de factores de decision maximos

vectorfactmin<-numeric(length=nfact) ## vector de factores de decision minimos

INICIALIZACIONES

matrizposicionfact[]<-0 ## inicializa a 0

vectorentropia[]<-0 ## inicializa a 0

matrizmejorposicionfact[]<-0 ## inicializa a 0

vectorentropiamejor[]<-0 ## inicializa a 0

matrizvelocidadfact[]<-0 ## inicializa a 0

vectorfactopt[]<-0 ## inicializa a 0

entropiaopt[]<-0 ## inicializa a 0

matrizposicionfactmem[]<-0 ## inicializa a 0

matrizvelocidadfactmem[]<-0 ## inicializa a 0

vectorfactmax[]<-0 ## inicializa a 0

vectorfactmin[]<-0 ## inicializa a 0

vectorfactmax[1]<-factMmax ## vector de factores de decision maximos

```
vectorfactmax[2]<-factmax ## vector de factores de decision maximos
vectorfactmax[3]<-factmax ## vector de factores de decision maximos

vectorfactmin[1]<-factMmin ## vector de factores de decision minimos
vectorfactmin[2]<-factmin ## vector de factores de decision minimos
vectorfactmin[3]<-factmin ## vector de factores de decision minimos

## GENERACION DE LA NUBE DE PARTICULAS (factMmmm>=factdecv>=factdeca)
for(jparticula in 1:nparticulas) ## posicion de las particulas
{ ## genera tres numeros aleatorios
    matrizposicionfact[jparticula,1]<-runif(1,vectorfactmin[1],vectorfactmax[1])
    matrizposicionfact[jparticula,2]<-
runif(1,vectorfactmin[2],min(vectorfactmax[2],matrizposicionfact[jparticula,1]))
    matrizposicionfact[jparticula,3]<-
runif(1,vectorfactmin[3],min(vectorfactmax[3],matrizposicionfact[jparticula,2]))
} ## fin del for jparticula

matrizmejorposicionfact<-matrizposicionfact

for(jparticula in 1:nparticulas)
{ ## velocidad de las particulas
    matrizvelocidadfact[jparticula,1]<-runif(1,((matrizposicionfact[jparticula,2]-
matrizposicionfact[jparticula,1])/2),(vectorfactmax[1]-
matrizposicionfact[jparticula,1]))
    matrizvelocidadfact[jparticula,2]<-runif(1,((matrizposicionfact[jparticula,3]-
matrizposicionfact[jparticula,2])/2),((matrizposicionfact[jparticula,1]-
matrizposicionfact[jparticula,2])/2))
```

```
        matrizvelocidadfact[jparticula,3]<-runif(1,-(matrizposicionfact[jparticula,3]-
vectorfactmin[3]),((matrizposicionfact[jparticula,2]-
matrizposicionfact[jparticula,3])/2))
} ## fin del for jparticula

## mejor posicion global inicial en el proceso de iteracion
mejorparticula<-round(runif(1,1,nparticulas))
vectorfactopt[]<-matrizposicionfact[mejorparticula,]


## EJECUTA niteraciones
for(jiteraciones in 1:niteraciones)
{ ## Ejecuta niteraciones

for(jparticula in 1:nparticulas)
{ ## para cada particula

        for(jfact in 1:nfact)
        { ## para cada dimension
                r1pso<-runif(1,0,1)
                r2pso<-runif(1,0,1)
                ## actualizar la velocidad de la partícula
                matrizvelocidadfactmem[jparticula,jfact]<-
wpso*matrizvelocidadfact[jparticula,jfact]+c1pso*r1pso*(matrizmejorposicionfact
[jparticula,jfact]-
matrizposicionfact[jparticula,jfact])+c2pso*r2pso*(vectorfactopt[jfact]-
matrizposicionfact[jparticula,jfact])
        } ## fin del for jfact
```

```
factorreduccion<-1
while ((matrizvelocidadfactmem[jparticula,1]>(vectorfactmax[1]-matrizposicionfact[jparticula,1]))||(matrizvelocidadfactmem[jparticula,1]<((matrizposicionfact[jparticula,2]-matrizposicionfact[jparticula,1])/2)))
{
    factorreduccion<-factorreduccion*factorcontraccion
    matrizvelocidadfactmem[jparticula,1]<-matrizvelocidadfactmem[jparticula,1]*factorreduccion
} ## fin del bucle while ((matrizposicionfact[jparticula

factorreduccion<-1
while ((matrizvelocidadfactmem[jparticula,2]>((matrizposicionfact[jparticula,1]-matrizposicionfact[jparticula,2])/2))||(matrizvelocidadfactmem[jparticula,2]<((matrizposicionfact[jparticula,3]-matrizposicionfact[jparticula,2])/2)))
{
    factorreduccion<-factorreduccion*factorcontraccion
    matrizvelocidadfactmem[jparticula,2]<-matrizvelocidadfactmem[jparticula,2]*factorreduccion
} ## fin del bucle while ((matrizposicionfact[jparticula

factorreduccion<-1
while ((matrizvelocidadfactmem[jparticula,3]>((matrizposicionfact[jparticula,2]-matrizposicionfact[jparticula,3])/2))||(matrizvelocidadfactmem[jparticula,3]< -(matrizposicionfact[jparticula,3]-vectorfactmin[3])))
{
    factorreduccion<-factorreduccion*factorcontraccion
```

```
        matrizvelocidadfactmem[jparticula,3]<-
matrizvelocidadfactmem[jparticula,3]*factorreduccion
} ## fin del bucle while ((matrizposicionfact[jparticula

matrizvelocidadfact[jparticula,]<-matrizvelocidadfactmem[jparticula,]

matrizposicionfact[jparticula,]<-
matrizposicionfact[jparticula,]+matrizvelocidadfact[jparticula,]

factMmmm<-matrizposicionfact[jparticula,1]
factdecv<-matrizposicionfact[jparticula,2]
factdeca<-matrizposicionfact[jparticula,3]

tiempo<-0
tiempoadimen<-1
entropiatiempo<-0 ## inicializa a 0 en cada ciclo de integracion


## INTEGRACION DE LA POSICION DE LOS RPAS EN CADA MISION SEAD
ISR
tiempo<-tiempo+pasotiempo
while (tiempo<=tiempoSEAD)
{ ## while

        ## INTEGRACION DE LA POSICION DE CADA RPA
        for(jindice in 1:nRPAs)
        {
```

```
## CALCULO DE LA DISTANCIA MINIMA DE CADA RPA AL RESTO DEL ENJAMBRE
distanciaRPAmin[]<-1000000000000 ## inicializa a infinito
for(jindice1 in 1:nRPAs)
{
    distanciaRPAcentroide[jindice1]<-sqrt((matrizposicion[jindice1,1]-matrizposicioncentroide[1])^2+(matrizposicion[jindice1,2]-matrizposicioncentroide[2])^2)

    for(jindice2 in 1:nRPAs)
    {
        if (jindice2!=jindice1)
        {
            distanciaRPAminima<-sqrt((matrizposicion[jindice1,1]-matrizposicion[jindice2,1])^2+(matrizposicion[jindice1,2]-matrizposicion[jindice2,2])^2)
            if (distanciaRPAminima<=distanciaRPAmin[jindice1])
            {
                distanciaRPAmin[jindice1]<-distanciaRPAminima
                indiceRPAdistanciamin[jindice1]<-jindice2
            } ## fin del if (distanciaRPAminima
        } ## fin del if (jindice2!=jindice1)
    } ## fin del for jindice2
```

```
                matrizRPAdistanciamin[jindice1,1]<-
(matrizposicion[indiceRPAdistanciamin[jindice1],1]-
matrizposicion[jindice1,1])/distanciaRPAmin[jindice1]
                matrizRPAdistanciamin[jindice1,2]<-
(matrizposicion[indiceRPAdistanciamin[jindice1],2]-
matrizposicion[jindice1,2])/distanciaRPAmin[jindice1]
                matrizdistanciaRPAcentro[jindice1,1]<-
(matrizposicioncentroide[1]-
matrizposicion[jindice1,1])/distanciaRPAcentroide[jindice1]
                matrizdistanciaRPAcentro[jindice1,2]<-
(matrizposicioncentroide[2]-
matrizposicion[jindice1,2])/distanciaRPAcentroide[jindice1]

        } ## fin del for jindice1

        modulovectorvelocidad<-
sqrt(matrizvelocidad[jindice,1]^2+matrizvelocidad[jindice,2]^2)

        ## angulo del vector velocidad
        angulovectorvelocidad<-
atan(matrizvelocidad[jindice,2]/matrizvelocidad[jindice,1])
        if (matrizvelocidad[jindice,1]<=0)
        {
                angulovectorvelocidad<-angulovectorvelocidad+pi
        } ## fin del if (matrizvelocidad[jindice,1]

        ## EXISTE RIESGO DE COLISION
```

```
if (distanciaRPAmin[jindice]<=factdecv*tam)
{ ## EL RPA GIRA EL VECTOR VELOCIDAD PARA EVITAR LA COLISION

## angulo del vector que apunta al RPA mas cercano
angulovectorRPAdismin<-atan(matrizRPAdistanciamin[jindice,2]/matrizRPAdistanciamin[jindice,1])
if (matrizRPAdistanciamin[jindice,1]<=0)
{
angulovectorRPAdismin<-angulovectorRPAdismin+pi
} ## fin del if (matrizRPAdistanciamin[jindice,1]

angulovectores<-(angulovectorvelocidad-angulovectorRPAdismin)
signoangulovectores<-abs(angulovectores)/angulovectores

if (abs(angulovectores)<pi)
{
angulovectorvelocidad<-angulovectorvelocidad+signoangulovectores*heta*pasotiempo
} ## fin del if (abs(angulovectores)<pi)
else ## angulovectores>=pi
{
angulovectorvelocidad<-angulovectorvelocidad-signoangulovectores*heta*pasotiempo
} ## fin del else ## angulovectores>=pi
```

```
                    ## EL RPA SE DECELERA PARA EVITAR LA COLISION
                    if (distanciaRPAmin[jindice]<=factdeca*tam)
                    {
                        modulovectorvelocidad<-modulovectorvelocidad-
deceleracion*pasotiempo
                    } ## fin del if (distanciaRPAmin[jindice] factdeca

                    if (modulovectorvelocidad<=velocidadper)
                    { ## el RPA no se decelera por debajo de la velocidad de
perdida
                        modulovectorvelocidad<-velocidadper
                    } ## fin del if (modulovectorvelocidad

                } ## fin del if (distanciaRPAmin[jindice] factdecv
                else ## NO HAY RIESGO DE COLISION
                { ## EL RPA ESTA MUY ALEJADO

                    if (distanciaRPAcentroide[jindice]>=factMmmm*tam)
                    { ## EL RPA GIRA EL VECTOR VELOCIDAD PARA
VOLVER AL ENJAMBRE
                        ## angulo del vector que apunta al centroide

                        angulovectorRPAcentro<-
atan(matrizdistanciaRPAcentro[jindice,2]/matrizdistanciaRPAcentro[jindice,1])

                        if (matrizdistanciaRPAcentro[jindice,1]<=0)
                        {
```

```
                                angulovectorRPAcentro<-
angulovectorRPAcentro+pi
                        } ## fin del if (matrizdistanciaRPAcentro[jindice,1]

                        angulovectores<-(angulovectorvelocidad-
angulovectorRPAcentro)
                        signoangulovectores<-
abs(angulovectores)/angulovectores

                        if (abs(angulovectores)<pi)
                        {
                        angulovectorvelocidad<-angulovectorvelocidad-
signoangulovectores*heta*pasotiempo
                        } ## fin del if (abs(angulovectores)<pi)
                        else ## angulovectores>=pi
                        {
                                angulovectorvelocidad<-
angulovectorvelocidad+signoangulovectores*heta*pasotiempo
                        } ## fin del else ## angulovectores>=pi

                } ## fin del if (matrizdistanciaRPAcentro[jindice]
                ## EL RPA SE ACELERA

                if (modulovectorvelocidad<velocidadcru)
                {
                        modulovectorvelocidad<-
modulovectorvelocidad+aceleracion*pasotiempo
                } ## fin del if (modulovectorvelocidad
```

```
                    if (modulovectorvelocidad>=velocidadcru)
                    { ## el RPA no se acelera por encima de la velocidad de
crucero
                        modulovectorvelocidad<-velocidadcru
                    } ## fin del if (modulovectorvelocidad
                } ## fin del else ## NO HAY RIESGO DE COLISION

        matrizvelocidad[jindice,1]<-
modulovectorvelocidad*cos(angulovectorvelocidad) ## velocidad x
        matrizvelocidad[jindice,2]<-
modulovectorvelocidad*sin(angulovectorvelocidad) ## velocidad y
        matrizposicion[jindice,1]<-
matrizposicion[jindice,1]+matrizvelocidad[jindice,1]*pasotiempo ## posicion x
        matrizposicion[jindice,2]<-
matrizposicion[jindice,2]+matrizvelocidad[jindice,2]*pasotiempo ## posicion y

        } ## fin del for jindice

## CALCULO DE LA ENTROPIA DEL ENJAMBRE A
velocidadminima[]<-1000000000000 ## inicializa a infinito
velocidadmaxima[]<--1000000000000 ## inicializa a -infinito
contadormicroestados[]<-0 ## inicializa a 0
indiceentropia[]<-0 ## inicializa a 0

for(jdimension in 1:dimension)
{
```

```
        for(jindice in 1:nRPAs)
        {
                if
(matrizvelocidad[jindice,jdimension]<=velocidadminima[jdimension])
                { ## calculo de la velocidad minima
                        velocidadminima[jdimension]<-
matrizvelocidad[jindice,jdimension]
                } ## fin del if (matrizvelocidad[jindice,


                if
(matrizvelocidad[jindice,jdimension]>=velocidadmaxima[jdimension])
                { ## calculo de la velocidad maxima
                        velocidadmaxima[jdimension]<-
matrizvelocidad[jindice,jdimension]
                } ## fin del if (matrizvelocidad[jindice,
        } ## fin del for jindice
} ## fin del for jdimension


for(jdimension in 1:dimension)
{
        deltavelocidad[jdimension]<-(velocidadmaxima[jdimension]-
velocidadminima[jdimension])/nRPAs
} ## fin del for jdimension


for(jdimension in 1:dimension)
{
        for(jindice1 in 1:nRPAs)
```

```
        { ## contador de matrizvelocidad
                for(jindice2 in 1:(nRPAs+1))
                {
                        if
((matrizvelocidad[jindice1,jdimension]>=(velocidadminima[jdimension]+(jindice2
-
1)*deltavelocidad[jdimension]))&&(matrizvelocidad[jindice1,jdimension]<(velocid
adminima[jdimension]+jindice2*deltavelocidad[jdimension])))
                        {
                                contadormicroestados[jindice2,jdimension]<-
contadormicroestados[jindice2,jdimension]+1
                        } ## fin del if (matrizvelocidad[jindice1,
                } ## fin del for jindice2
        } ## fin del for jindice1
} ## fin del for jdimension

contadormicroestados[nRPAs,1]<-
contadormicroestados[nRPAs,1]+contadormicroestados[(nRPAs+1),1]
contadormicroestados[nRPAs,2]<-
contadormicroestados[nRPAs,2]+contadormicroestados[(nRPAs+1),2]
contadormicroestados[(nRPAs+1),]<-0

for(jdimension in 1:dimension)
{
        for(jindice2 in 1:nRPAs)
        {
                if (contadormicroestados[jindice2,jdimension]>0)
                {
```

```
                    indiceentropia[jindice2,jdimension]<-1
                } ## fin del if (matrizvelocidad[jindice2
        entropiatiempo<-entropiatiempo+indiceentropia[jindice2,jdimension]
        } ## fin del for jindice2
} ## fin del for jdimension

## MAGNITUDES GLOBALES
tiempoadimen<-tiempoadimen+1


## posicion global
matrizposicionglobal[tiempoadimen,1]<-matrizposicion[1,1] ## posicion x del RPA 1
matrizposicionglobal[tiempoadimen,2]<-matrizposicion[1,2] ## posicion y del RPA 1
matrizposicionglobal[tiempoadimen,3]<-matrizposicion[2,1] ## posicion x del RPA 2
matrizposicionglobal[tiempoadimen,4]<-matrizposicion[2,2] ## posicion y del RPA 2
matrizposicionglobal[tiempoadimen,5]<-matrizposicion[3,1] ## posicion x del RPA 3
matrizposicionglobal[tiempoadimen,6]<-matrizposicion[3,2] ## posicion y del RPA 3
matrizposicionglobal[tiempoadimen,7]<-matrizposicion[4,1] ## posicion x del RPA 4
matrizposicionglobal[tiempoadimen,8]<-matrizposicion[4,2] ## posicion y del RPA 4
```

```
matrizposicionglobal[tiempoadimen,9]<-matrizposicion[5,1] ## posicion x del RPA 5

matrizposicionglobal[tiempoadimen,10]<-matrizposicion[5,2] ## posicion y del RPA 5

matrizposicionglobal[tiempoadimen,11]<-matrizposicion[6,1] ## posicion x del RPA 6

matrizposicionglobal[tiempoadimen,12]<-matrizposicion[6,2] ## posicion y del RPA 6

matrizposicionglobal[tiempoadimen,13]<-matrizposicion[7,1] ## posicion x del RPA 7

matrizposicionglobal[tiempoadimen,14]<-matrizposicion[7,2] ## posicion y del RPA 7

matrizposicionglobal[tiempoadimen,15]<-matrizposicion[8,1] ## posicion x del RPA 8

matrizposicionglobal[tiempoadimen,16]<-matrizposicion[8,2] ## posicion y del RPA 8


## velocidad global

matrizvelocidadglobal[tiempoadimen,1]<-matrizvelocidad[1,1] ## velocidad x del RPA 1

matrizvelocidadglobal[tiempoadimen,2]<-matrizvelocidad[1,2] ## velocidad y del RPA 1

matrizvelocidadglobal[tiempoadimen,3]<-matrizvelocidad[2,1] ## velocidad x del RPA 2

matrizvelocidadglobal[tiempoadimen,4]<-matrizvelocidad[2,2] ## velocidad y del RPA 2

matrizvelocidadglobal[tiempoadimen,5]<-matrizvelocidad[3,1] ## velocidad x del RPA 3
```

```
matrizvelocidadglobal[tiempoadimen,6]<-matrizvelocidad[3,2] ## velocidad y del RPA 3

matrizvelocidadglobal[tiempoadimen,7]<-matrizvelocidad[4,1] ## velocidad x del RPA 4

matrizvelocidadglobal[tiempoadimen,8]<-matrizvelocidad[4,2] ## velocidad y del RPA 4

matrizvelocidadglobal[tiempoadimen,9]<-matrizvelocidad[5,1] ## velocidad x del RPA 5

matrizvelocidadglobal[tiempoadimen,10]<-matrizvelocidad[5,2] ## velocidad y del RPA 5

matrizvelocidadglobal[tiempoadimen,11]<-matrizvelocidad[6,1] ## velocidad x del RPA 6

matrizvelocidadglobal[tiempoadimen,12]<-matrizvelocidad[6,2] ## velocidad y del RPA 6

matrizvelocidadglobal[tiempoadimen,13]<-matrizvelocidad[7,1] ## velocidad x del RPA 7

matrizvelocidadglobal[tiempoadimen,14]<-matrizvelocidad[7,2] ## velocidad y del RPA 7

matrizvelocidadglobal[tiempoadimen,15]<-matrizvelocidad[8,1] ## velocidad x del RPA 8

matrizvelocidadglobal[tiempoadimen,16]<-matrizvelocidad[8,2] ## velocidad y del RPA 8



tiempo<-tiempo+pasotiempo

} ## fin del while (tiempo<=tiempoSEAD)


vectorentropia[jparticula]<-entropiatiempo
```

```
if (vectorentropia[jparticula]>vectorentropiamejor[jparticula])
{
        matrizmejorposicionfact[jparticula,]<-matrizposicionfact[jparticula,]
        vectorentropiamejor[jparticula]<-vectorentropia[jparticula]
} ## fin del if (vectorentropia[jparticula]

if (vectorentropia[jparticula]>entropiaopt)
{
        vectorfactopt<-matrizposicionfact[jparticula,]
        entropiaopt<-vectorentropia[jparticula]
} ## fin del if (vectorentropia[jparticula]

} ## fin del for jparticula


## ALGORITMO EVOLUTIVO

## selección de la particula menos apta
nparticulaNA <- which.min(vectorentropia) ## la particula menos apta es la que tiene la entropia mas pequeña

## cruce mediante la tecnica de torneo
## eleccion al azar de dos subgrupos de individuos
pcorte <- round(runif(1,1,nparticulas-1))

## eleccion del primer progenitor
```

```
nparticulaprog1 <- which.max(vectorentropia[1:pcorte]) ## primer progenitor

## eleccion del segundo progenitor
nparticulaprog2 <- pcorte+which.max(vectorentropia[(pcorte+1):nparticulas]) ## segundo progenitor

## cruce, combinación de material genético del nuevo individuo
for(jfact in 1:nfact)
{ ## para cada dimension
    matrizvelocidadfactmem[nparticulaNA,jfact]<-runif(1,0,1)*matrizvelocidadfact[nparticulaprog1,jfact]+runif(1,0,1)*matrizvelocidadfact[nparticulaprog2,jfact]
} ## fin del for jfact

## mutacion
if (runif(1,0,1)<errormuta) ## se produce la mutacion si se cumple la condicion
{
    nparticulamut<-round(runif(1,1,nparticulas)) ## eleccion aleatoria de la particula que sufrira la mutacion
    ncoordmut<-round(runif(1,1,nfact)) ## eleccion aleatoria de la coordenada que sufrira la mutación

    if (ncoordmut==1)
    {
        matrizvelocidadfactmem[jparticula,1]<-runif(1,((matrizposicionfact[jparticula,2]-
```

```
matrizposicionfact[jparticula,1])/2),(vectorfactmax[1]-
matrizposicionfact[jparticula,1]))
        } ## fin del if (ncoordmut==1)
        if (ncoordmut==2)
        {
                matrizvelocidadfactmem[jparticula,2]<-
runif(1,((matrizposicionfact[jparticula,3]-
matrizposicionfact[jparticula,2])/2),((matrizposicionfact[jparticula,1]-
matrizposicionfact[jparticula,2])/2))
        } ## fin del if (ncoordmut==2)
        if (ncoordmut==3)
        {
                matrizvelocidadfactmem[jparticula,3]<-runif(1,-
(matrizposicionfact[jparticula,3]-
vectorfactmin[3]),((matrizposicionfact[jparticula,2]-
matrizposicionfact[jparticula,3])/2))
        } ## fin del if (ncoordmut==2)

}

factorreduccion<-1
while           ((matrizvelocidadfactmem[nparticulaNA,1]>(vectorfactmax[1]-
matrizposicionfact[nparticulaNA,1]))||(matrizvelocidadfactmem[nparticulaNA,1]<
((matrizposicionfact[nparticulaNA,2]-matrizposicionfact[nparticulaNA,1])/2)))
{
        factorreduccion<-factorreduccion*factorcontraccion
        matrizvelocidadfactmem[nparticulaNA,1]<-
matrizvelocidadfactmem[nparticulaNA,1]*factorreduccion
```

```
} ## fin del bucle while ((matrizposicionfact[nparticulaNA

factorreduccion<-1
while
((matrizvelocidadfactmem[nparticulaNA,2]>((matrizposicionfact[nparticulaNA,1]-
matrizposicionfact[nparticulaNA,2])/2))||(matrizvelocidadfactmem[nparticulaNA,
2]<((matrizposicionfact[nparticulaNA,3]-matrizposicionfact[nparticulaNA,2])/2)))
{
        factorreduccion<-factorreduccion*factorcontraccion
        matrizvelocidadfactmem[nparticulaNA,2]<-
matrizvelocidadfactmem[nparticulaNA,2]*factorreduccion
} ## fin del bucle while ((matrizposicionfact[nparticulaNA

factorreduccion<-1
while
((matrizvelocidadfactmem[nparticulaNA,3]>((matrizposicionfact[nparticulaNA,2]-
matrizposicionfact[nparticulaNA,3])/2))||(matrizvelocidadfactmem[nparticulaNA,
3]< -(matrizposicionfact[nparticulaNA,3]-vectorfactmin[3])))
{
        factorreduccion<-factorreduccion*factorcontraccion
        matrizvelocidadfactmem[nparticulaNA,3]<-
matrizvelocidadfactmem[nparticulaNA,3]*factorreduccion
} ## fin del bucle while ((matrizposicionfact[nparticulaNA

matrizvelocidadfact[nparticulaNA,]<-matrizvelocidadfactmem[nparticulaNA,]

matrizposicionfact[nparticulaNA,]<-
matrizposicionfact[nparticulaNA,]+matrizvelocidadfact[nparticulaNA,]
```

```
} ## Fin del bucle for jiteraciones
```

Referencias

[1] Alexandros Karatzoglou, David Meyer y Kurt Hornik, Support Vector Machines in R, Journal of Statistical Software, April 2006.

[2] Argel A. Bandala, Elmer P. Dadios, Ryan Rhay P. Vicerra y Laurence A. Gan Lim, Swarming Algorithm for Unmanned Aerial Vehicle (UAV) Quadrotors –Swarm Behavior for Aggregation, Foraging, Formation, and Tracking, De La Salle University, Manila, 2014.

[3] Beatriz A. Garro and Roberto A. Vázquez, Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms, Computational Intelligence and Neuroscience, 2015.

[4] Beatriz Hernández Pajares, Clasificación automática multiclase de tweets y su representación gráfica, Universidad Rey Juan Carlos, 2014.

[5] Benjamín Barán y Augusto Hermosilla, Comparación de un Sistema de Colonias de Hormigas y una Estrategia Evolutiva para el Problema del Ruteo de Vehículos con Ventanas de Tiempo en un Contexto Multiobjetivo, Centro Nacional de Computación, Universidad Nacional de Asunción San Lorenzo, Casilla de Correos 1439 - Paraguay

[6] Casillas, González de Lena y Martínez, Algoritmo de Clustering On Line utilizando Metaheurísticas y Técnicas de Muestreo, XIX Congreso de SEPLN, 2004.

[7] César Adrián Muñoz, Ramón A. Gallego y Eliana Mirledy Toro, Comparación del desempeño computacional de los algoritmos genéticos de Chu-Beasley y colonia de hormigas en la solución del problema de p-

centdiana, Universidad Tecnológica de Pereira, Vereda La Julita, Pereira, Risaralda, Colombia.

[8] Chang-Su Park, Min-Jea Tahk y Hyochoong, Bang Multiple Aerial Vehicle Formation Using Swarm Intelligence, Korea Advanced Institute of Science Technology (KAIST), 2003.

[9] Chen Shi-Ming y Fang Hua-Jing, Modelling And Stability Analysis of Emergent Behavior of Scalable Swarm System, Department of Control Science And Engineering, Huazhong University of Science And Technology, Wuhan, School of Electrical And Electronic Engineering, East China Jiaotong University, Nanchang, 2006.

[10] Cheney, W. y Kincaid, D., Numerical Mathematics and Computing, Brooks Cole, 2004.

[11] Christian Eduardo Macaya Gatica, Implementación de LS-SVM con PSO para el pronóstico de la temperatura superficial del mar en la costa del Norte de Chile, Pontificia Universidad Católica De Valparaíso, 2010.

[12] David Meyer, Support Vector Machines The Interface to libsvm in package e1071, FH Technikum Wien, Austria, 2017.

[13] David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, Friedrich Leisch, Chih-Chung Chang (libsvm C++-code), Chih-Chen Lin (libsvm C++-code), Package “e1071” de r-project, 2017.

[14] De Nardi Renzo, Flocking of UAVs Software model and limited vision simulations, Università degli Studi di Padova, 2004.

[15] Deepali Sharma, Prerna Gaur & A. P. Mittal, Comparative Analysis of Hybrid GAPSO Optimization Technique With GA and PSO Methods for Cost Optimization of an Off-Grid Hybrid Energy System, Energy Technology & Policy An Open Access Journal, Agosto 2014.

[16] Dian Palupi Rini, Siti Mariyam Shamsuddin y Siti Sophiyati Yuhaniz, Particle Swarm Optimization: Technique, System and Challenges, International Journal of Computer Applications, Enero 2011.

[17] Dustin J. Nowak, Exploitation of self organization in UAV swarms for optimization in combat environments, Thesis, Air force institute of technology, Wright Patterson, 2003.

[18] E. W. Justha y P. S. Krishnaprasad, A Simple Control Law for UAV Formation Flying, Institute for Systems Research University of Maryland, 2002.

[19] Enrique Alba Torres, Análisis y Diseño de Algoritmos Genéticos Paralelos Distribuidos, Tesis Doctoral, Universidad de Málaga, 2015.

[20] Enrique Gabriel Baquela y Andrés Redchuk, Optimización Matemática con R, Bubok Publishing, S.L., 2013.

[21] Eva M. García Polo, Técnicas de Localización en Redes Inalámbricas de Sensores, Instituto de Investigación en Informática de Albacete, Universidad de Castilla-La Mancha, 2008.

[22] Félix Hernan Castro Fuentes, Implementación de LS-SVM con PSO para el pronóstico de la temperatura superficial del mar en la costa del Norte de Chile, Pontificia Universidad Católica De Valparaíso, 2010.

[23] Francisco Cucharero Pérez, Balística exterior, Ministerio de Defensa, 1992.

[24] Francisco Cucharero Pérez, Guiado y Control de Misiles, Ministerio de Defensa, 1995.

[25] Francisco de Borja Ibarrondo Hernández, Optimization of Integrated Guidance and Control for a Dual Aerodynamic Control Missile, Tesis Doctoral, UPM, 2015.

[26] Gilles Labonté, Self organization of formations and swarms, Royal Military College, Kingston, Ontario, 2010.

[27] Ginés Rubio Flores, Modelos avanzados de inteligencia computacional para aproximación funcional y predicción de series temporales en arquitecturas paralelas, Tesis Doctoral, Universidad de Granada, 2010.

[28] Guido Maria Cortelazzo, Adrian F. Clark, y John C. Woods, Flocking of UAVs Software Model and Limited Vision Simulations, University of Padova, 2004.

[29] H. Van Dyke Parunak y Sven A. Brueckner, The Cognitive Aptitude of Swarming Agents, Vector Research Center of TTGSI, Ann Arbor, 2017.

[30] Hernández López, Predicción económica con algoritmos genéticos: operadores genéticos versus matriz de transición, Estadística Española, Vol. 46, nº 157, pp. 389-407, 2004.

[31] https://es.wikipedia.org.

[32] https://www.r-project.org/, 2018.

[33] James T. Lotspeich, Distributed control of a swarm of autonomous unmanned aerial vehicles, Thesis, Air force institute of technology, Wright Patterson, 2003.

[34] Janne Karelahti, Kai Virtanen, y Tuomas Raivio, Near-Optimal Missile Avoidance Trajectories via Receding Horizon Control, Journal of Guidance,Control, and Dynamics Vol. 30, No. 5, September–October 2007.

[35] Jesús Ramón Pérez y José Basterrechea, Aplicación de algoritmos genéticos y recocido simulado a la reconstrucción del diagrama de radiación de antenas, Dpto. Ing. Comunicaciones ETSIIT Universidad de Cantabria

[36] Jiabo Wang, Li Liu, Teng Long y Zhu Wang, Three-Dimensional Constrained UAV Path Planning using Modified Particle Swarm Optimization with Digital Pheromones, 3rd International Conference on Engineering Optimization, Rio de Janeiro, Julio 2012.

[37] José de Rugeles y Deiby Leó, Técnicas de localización de nodos inalámbricos mediante redes de sensores, Universidad Militar Nueva Granada, Bogotá, 2013.

[38] Joshua J. Corner, Swarming reconnaissance using unmanned aerial vehicles In A Parallel discrete event simulation, Thesis, Air force institute of technology, Wright Patterson, 2004.

[39] Juan Angel Resendiz Trejo, Las máquinas de vectores soporte para identificación en línea, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México, 2006.

[40] K. Zettl, S. S. Muhammad, C. Chlestil, E. Leitgeb, A. Friedl, N. P. Schmitt y W. Rehm, High bit rate optical wireless systems for swarm unmanned aerial vehicles: A feasibility study, Institute of Broadband Communications, Graz University of Technology, Graz, Austria y EADS Innovation Works, Munich, Germany, 2007.

[41] Kevin M. Milam, Evolution of control programs for a swarm of autonomous unmanned aerial vehicles, Thesis, Air force institute of technology, Wright Patterson, 2004.

[42] Laura García-Castaño Fernández, Guerra Electrónica, Universidad Pontificia de Comillas, Madrid, 2006.

[43] Li Bing, Li Jie y Huang Ke Wei, Modeling and Flocking Consensus Analysis for Large-Scale UAV Swarms, School of Mechatronical Engineering, Beijing Institute of Technology, Beijing, 2013.

[44] López de Haro, Sánchez Martín y Conde Collado, Secuenciación de tareas mediante metaheurísticos, VIII Congreso de Ingeniería de Organización, 2004.

[45] Luis Parraguez y José Eduardo Rengel, Sobre PSO (Particle Swarm Optimization): una implementación paralela y distribuida, Universidad de Oriente, Venezuela, 2015.

[46] Marco Mamei, Franco Zambonelli y Letizia Leonardi, Co-Fields: an Adaptive Approach for Motion Coordination, Università di Modena e Reggio Emilia, 2005.

[47] Marco Mamei, Franco Zambonelli, Letizia Leonardi, Co-Fields: A Physically Inspired Approach to Distributed Motion Coordination, Università di Modena e Reggio Emilia, 2004.

[48] Marco Mamei, Franco Zambonelli, Letizia Leonardi, Co-Fields: Towards a Unifying Model for Swarm Intelligence, Università di Modena e Reggio Emilia, 2002.

[49] Mario A. Muñoz, Jesús A. López y Eduardo F. Caicedo, Inteligencia de enjambres: sociedades para la solución de problemas, Revista ingeniería e investigación, Agosto de 2008.

[50] Melián, Moreno Pérez, J. A. y Moreno Vega J. M., Metaheuristics: A global view, Inteligencia Artificial Revista Iberoamericana de Inteligencia Artificial, nº 19, pp. 7-28, 2003.

[51] Melián, Optimización Metaheurística para la planificación de redes WDM, Tesis Doctoral Universidad de la Laguna, 2003.

[52] Mini, Micro, and Swarming UAVs: a baseline study, Library of Congress - Federal Research Division, Washington D.C., 2006.

[53] Nils J. Nilsson, Inteligencia artificial una nueva síntesis, McGraw Hill, 2000.

[54] Nilton Luiz Queiroz Junior, Luis Gustavo Araujo Rodriguez y Anderson Faustino da Silva, Combining Machine Learning with a Genetic Algorithm to Find Good Complier Optimizations Sequences, 19th International Conference on Enterprise Information Systems, 2017.

[55] Robert W. Chalmers, David H. Scheidt, Todd M. Neighoff, Stephan J. Witwicki y Robert J. Bamberger, Cooperating Unmanned Vehicles, Johns Hopkins University Applied Physics Laboratory, 1st Intelligent Systems Technical Conference, Chicago, 2004.

[56] Santana Sepúlveda, Julio Sergio, El arte de programa en R: un lenguaje para la estadística, Instituto Mexicano de Tecnología del Agua, 2014.

[57] Satyobroto Talukder, Mathematical Modelling and Applications of Particle Swarm Optimization, School of Engineering Blekinge Institute of Technology, Karlskrona Sweden, Febrero 2011.

[58] Shripad Gade y Ashok Joshi, Heterogeneous UAV Swarm System for Target Search in Adversarial Environment, Department of Aerospace Engineering Indian Institute of Technology Mumbai, 2013.

[59] Su-Cheol Han y Hyochoong Bang, Proportional Navigation-Based Optimal Collision Avoidance for UAVs, 2d International Conference on

Autonomous Robots and Agents, December, Palmerston North, New Zealand, 2004.

[60] Tecnologías asociadas a sistemas de enjambres de μUAV, Centro Superior de Estudios de la Defensa Nacional, Ministerio de Defensa, 2012.

[61] Veysel Gazi, Stability Analysis of Swarms, Tesis Doctoral, Universidad Estatal de Ohio, 2002.

[62] Vishal A. Rane, Particle Swarm Optimization (PSO) Algorithm: Parameters Effect and Analysis, International Journal of Innovative Research & Development, Julio 2013.

[63] Walter Mora, Cómo utilizar R en métodos numéricos, Revista digital, Matemática, Educación e Internet (http://tecdigital.tec.ac.cr/revistamatematica/), Vol 16, No 1. Setiembre Febrero 2016.

[64] Yvonne Gala Garcia, Algoritmos SVM para problemas sobre big data, Universidad Autónoma de Madrid, 2013.

[65] Zhang Junbin, Cai Jinyan, Meng Yafeng y Meng Tianzhen, Genetic Algorithm Particle Swarm Optimization Based Hardware Evolution Strategy, WSEAS Transactions on Circuits and Systems, 2014.


CENTRO UNIVERSITARIO DE LA DEFENSA
MDE