Instituto de Computación, Facultad de Ingeniería
Universidad de la República, Montevideo, Uruguay

Informe de Proyecto de Grado

Planificación de vuelo utilizando algoritmos evolutivos

Américo Gaudín
rolando.gaudin@fing.edu.uy

Gabriel Madruga
gabriel.madruga@fing.edu.uy

Carlos Rodríguez
carlos.rodriguez@fing.edu.uy

Marzo de 2020

Supervisores:
Sergio Nesmachnow, Universidad de la República
Santiago Iturriaga, Universidad de la República.

Supervisor externo:
Dr. Grégoire Danoy, Luxembourg Center for Systems Biomedicine (LCSB) of the University of Luxembourg

Planificación de vuelo utilizando algoritmos evolutivos
Gaudín, Américo; Madruga, Gabriel; Rodríguez, Carlos
Informe de Proyecto de Grado
Instituto de Computación - Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay, Marzo de 2020

Agradecimientos

Este proyecto representó un gran esfuerzo, no solo de quienes lo llevamos a cabo, sino de muchas personas que de alguna u otra manera realizaron contribuciones importantes. Por eso queremos agradecer principalmente a nuestros familiares y amigos, por el constante apoyo brindado a lo largo de este proyecto, a los docentes y estudiantes de Facultad de Ingeniería integrantes de proyectos relacionados a drones que se llevaron y llevan a cabo en la facultad, por su cooperación a lo largo del proyecto, a nuestros tutores, por hacer este proyecto posible, y a GEOCOM por la compra de los drones utilizados durante el proyecto.

Resumen

Un dron o *Unmanned Aerial Vehicle* (UAV) es un vehículo aéreo autónomo o controlado remotamente. La utilización de una flota de drones presenta un número importante de ventajas en misiones de vigilancia y reconocimiento en comparación a la utilización de un único dron. Una flota de drones que colaboran entre sí permite aumentar el área o reducir el tiempo requerido de una misión. Sin embargo, la coordinación de una flota de drones con un propósito colaborativo introduce múltiples problemas, principalmente cuando los drones deben actuar de forma autónoma. El problema abordado consiste en la generación de rutas para los integrantes de la flota optimizando métricas como el tiempo de vuelo, la cobertura del área y la conectividad entre los drones. A su vez se aborda la utilización de programación orientada a agentes como técnica que permite obtener reacciones en tiempo real de los drones durante la ejecución de la misión.

En este informe se presenta el diseño de algoritmos de inteligencia computacional para la planificación de la movilidad de una flota de drones autónomos utilizando simulación. Se utilizaron algoritmos heurísticos y metaheurísticos para la planificación del movimiento fuera de línea para la flota en su conjunto, y en línea para cada integrante de la misma de forma independiente y descentralizada. Con esta finalidad, cada dron cuenta con un dispositivo de comunicación inalámbrica que le permite intercambiar información con otros drones que se encuentran dentro de su rango de alcance. Cada dron determina su plan de vuelo solamente con la información parcial del plan de vuelo de los drones dentro de su rango de alcance.

Se analizó el problema y se definieron modelos matemáticos considerando diferentes parámetros como la cantidad de drones en la flota, velocidad y rango de alcance de los drones, frecuencia de planificación, etc. Se diseñaron diferentes algoritmos para resolver el problema y se realizaron estudios estadísticos para compararlos y determinar las bondades y debilidades de cada uno de ellos.

La investigación realizada y los algoritmos construidos permitieron la ejecución eficiente de misiones de reconocimiento y vigilancia en simulaciones, realizando una contribución al estudio de problemas de cooperación entre drones.

Para el caso del algoritmo fuera de línea, se lograron mejoras en el fitness de las soluciones de hasta 10 veces comparado con el algoritmo *Mutation or Selection Evolution Strategy* (MOSES). Por otra parte, el algoritmo en línea con reacción es casi dos veces y medio mejor en la detección de objetivos con respecto a la versión sin reacción.

Palabras clave: flota, Dron, UAV, planificación, colaboración, programación orientada a agentes

Índice general

1. Introducción **1**

2. Sistemas aéreos no tripulados **5**
2.1. Descripción general 5
2.2. Tecnologías de drones 6
2.2.1. Tipos de drones y sus características técnicas 6
2.2.2. Tipos de cargas útiles y sus aplicaciones 11
2.2.3. Desarrollos futuros 17
2.3. Componentes del vuelo autónomo 18
2.3.1. Controladores de vuelo y pilotos automáticos 18
2.3.2. Protocolos de comunicación 21
2.4. Drones en Uruguay 21
2.4.1. Empresas destacadas 21

3. Algoritmos **25**
3.1. Heurísticas 25
3.1.1. ¿Por qué algunos problemas son difíciles de resolver? 25
3.1.2. Conceptos básicos de modelado 30
3.2. Algoritmos de planificación 32
3.2.1. Motivación y aplicaciones 33
3.2.2. Ingredientes básicos de la planificación 37
3.3. Planificación de trayectoria y movimiento 39
3.4. Algoritmos evolutivos 41
3.4.1. Algoritmo genético 44
3.4.2. Algoritmo MOSES 45
3.5. Programación orientada a agentes 47
3.5.1. Modelo Belief-Desire-Intention 49
3.5.2. Metodología TROPOS para desarrollo de sistemas AOP 49

4. Presentación del problema y estado del arte **51**
4.1. Presentación del problema 51

4.1.1. Conceptos generales 52
4.1.2. Problema fuera de línea 53
4.1.3. Problema en línea 54
4.2. Relevamiento del estado del arte 55

5. Metodologías de resolución **59**
5.1. Planificación fuera de línea 59
5.1.1. Formulación matemática 59
5.1.2. Diseño de la solución 61
5.1.3. Diseño de los algoritmos evolutivos propuestos 62
5.1.4. Detalles de implementación 62
5.1.5. Criterio de parada 64
5.1.6. Diseño del algoritmo ávido 65
5.2. Planificación en línea 65
5.2.1. Formulación 65
5.2.2. Agente UAV 67
5.2.3. Agente Objetivo 67
5.2.4. Agente Entorno 68
5.2.5. Diseño de la solución 68
5.3. Implementación Vuelo Fuera de Línea 71
5.3.1. Ambiente utilizado 71
5.3.2. Arquitectura de la implementación 72
5.3.3. Descripción de la implementación 72
5.3.4. Compilación de Python 3.7 72
5.3.5. Resultados 72

6. Análisis experimental **73**
6.1. Plataforma de ejecución y desarrollo 73
6.2. Instancias del problema 74
6.3. Planificación fuera de línea 74
6.3.1. Configuración paramétrica 74
6.3.2. Comparación de resultados 83
6.4. Planificación en línea 89
6.4.1. Instancias del problema 89
6.4.2. Calidad de soluciones 91
6.4.3. Comparación de resultados 92

7. Conclusiones y trabajo futuro **93**
7.1. Conclusiones generales 93
7.2. Trabajo futuro 95

Bibliografía **97**

Capítulo 1

Introducción

Los UAV (o como se los suele llamar más comúnmente, drones) son cada vez más utilizados. En los últimos 30 años el número de drones en el mundo ha crecido exponencialmente. En 2016, la *Federal Aviation Administration* (FAA) predijo que la venta de drones pequeños para entretenimiento crecerá de 1.9 millones a 4.3 millones en el 2020. Se espera que las ventas de drones para uso comercial crezcan de 600 mil a 2.7 millones en el 2020. Las ventas de drones, para entretenimiento y para uso comercial combinadas, se esperan que crezcan de 2.5 millones a 7 millones en el 2020. La industria de los drones es una industria de billones de dólares proyectada a crecer a 25 billones en el 2020.[1]

Los drones son vehículos aéreos no tripulados. Los mismos pueden ser radiocontrolados o pueden ser autónomos. Existen drones de muchas formas y tamaños, los cuales pueden realizar tareas diferentes. Usualmente se usan en situaciones donde el vuelo tripulado es considerado demasiado peligroso.

La investigación y el desarrollo de drones ha progresado en gran medida en los últimos años, lo que ha abierto la puerta a la utilización de los mismos en nuevas áreas. En el ámbito de usuarios no profesionales su uso es principalmente recreativo, pero usados correctamente los drones pueden desempeñar un papel importante en ámbitos variados. La mayoría de los departamentos de policía municipales y estatales no pueden permitirse un helicóptero o avión, por lo que sus niveles de vigilancia son limitados. Sin embargo, dado que el precio de ciertos drones útiles para vigilancia es mucho más barato que el costo de un helicóptero o avión, los departamentos de policía podrían comprarlos, haciendo posible observar la tierra desde el aire.

El uso de drones para ver la tierra desde el aire permite realizar diversas tareas de forma más eficiente. Por ejemplo, los drones pueden llevar a cabo operaciones de búsqueda y rescate, combatir incendios, inspeccionar tuberías, rociar cultivos, medir datos meteorológicos, etc. Los drones hacen que estas tareas sean posibles y también las hacen más rentables y seguras. Los drones pueden encontrar todos los peligros mientras el operador está a salvo de los mismos. A medida que se continúan

fabricando y estudiando las tecnologías relacionadas a los drones, estarán disponibles aplicaciones cada vez más útiles. [2]

La mayoría de los drones cuentan con un sistema de control de vuelo que les permite mantener un vuelo estable y realizar movimientos según órdenes. Estos movimientos son ordenados desde tierra por una persona o se realizan siguiendo un plan de vuelo preestablecido, por lo que la mayoría de los drones cuentan con la capacidad de vuelo automatizado básico.

El problema de transformar la especificación de una tarea (proporcionada por humanos) en una descripción de bajo nivel adecuada para controlar los drones, por ejemplo para tareas de vigilancia y reconocimiento, no es sencillo dada la cantidad de variables en juego. En este conexto es que en este proyecto se propone la utilización de técnicas de inteligencia computacional para facilitar la generación de planes de vuelo que cumplan con los intereses de la misión.

Ya que la mayoría de los drones cuentan únicamente con sistemas de tipo automático, i.e. sistemas que permiten seguir un plan de vuelo preestablecido, en este proyecto se propone primeramente una técnica basada en algoritmos evolutivos para encontrar cual es el mejor plan de vuelo. En el caso de drones con un sistema de tipo autónomo, i.e un sistema que permite al dron reaccionar durante el vuelo, el plan de vuelo puede ser extendido con la segunda técnica propuesta en este proyecto, la cual se basa en programación orientada a agentes (AOP).

Las principales contribuciones de este proyecto son: el estudio y la creación de un mecanismo para facilitar la generación de rutas de vuelo para sistemas automáticos utilizando algoritmos evolutivos, así como el estudio e implementación de sistemas autónomos utilizando programación orientada a agentes para la vigilancia y reconocimiento de un área de interés. Este proyecto presenta el estudio de dos algoritmos fuera de línea para la resolución del problema de la planificación estática de rutas para una flota de drones teniendo como objetivos la búsqueda y vigilancia de objetivos y la cobertura de una zona mediante el uso de algoritmos evolutivos. Asimismo, se analiza la pertinencia de utilizar un algoritmo en línea, utilizando la técnica AOP, para mejorar la vigilancia de los objetivos una vez que la misión está en ejecución.

La investigación realizada y los algoritmos construidos permitieron la ejecución eficiente de misiones de reconocimiento y vigilancia en simulaciones, realizando una contribución al estudio de problemas de cooperación entre drones.

Para el caso del algoritmo fuera de línea, se lograron mejoras en el fitness de las soluciones de hasta 10 veces comparado con el algoritmo *Mutation or Selection Evolution Strategy* (MOSES). Por otra parte, el algoritmo en línea con reacción es casi dos veces y medio mejor en la detección de objetivos con respecto a la versión sin reacción.

El trabajo se divide en 7 capítulos. La introducción presenta el contexto, motivación y presentación del problema, descripción de la metodología, descripción de las soluciones, comentarios sobre el análisis experimental y principales contribuciones. El capítulo 2 desarrolla una descripción de los principales conceptos vinculados a los sistemas aéreos no tripulados. Se describen los drones, como se clasifican, que cargas pueden llevar para realizar diferentes tareas, cuales son esas tareas, y que depara el futuro en cuanto a esta tecnología. Se aclara la diferencia entre autonomía y automatización, se expone cuales son los componentes del vuelo autónomo, incluyendo una breve mención a los controladores de vuelo y protocolos de comunicación más populares. A su vez se informa sobre el uso de los drones en el Uruguay. El capítulo 3 expone los diferentes tipos de algoritmos utilizados para resolver problemas de planificación. Explica por qué hay ciertos problemas que son difíciles de resolver, describe qué son los algoritmos de planificación, los algoritmos evolutivos, la programación orientada a agentes y cuales son los mecanismos por los cuales se llega y construyen soluciones adecuadas. El capítulo 4 presenta el problema detalladamente y un resumen del estado del arte. El capítulo 5 presenta la metodología de resolución del problema. El capítulo 6 describe la plataforma de ejecución y desarrollo utilizada, las instancias del problema y el análisis experimental realizado sobre las instancias. Finalmente, el capítulo 7 presenta las conclusiones del proyecto y las principales líneas de trabajo futuro.

Capítulo 2

Sistemas aéreos no tripulados

En este capítulo se desarrolla una descripción de los principales conceptos vinculados a los sistemas aéreos no tripulados. La primera sección habla muy brevemente de que son los drones, menciona las diferentes nomenclaturas para los drones y a que se debe cada una. La segunda sección describe con mayor detalle los diferentes tipos de drones, como se clasifican, que pueden llevar encima para realizar diferentes tareas, cuales son esas tareas y que depara el futuro en cuanto a esta tecnología. En la sección tres se aclara la diferencia entre autonomía y automatización, seguido de una exposición de los componentes del vuelo autónomo, incluyendo una breve mención a los controladores de vuelo y protocolos de comunicación más populares. El capitulo termina con una sección dedicada al uso de los drones en el Uruguay.

2.1. Descripción general

Un vehículo aéreo no tripulado (VANT), UAV (del inglés unmanned aerial vehicle), o comúnmente dron, es una aeronave que vuela sin piloto humano a bordo. Un dron es un vehículo sin tripulación, capaz de mantener de manera automática un nivel de vuelo estable y sostenido. Su vuelo puede ser controlado tanto de forma autónoma, como por una computadora por control remoto.

Dado que la responsabilidad del correcto uso del dron recae en el operador, no es posible remplazarlo por una tecnología a menos que se cuente expresamente con un permiso por las normativas vigentes donde vuele. El termino RPA (del inglés Remotely Piloted Aircraft) es también utilizado, especialmente en ámbitos regulatorios. Por la misma razón, el piloto debe ser capaz de intervenir en el vuelo del dron. Es decir, el operador debe tener la capacidad de tomar el control inmediato de la aeronave no tripulada que vuela haciendo uso de un sistema de vuelo automático [3]. Esto implica que por más que vuelen de forma automática y este habilitado por las normativas vigentes, siempre debe existir un operador monitorizando el vuelo.

Los RPA son un subconjunto de los UAV, ya que pueden haber sistemas de vuelo sin piloto. Dado que utilizar sistemas sin piloto es irresponsable, en este documento utilizaremos el termino UAV, dron o VANT indistintamente, siempre refiriéndonos a un RPA con capacidades de vuelo

autónomo y automático.

2.2. Tecnologías de drones

Los drones se pueden clasificar en términos del tipo (ala fija, multirrotor, etc.), el grado de autonomía, el tamaño, el peso, y la fuente de energía. Estas especificaciones son importantes al momento de seleccionar un dron para una tarea. Y usualmente están relacionadas, como por ejemplo, la duración máxima del vuelo y el límite de carga. Al dron (a veces denominado como la plataforma) se le pueden adjuntar distintos tipos de cargas útiles. Por ejemplo para el transporte de objetos (paquetes de correo, medicamentos, material de extinción de incendios, folletos, etc.) y diferentes tipos de sensores (por ejemplo, cámaras, rastreadores, sensores meteorológicos, etc.).

2.2.1. Tipos de drones y sus características técnicas

Para comprender mejor los drones, es importante analizar sus diferentes características técnicas. En esta sección, se discuten estas características y, para poder visualizar estas características tecnológicas, se describen ejemplos de drones existentes con estas características.

La característica más notable es lo que se suele llamar el tipo de dron, es decir, la tecnología utilizada para mantener el vuelo del dron. Esta característica es también el factor determinante en la forma y el aspecto del dron. Una segunda característica es el nivel de autonomía de vuelo del dron. La autonomía puede variar desde una operación totalmente autónoma hasta un control total por parte de un piloto. Otra característica destacable es la diferencia de tamaño entre drones. El tamaño puede variar desde drones del tamaño de un insecto a drones del tamaño de un avión comercial. El peso también es una característica importante. El peso de los drones puede variar desde varios gramos hasta cientos de kilogramos. La característica final discutida en esta sección es la diferencia en la fuente de energía. Ejemplos de fuentes de energía son baterías eléctricas, celdas solares y, el tradicionalmente utilizado en aviones, combustible.

Principales tipos de drones

Como se indicó anteriormente, una característica técnica importante de los drones es el tipo de dron. Los principales tipos de drones son los de **ala fija** y los **multirrotor**. La mayoría de los drones existentes se pueden definir dentro de estos dos tipos, aunque existen otros, como los sistemas **híbridos** y **ornitópteros**.

Ala fija es un término utilizado principalmente en la industria de la aviación para definir aeronaves que usan alas fijas y estáticas en combinación con la resistencia aerodinámica delantera para generar sustentación (la sustentación es la fuerza, de dirección perpendicular a la de la velocidad de la corriente incidente, generada sobre un cuerpo que se desplaza a través de un fluido). Ejemplos de este tipo de aeronaves son los aviones tradicionales, las cometas, y diferentes tipos de planeadores como parapentes. Incluso un simple avión de papel puede definirse como un sistema de ala fija.

Los sistemas **multirrotor** son el subconjunto de las aeronave de alas giratorias (del ingles rotorcraft). El término rotorcraft se usa en aviación para definir aeronaves que usan alas rotativas para generar sustentación. Un ejemplo popular de una nave de rotor es el helicóptero tradicional. Los drones que utilizan sistemas rotativos casi siempre están equipados con varios rotores pequeños, que son necesarios para su estabilidad, de ahí el nombre de sistemas multirrotor. Comúnmente, estos drones usan al menos cuatro rotores para mantenerse en vuelo. Las diferencias entre los drones de ala fija y los drones multirrotor son importantes dependiendo de la aplicacion para la que se quiera usar el dron. Por ejemplo, los drones multirrotor no necesitan una pista de aterrizaje, hacen menos ruido que sus homólogos de ala fija y pueden flotar en el aire en un punto fijo. Los drones de alas fijas pueden volar más rápido y son más adecuados para distancias largas que sus homólogos multirrotor. Estas características determinan cuál de estos tipos de drones se utilizará para una aplicación específica.

Algunos tipos de drones no se pueden etiquetar como un dron de ala fija o multirrotor. A veces porque simplemente no es de ala fija ni multirrotor, a veces porque tiene características de ambos tipos. Los sistemas **híbridos** son sistemas que tienen características tanto de sistemas multirrotor como de ala fija. Los drones que no son de ala fija ni sistemas multirrotor son mucho menos frecuentes. Un ejemplo de tales características son los ornitóptero. Estos drones vuelan imitando la biomecánica de las alas de los insectos o pájaros. La mayoría de estos tienen el tamaño de las aves o insectos que representan. Estos drones están en su mayoría aún en desarrollo y no son aún muy utilizados en la práctica. Un ejemplos de ornitóptero es el DelFly Explorer, un dron que imita a una libélula [4].

Otros ejemplos de drones que no son de ala fija o multirrotor son los drones que usan motores a reacción. El dron T-Hawk es un ejemplo de este tipo de dron [5]. Por completitud, los globos aerostáticos (llenos de, por ejemplo, aire caliente, helio o hidrógeno) también se mencionan aquí. Los globos aerostáticos no tripulados son un tipo especial de UAV, pero no son vistos comúnmente como drones. Lo mismo ocurre con los cohetes.

Nivel de autonomía

Debido a la ausencia de un piloto a bordo, los drones siempre tienen un cierto nivel de autonomía. Una distinción importante dentro del concepto de autonomía es la diferencia entre sistemas automáticos y autónomos. Un sistema automático es un sistema completamente preprogramado que puede realizar una asignación preprogramada por sí mismo. La automatización también incluye aspectos como la estabilización automática del vuelo. Los sistemas autónomos, por otro lado, pueden lidiar con situaciones inesperadas usando un conjunto de reglas preprogramadas para ayudarles a tomar decisiones. Los sistemas automáticos no pueden ejercer esta libertad de elección. En esta sección y en el proyecto en general, el enfoque se centra en la autonomía de las rutas y operaciones de vuelo en lugar de en la automatización (como la estabilización de vuelo).

El Departamento de Defensa de los Estados Unidos distingue cuatro niveles de autonomía [6]. El nivel más básico de autonomía es un sistema operado por humanos en el cual un operador humano toma todas las decisiones con respecto a la operación del dron. Este sistema no tiene ningún control autónomo sobre su entorno. Un nivel más alto de autonomía es un sistema con delegación. Este

sistema puede realizar muchas funciones independientes del control humano. Puede realizar tareas cuando está delegado para hacerlo, sin aportes humanos. Algunos ejemplos son los controles del motor, los controles automáticos y cualquier otra automatización que debe ser activada o desactivada por un controlador humano. El tercer nivel de autonomía es un sistema supervisado. Este sistema puede realizar varias tareas cuando un humano le da ciertos permisos y direcciones. Tanto el sistema como el supervisor pueden iniciar acciones basadas en datos detectados. Sin embargo, el sistema solo puede iniciar estas acciones dentro del alcance de la tarea actual. El último nivel de autonomía es un sistema totalmente autónomo. Este sistema recibe comandos ingresados por un ser humano y los traduce en tareas específicas sin más interacción humana. En caso de una emergencia, un operador humano puede interferir en estas tareas.

Tamaño y peso

Otras características importantes de un dron son su tamaño y su peso. Muchos países distinguen drones grandes y pequeños (o ligeros y pesados). Por ejemplo, la Inspección de Transporte y Medio Ambiente Humano de los Países Bajos (ILT) hace una distinción entre drones ligeros y drones pesados. Los drones ligeros son drones más livianos que 150 kg y los drones pesados son drones de 150 kg o más [7]. El desarrollo de los drones actualmente se centra en hacer drones más pequeños y ligeros para el público en general. Los drones grandes se utilizan principalmente para fines militares. Por lo tanto, se puede observar un cambio de drones grandes a drones más pequeños. Esto ha llevado a cambios en las categorías de referencia y los parámetros de categoría. Hoy en día se suele utilizar el término dron grande para drones de ala fija de entre 20 y 150 kg y drones multirrotor de entre 25 y 100 kg. Menor a 2 kg son mini, y entre 2 y 25 kg son pequeños.

Fuente de energía

La característica final de los drones discutida aquí es la fuente de energía. Hay cuatro fuentes de energía principales: combustible tradicional de avión, baterías eléctricas, celdas de combustible y células solares. El combustible de avión (queroseno) se usa principalmente en grandes aviones no tripulados de ala fija. Un ejemplo de este tipo de avión no tripulado es el avión militar Predator. Este avión es usado mucho por el ejército de los Estados Unidos y puede equiparse con varios sensores diferentes, pero también con cohetes y otros tipos de municiones.

Las baterías eléctricas se utilizan principalmente en drones multirrotor más pequeños. Estos drones son de corto alcance y tienen menos tiempo de operación que los drones que usan queroseno. Estos drones a menudo son para uso recreativo, por lo que es más práctico que el dron use una batería recargable.

Una celda de combustible es un dispositivo electroquímico que convierte la energía química del combustible directamente en energía eléctrica. Debido a que la conversión está exenta de cualquier proceso térmico o mecánico intermedio, esta conversión es eficiente y amigable con el medio ambiente. Pero actualmente las celdas de combustible se utilizan raramente en drones. Solo los drones de ala fija pueden equiparse con tales células debido al peso relativamente alto de las mismas. Una

gran ventaja de usar una celda de combustible es el hecho de que los drones pueden volar distancias más largas sin necesidad de recargarse. Por ejemplo, el dron Stalker que usa una celda de combustible tiene un tiempo de vuelo de 8 horas en lugar de 2 horas [8].

Los drones que utilizan células solares son raros en la industria de los drones. Son principalmente drones de ala fija, ya que debido a la baja eficiencia de las células solares actuales, estas células generalmente no son adecuadas para los drones multirrotor. Sin embargo, las células solares son adecuadas para pequeños ornitópteros. Los drones de células solares atrajeron mucha atención de los medios cuando tanto Google como Facebook alcanzaron acuerdos con los fabricantes de estos drones [9, 10]. Su objetivo era permitir que los drones que funcionan con energía solar volaran en la atmósfera de forma permanente para permitir que las personas se conecten a Internet a través de los mismos.

Modelos de drones comúnmente usados

Para ilustrar mejor las características de los drones descritas anteriormente, se describen algunos modelos específicos en esta sección. Actualmente, debido a la creciente popularidad de la tecnología de drones, los nuevos modelos se desarrollan a un alto ritmo. Es imposible describir aquí todos los modelos de drones actualmente existentes. Por lo tanto, solo se describen algunos modelos que han estado en los medios de comunicación en cierta medida y modelos que están ampliamente disponibles para los gobiernos, la industria y los ciudadanos. Estos son los modelos de drones ampliamente utilizados, conocidos y disponibles. El orden en que se discuten estos modelos es de pequeño a grande.

Delfly

Los drones Delfly, Micro, Explorer y Nimble son drones que vuelan como insectos, por ejemplo el Explorer vuela como una libélula. Estos drones están siendo desarrollados por la Universidad de Tecnología de Delft en los Países Bajos. Estos pueden despegar y volar de forma completamente autónoma dentro de un entorno cerrado. Pueden evitar obstáculos utilizando cámaras. Sus pesos se miden en gramos y funcionan solamente durante unos minutos debido a las restricciones de tamaño y peso de la batería. En el futuro, estos modelos podrían usarse para reconocimiento y fotografía aérea, pero también para aplicaciones como inspecciones en invernaderos para verificar si la fruta está madura [11].

Hubsan x4 Drone

El Hubsan x4 es un pequeño dron multirrotor desarrollado por la empresa china Hubsan. Este mini dron es bastante simple en diseño y operación. Tiene cuatro rotores y es operado por control remoto. Algunos modelos del dron x4 vienen con una cámara incorporada para tomar fotos y grabar videos. Tiene un peso de 30 gramos, un radio aproximado de alcance de 100 metros y puede funcionar durante 7 minutos con una batería completamente cargada. A diferencia de la mayoría de los otros modelos discutidos, este dron no tiene características avanzadas y está construido principalmente para fines recreativos [12].

Parrot Bebop

La línea de drones Parrot es una línea de drones construidos principalmente para fines recreativos. Cuenta con un sistema multirrotor que puede ser controlado por un teléfono inteligente o tableta. El dron puede funcionar entre 12 y 18 minutos y pesa unos 400 gramos. Su velocidad es de unos 18 kilometros por hora y tiene un alcance de unos 50 metros. Tiene dos cámaras, tecnología Bluetooth y WiFi y utiliza puntos de ruta GPS (waypoints) para volar una ruta preprogramada. El Parrot es similar al Phantom mencionado a continuación, tanto en aplicaciones como en funciones. Además de software de fotografía y vídeo, el dron también está equipado con software de juegos, lo que hace que su énfasis en la recreación sea más claro. [13].

Para las pruebas experimentales realizadas en este proyecto, este es el dron que utilizado. Es un dron atractivo ya que tiene un costo relativamente bajo, y es posible instalar software libre para su control, lo que brinda mayor libertad a la hora de programarlo. Sin embargo las prestaciones de los drones DJI, mencionados a continuación, son mejores.

DJI Mavic Pro y DJI Phantom

Los drones Mavic Pro y Phantom son drones multirrotor con cuatro rotores y están construidos principalmente para fines recreativos. Aunque actualmente las ultimas iteraciones de estos drones cuentan con cargas útiles que los hacen adecuados para otros usos profesionales. Estos drones vienen con una cámara y se controlan a través de un control remoto, pero algunos modelos también permiten controlarlos con un teléfono inteligente a través de una red WiFi. Todos los modelos permiten conectar un teléfono inteligente al control, lo que permite ver en tiempo real la cámara, moverla y hacer fotos o grabar vídeos a través de una aplicación móvil. DJI proporciona una aplicación con varias funcionalidades, incluyendo tomas automáticas y post procesamiento de los contenidos. Además DJI proporciona un kit de desarrollo de software (Software Development Kit, o SDK) para hacer aplicaciones a medida. El Phantom puede volar a unos 60 kilómetros por hora y operar durante unos 30 minutos. Con solo programar la altitud de vuelo y ciertos puntos de ruta (waypoints), el dron puede despegar, aterrizar, hacer grabaciones y regresar automáticamente [14].

senseFly eBee

El senseFly eBee es un dron de ala fija de tamaño pequeño, construido principalmente con fines de mapeo. Es capaz de capturar fotografías aéreas de alta resolución que se pueden transformar en orto-mosaicos (mapas) y modelos 3D precisos. El eBee puede cubrir hasta 12 kilómetros cuadrados en un solo vuelo de mapeo automatizado, mientras que los vuelos sobre áreas más pequeñas, a altitudes más bajas, pueden adquirir imágenes con una distancia de muestreo en el suelo de hasta 1,5 centímetros por píxel. Es reconocido como el dron de mapeo más fácil de usar en el mercado, contiene todo lo necesario para comenzar a mapear: cámara RGB, baterías, radio módem, programas informáticos, y la empresa brinda servicios en la nube para post procesamiento de las imágenes. El eBee Classic pesa solo 700 gramos, minimizando enormemente su energía cinética. Además, el piloto automático maneja una amplia gama de comportamientos inteligentes a prueba de fallas [15].

Raven

El Raven es un dron de alas fijas desarrollado en 2002. El dron fue desarrollado originalmente para el Ejército de los EE.UU., pero también es usado frecuentemente por muchos otros países, lo que lo

convierte en uno de los drones más utilizados en el mundo [16]. El objetivo principal del Raven es la vigilancia y se puede controlar de forma remota o programada para un funcionamiento autónomo. El Raven tiene un ancho de 1,4 m, pesa aproximadamente 2 kg y puede permanecer operativo durante 60 a 90 minutos dentro de un rango de 10 km. Está equipado con una cámara óptica y una cámara de infrarrojos. Al igual que los modelos de aviones regulares, el Raven puede despegar lanzándolo al aire. Aterriza deslizándose hacia un lugar de aterrizaje predefinido y compensando la fuerza del impacto de golpear el suelo desensamblándose [17].

ScanEagle

El ScanEagle es un avión no tripulado de ala fija que data de 2004 y se utiliza principalmente como herramienta de vigilancia. Está equipado con una cámara óptica y / o infrarroja y puede funcionar durante más de 20 horas. Tiene 3,1 metros de ancho, 1,2 metros de largo, pesa 18 kilogramos y tiene una velocidad de crucero de 89 kilómetros por hora. El dron puede ser lanzado por presión neumática y puede aterrizar con un sistema de gancho (skyhook), que lo baja del aire. Por lo tanto, una tira de aterrizaje no es necesaria. Al contrario de la mayoría de los drones de ala fija, el ScanEagle necesita poco espacio para despegar o aterrizar [18].

En esta sección, se describieron una serie de características principales de los drones para determinar las principales diferencias entre los mismos y sus propiedades técnicas. Estas características se muestran esquemáticamente en la Tabla 2.1. La Tabla 2.2 tiene imágenes de los distintos drones.

2.2.2. Tipos de cargas útiles y sus aplicaciones

Esta sección tratará los tipos de carga útil que se pueden adjuntar a los drones. Prácticamente todos los tipos de carga útil se pueden adjuntar a los drones, las únicas restricciones suelen ser el peso y el tamaño de la carga útil. La mayoría de los drones están equipados con cámaras de su fabricante. Se pueden solicitar otras cargas útiles a los fabricantes, como es el caso de DJI que ofrece varias opciones para drones como el Matrice 210 o Mavic Pro 2. En esta sección, distinguiremos entre sensores y otros tipos de carga útil. También describiremos algunas aplicaciones para estas cargas útiles.

Sensores

El peso, el modelo y la fuente de energía de un dron son factores importantes que influyen en su altitud máxima, duración de vuelo, rango de vuelo y carga útil máxima. Una categoría importante de carga útil son los sensores. La mayoría de los drones están equipados con cámaras. Las cámaras y los micrófonos son las cargas útiles más utilizadas para los drones y, a menudo, son estándar cuando se compra uno. Las cámaras pueden ser cámaras normales pero también infrarrojas. Dichas cámaras pueden permitir la visión nocturna y la detección de calor. Otros sensores incluyen sensores biológicos que pueden rastrear microorganismos, sensores químicos que pueden medir composiciones químicas y rastros de sustancias químicas particulares, incluidas partículas radioactivas y sensores meteorológicos que pueden medir el viento, la temperatura, la humedad, etc.

Los sensores mencionados anteriormente son sensores exteroceptivos. Las mediciones obtenidas por estos sensores se utilizan principalmente para determinar la configuración del ambiente y son el

Tabla 2.1: Características generales de algunos drones

Características	Modelo de dron						
	Delfy	Hubsan x4	Bebop 2	Phantom 4	ebee	Raven	ScanEagle
Tipo							
Ala fija	-	-	-	-	✓	✓	✓
Multicóptero	-	✓	✓	✓	-	-	-
Otro	✓	-	-	-	-	-	-
Autonomía							
Operado por humano	-	✓	✓	✓	✓	✓	✓
Con delegación	-	-	✓	✓	✓	✓	✓
Sistema supervisado	✓	-	✓	✓	-	-	-
Autónomo	-	-	-	-	-	-	-
Tamaño/Peso							
Grande (25-150 kg)	-	-	-	-	-	-	-
Pequeño (2-25 kg)	-	-	-	-	-	✓	✓
Mini (hasta 2 kg)	✓	✓	✓	✓	✓	-	-
Fuente de energía							
Combustible de avión	-	-	-	-	-	-	✓
Baterías electricas	✓	✓	✓	✓	✓	✓	-
Celdas de combustible	-	-	-	-	-	-	-
Celdas solares	-	-	-	-	-	✓	-

tipo de sensores que se describen con mayor detenimiento en esta sección, dado que el uso principal del otro tipo de sensores, mencionados a continuación, están ligados a la orientación y posición del dron.

El otro tipo de sensor son los los sensores propioceptivos, cuyas mediciones están principalmente influenciadas por el estado del robot. Ejemplo de este tipo de sensores son las brújulas, *Global Positioning System* (GPS) e *Inertial Measurement Unit* (IMU). Las IMU contienen tres sistemas: acelerómetros de 3 ejes, que miden la aceleración del sensor, un giroscopio de 3 ejes, que miden la velocidad angular del sensor y un magnetómetro de 3 ejes que mide la dirección del campo magnético local. En este trabajo se considera que las mediciones de los sensores son ideales, despreciando los posibles errores de medición.

Las cámaras pueden ser útiles para la prevención, la investigación criminal, el enjuiciamiento penal y la sentencia de conductas delictivas. La mayoría de las aplicaciones asumen que los drones son cámaras de vigilancia voladoras. La función preventiva de las cámaras de vigilancia, incluyendo los drones, probablemente será muy limitada cuando no haya al menos un número sustancial de drones en el cielo. Sin embargo, incluso con una gran cantidad de drones, la función preventiva puede

estar limitada, como ocurre con la vigilancia normal de la cámara [19]. A menudo se asume que las imágenes de monitoreo en vivo o la revisión de las imágenes de la cámara después de un crimen pueden ser útiles para reconstruir los incidentes o para rastrear, arrestar y procesar a los perpetradores. La revisión de imágenes puede proporcionar información útil para resolver el crimen, por ejemplo, para rastrear y arrestar a sospechosos, para excluir a sospechosos potenciales, identificar testigos, encontrar personas extraviadas, reconstruir incidentes y encontrar objetos y vehículos robados. Las imágenes recopiladas por las cámaras pueden ser útiles como información de orientación para la policía durante las investigaciones criminales. Los drones también pueden ser útiles para los forenses, ya que pueden usarse para investigar escenas de crímenes sin pisar huellas valiosas. Sin embargo, debido al alto ángulo con el que los drones graban imágenes, no siempre es probable que se puedan reconocer las caras.

Los usos de drones para hacer cumplir la ley no se limitan al uso de cámaras. Otros sensores también pueden proporcionar oportunidades. Por ejemplo, los sensores de calor son muy útiles para detectar el cáñamo que las personas están creciendo en sus áticos. [20] Los sensores químicos pueden ser útiles para detectar rastros de drogas ilegales. Los drones equipados con puntos de acceso WiFi pueden proporcionar pistas sobre la posición de alguien y pueden usarse para interferir teléfonos y redes de computadoras.

En el área de seguridad, los drones son útiles como instrumentos de observación y vigilancia. Webster distingue tres mecanismos [21]: sistemas no activos, en los cuales las cámaras actúan como un elemento de disuasión visual al usar cámaras falsas para crear la ilusión de vigilancia sin monitoreo o almacenamiento real, sistemas reactivos, que cuentan con instalaciones de grabación, almacenamiento y reproducción, y sistemas proactivos con vigilancia en vivo desde una sala de control con instalaciones de grabación, almacenamiento y reproducción, que permiten una respuesta inmediata a los incidentes a medida que ocurren. Los drones se pueden utilizar para los tres tipos de vigilancia. Sin embargo, es poco probable que los ciudadanos se sientan más seguros, como lo ha demostrado la investigación para monitoreo en vivo [22]. De hecho, las personas generalmente no saben si los sistemas de cámaras son proactivos, reactivos o no activos.

Los drones equipados con sensores también pueden proporcionar información útil sobre situaciones particulares, como la presencia de personas o edificios en áreas específicas o hacer reconocimiento de áreas. En caso de desastres o crisis, la información recopilada con drones puede contribuir a mejorar la conciencia situacional. Las áreas remotas o lugares que son difíciles de alcanzar (por ejemplo, debido a los atascos de tráfico), pueden ser fácilmente accesibles para los drones. La posición de mayor altitud de un dron puede proporcionar mejores vistas generales y proporcionar imágenes para reconstrucciones, pruebas y reclamaciones de seguros. En el área de seguridad, los drones también son útiles para fines de gestión de multitudes, por ejemplo, en grandes manifestaciones, festivales de música, juegos deportivos y otros eventos. Los drones con cámaras son útiles para rastrear personas y evaluar posibles problemas. Por ejemplo, los drones pueden proporcionar información de un grupo de manifestantes que se dirigen en una dirección particular o información sobre dos grupos de fanáticos del fútbol que se mueven uno hacia el otro. Los drones pueden ser útiles para proteger a personas importantes, edificios vulnerables (centrales nucleares, puertos, aeropuertos) e infraestruc-

tura crítica (suministro de agua, internet, etc.). En el caso de incendios grandes, los drones pueden proporcionar información sobre el tamaño y el desarrollo del incendio, la liberación de partículas tóxicas y la dirección de los vientos locales. En caso de accidentes con centrales nucleares, los drones pueden rastrear la presencia y diseminación de la radioactividad. La mayoría de estas aplicaciones se centran en los movimientos; identificar individuos es mucho más difícil con los drones.

Para las inspecciones y el mantenimiento de infraestructura, como carreteras, ferrocarriles, molinos de viento, puentes, tuberías, presas y cables de alta tensión, los drones pueden ser una herramienta útil. Las cámaras pueden detectar puntos débiles, erosión o desgaste. El uso de infraestructura y bienes, como el movimiento de vehículos, aeronaves y barcos, se puede monitorear fácilmente. En caso de atascos de tráfico, el tráfico se puede desviar y los datos recopilados se pueden utilizar para posteriores análisis de tráfico. Las tuberías con fugas de gas o agua pueden ser detectadas. Los objetos altos como techos, chimeneas, molinos de viento y cables de redes eléctricas pueden inspeccionarse desde una distancia cercana cuando se usan drones. Nuevos emprendimientos están trabajando en automatizar y hacer más fácil este tipo de actividades, por ejemplo Sterblue desarrolla un sistema que permite crear rutas en tres dimensiones para inspeccionar estructuras. [23]

Los drones con sensores pueden ser útiles para supervisar y hacer cumplir permisos, por ejemplo, permisos para construir una estructura, permisos de estacionamiento y permisos para remover árboles. Controlar el crecimiento de asentamientos, y detectar irregularidades en propiedades. Los drones también pueden ser útiles para propósitos de cartografía y geomapping. Estas son aplicaciones prometedoras del uso de drones en un futuro cercano [24]. Los aviones no tripulados son más baratos que la fotografía aérea desde aviones tripulados y también más baratos que las imágenes satelitales. Dado que los drones pueden acercarse a la superficie, también pueden alcanzar diferentes ángulos y realizar otras mediciones, como el modelado de terreno en 3D, la investigación sobre vegetación y la geomorfología (erosión, actividad sismográfica, etc.). La medición de volúmenes por medio de fotogrametría es particularmente útil en la industria maderera. La fotogrametría también se puede utilizar en arquitectura para obtener modelos tridimensionales de edificios y terrenos [25].

Cuando están equipados con sensores de partículas, los drones son útiles para detectar la emisión de partículas. Se pueden medir las concentraciones y las tasas de emisión de óxidos de azufre, óxidos de nitrógeno y amoníaco. Otros sensores pueden medir la luz, el sonido y la radiación. Estas aplicaciones de drones pueden contribuir al medio ambiente y también son menos contaminantes que las aeronaves tripuladas [26]. Los drones también pueden ayudar a controlar el vertido ilegal de desechos y el transporte de residuos tóxicos. Cuando a determinados animales se les proporcionan etiquetas RFID, los drones pueden rastrear migraciones, biodiversidad, caza furtiva y hábitat. Las imágenes creadas por drones también pueden ser útiles para estimar las poblaciones de animales y rastrear su comportamiento [27]. Los drones con sensores se utilizan actualmente en la agricultura, por ejemplo, para monitorear el crecimiento de cultivos, estimar la biomasa, revisar las malezas y enfermedades de las plantas, y evaluar la calidad y nivel del agua.

El uso de drones en la vigilancia fronteriza es particularmente útil en vastas áreas y áreas de difícil acceso. La vigilancia fronteriza puede prevenir el tráfico de drogas ilegales y la migración ilegal. El gobierno de los Estados Unidos usa drones en la frontera con México [28]. El gobierno

australiano ha anunciado el uso de drones para la vigilancia de la frontera, particularmente para encontrar refugiados en botes. Frontex, la agencia de la Union Europea para la seguridad fronteriza, menciona explícitamente el uso de drones en el establecimiento del sistema de vigilancia de fronteras EUROSUR [29].

En el campo de la cinematografía, la televisión y el entretenimiento, hay grandes posibilidades para los drones. Los drones brindan la oportunidad de tomar fotos a grandes alturas [30]. Durante los Juegos Olímpicos de Invierno de 2014 en Sochi, Rusia los drones se utilizaron para filmar deportistas. Además, los drones son particularmente útiles para proporcionar vistas generales de paisajes, ciudades y edificios. En las películas, las escenas de persecuciones se pueden grabar desde una perspectiva aérea. Los drones pueden llenar el hueco entre las grúas de elevación (con altura limitada) y los helicópteros (con altos costos). En lo que se llama periodismo de drones, los drones permiten a los periodistas cubrir noticias, grandes eventos y actividades policiales.

Los ciudadanos pueden solicitar fácilmente drones pequeños a través de Internet y, por lo general, dichos drones están equipados con cámaras. Las personas usan estas cámaras para grabar o tomar fotografías de sus hogares y su vecindario, a veces solo por diversión, a veces con otros fines, como la prevención de delitos en el vecindario. Otros propósitos recreativos en los que se utilizan drones son la observación de aves, deportistas que se registran a sí mismos y selfies (autorretratos con el fotógrafo) [31].

El uso de drones en la ciencia es también un dominio creciente. Los drones pueden ser útiles para recopilar todo tipo de datos de investigación. Por ejemplo, en meteorología, los drones pueden recopilar datos sobre la humedad, la presión, la temperatura, la fuerza del viento, la radiación, etc. En caso de tornados o huracanes cercanos, las personas pueden ser evacuadas oportunamente. Los drones pueden recopilar datos relevantes en lugares que hasta ahora eran difíciles o costosos de alcanzar, datos que pueden proporcionar nuevos conocimientos científicos, un mayor conocimiento sobre el medio ambiente, la atmósfera y el clima [32]. Tal conocimiento puede mejorar los modelos existentes y proporcionar predicciones más precisas.

Los drones también se están volviendo más comunes en la arqueología [33]. Los drones pueden explorar paisajes a menor costo y con mayor detalle que los satélites. Desde el aire, se pueden observar patrones en los paisajes, por ejemplo, vegetación que indica una antigua carretera o asentamiento. Las imágenes recolectadas por drones también pueden usarse para reconstruir sitios y excavaciones. En geografía, los drones pueden ser útiles para estimar poblaciones, por ejemplo, en barrios marginales. Incluso en los países desarrollados, las poblaciones reales pueden diferir de lo que está oficialmente registrado, ya que puede haber un número significativo de inmigrantes ilegales. Las nuevas tribus en áreas remotas, como en las selvas tropicales del Amazonas, todavía pueden ser descubiertas con el uso de drones. Los drones pueden ser útiles para mapear y monitorear la urbanización y los flujos de tráfico. Los estudios geológicos con drones ya se utilizan para encontrar nuevas fuentes de gas y petróleo [34].

Otras cargas útiles

Además de los sensores descritos en la subsección anterior, otros tipos de cargas útiles se pueden adjuntar a los drones. La mayoría de las cargas útiles que no son sensores involucran llevar algo que debe ser entregado, es decir, correo como cartas y paquetes, medicamentos, comidas, suministros o extintores de incendios. La carga también puede ser ilegal, como los narcóticos y armas de fuego. En otros casos, la carga no está destinada a la entrega, por ejemplo anuncios y puntos de acceso WiFi.

Desde una perspectiva comercial, los drones se consideran interesantes para entregar correo, paquetes y otros cargamentos. Un ejemplo típico sería usarlos para el suministro de plataformas de perforación petrolera o islas remotas. En los Estados Unidos hay especulaciones sobre la entrega de pizzas con drones y en Rusia se han entregado pizzas con drones [35, 36]. En China, los drones entregan pasteles. [37] Y en Brasil Dronfies Labs entregó remeras a clientes en una feria de tecnología [38]. Sin embargo, es probable que estos experimentos sean principalmente interesantes por motivos publicitarios, para llamar la atención sobre una empresa o producto específico, en lugar de una mejora en la eficiencia logística, ya que existen límites obvios en cuanto al tamaño y peso de la carga que pueden transportar los drones pequeños. En los Estados Unidos, Amazon tiene la intención de entregar sus pedidos utilizando drones, pero las autoridades han prohibido su uso [39].

Otra aplicación comercial de los drones es la de volar anuncios. Se pueden adjuntar objetos, carteles, cintas de teletipo y altavoces a los drones para diseminar mensajes de marketing. Uber utilizó drones para realizar una campaña de marketing BTL (Below The Line o "debajo de la línea") de este tipo en México [40].

Como se mencionó anteriormente, los drones en el dominio de seguridad a menudo usan cámaras y otros sensores. Otras cargas útiles útiles incluyen, por ejemplo, materiales de extinción de incendios [41] y parlantes y señales luminosas para fines de control de multitudes [42]. Más controvertido es el uso de drones equipados con armas, gases lacrimógenos, etc [43]. Para operaciones de búsqueda y rescate, los drones pueden usarse para suministrar agua, alimentos, medicamentos y desfibriladores. Las cámaras de infrarrojos pueden ser útiles para encontrar personas perdidas y salvarlas de la hipotermia, la deshidratación, etc. Después de desastres como terremotos o tsunamis, las infraestructuras completas pueden quedar inhabilitadas, pero los drones pueden estar equipados con WiFi para restaurar las redes de comunicación. Altamente controversial es el uso de drones para el asesinato selectivo [44]. El esparcimiento de veneno para ratas fue una técnica efectiva para la eliminación de ratas en la isla Galápagos [45].

Los drones en la agricultura no solo se centran en el monitoreo. En Japón, actualmente el 30 % de los campos de arroz están rociados con drones [46]. Los pesticidas y los fertilizantes se pueden usar en cantidades mínimas por medio de lo que se denomina agricultura de precisión. Los drones son más rápidos, más seguros y menos dañinos que los tractores. También pueden ahuyentar a las aves, plantar semillas e impregnar árboles frutales, [47] aunque estas aplicaciones puede que requieran mucha más precisión de la que actualmente es posible desde una perspectiva tecnológica.

Algunos usan los drones como muestra de su libertad de expresión o para usos más recreativos. Algunos de estos usos incluyen drones equipados con proyectores para difundir noticias o imágenes en paredes [48]. Una forma creativa de usar drones son spaxels (píxeles en el espacio). En esta apli-

cación, los drones equipados con iluminación LED vuelan en el cielo nocturno y dibujan figuras en 3D, algo similar a los fuegos artificiales [49, 50].

2.2.3. Desarrollos futuros

Hay tres áreas a desarrollar a futuro en la tecnología de drones: miniaturización, autonomía y enjambres. El primer desarrollo, la miniaturización, es el desarrollo más incremental. Como en la mayoría de las áreas de robótica, cada nueva generación de drones es un poco más pequeña, más liviana y más económica que la generación anterior. Los nuevos materiales y las baterías más ligeras y eficientes crean mejores compensaciones entre el dron y su rango de vuelo, la altitud máxima y la carga útil máxima. Los límites de la miniaturización son desconocidos. Los drones más pequeños disponibles comercialmente son más o menos del tamaño de las tarjetas de crédito, pero los expertos indican que dentro de unos años podemos esperar drones del tamaño de insectos.

Los drones más baratos y más pequeños también pueden dar como resultado la ubicuidad de los drones. Mientras que los drones ahora pueden ser una visión rara en el cielo, se espera que dentro de unos años haya muchos drones disponibles entre el público en general. Esta expectativa se basa en la velocidad a la que se fabrican y venden los drones. Los drones son regalos populares en los cumpleaños y regalos de navidad para los adolescentes, son populares entre los fotógrafos y deportistas y hay un incremento en la cantidad de pequeñas empresas que ofrecen servicios de drones.

El segundo desarrollo importante es la creciente autonomía de los drones. Los drones a menudo son vistos como aviones de control remoto, pero existen tecnologías que permiten operaciones autónomas, en las que el control remoto por parte de un operador humano está parcialmente excluido por completo. La mayoría de los drones disponibles comercialmente se controlan de forma remota, pero al mismo tiempo ya contienen elementos de autonomía, principalmente software para la estabilización de vuelo. Drones más profesionales ofrecen la posibilidad de preprogramar vuelos. En un futuro cercano, se espera una mayor autonomía en relación al armado de rutas de vuelo, evasión de obstáculos y realización de maniobras evasivas.

El tercer desarrollo importante es el uso de drones en enjambres. La creciente autonomía de los drones permite la cooperación entre drones en los llamados enjambres. El uso de enjambres puede ampliar el alcance, la duración del vuelo y la carga útil máxima para aplicaciones particulares. Por ejemplo, al usar drones en enjambres, un dron puede asumir una tarea de otro dron con una batería agotada. De esta manera, el rango de vuelo se puede extender más allá del rango del primer dron. Los drones que vuelan más allá del alcance de las señales de control o se dañan durante su vuelo pueden ser reemplazados por otros drones. La carga útil pesada en algunos casos puede distribuirse entre varios drones, excediendo la carga útil de un solo dron. Se pueden usar enjambres de drones como redes de sensores. Cuando los drones se usan para seguir a varias personas, puede surgir un problema cuando se separan. Cuando se usan enjambres, cada dron puede seguir a un individuo en lugar de tener que elegir a quién seguir. Una dificultad tecnológica para superar surge del hecho de que los drones en enjambres tienen que comunicarse entre sí además de comunicarse con el control en tierra, lo cual requiere muchos más canales de comunicación.

2.3. Componentes del vuelo autónomo

Para entender los problemas abordados se debe aclarar la diferencia entre autonomía y automatización. Muchas definiciones son posibles, pero aquí se hace foco en la necesidad de hacer elecciones, un requisito común para los sistemas fuera de control directo. Un sistema automatizado no hace elecciones por sí mismo – sigue un script, aunque un script potencialmente sofisticado, en el que todos los posibles cursos de acción ya se han tomado en cuenta. Por lo tanto, para un sistema automatizado, las opciones ya se han hecho y codificado. Por el contrario, un sistema autónomo hace elecciones por sí mismo. Intenta alcanzar sus objetivos localmente, sin intervención humana, incluso cuando se encuentra con incertidumbre o con acontecimientos imprevistos.

Un sistema inteligente autónomo hace elecciones usando mecanismos más sofisticados que otros sistemas. Estos mecanismos a menudo se parecen a los utilizados por los seres humanos. En última instancia, el nivel de inteligencia de un sistema autónomo se juzga por la calidad de las elecciones que hace. Independientemente de los detalles de la implementación, sin embargo, los sistemas autónomos inteligentes son capaces de encontrar soluciones más creativas a problemas ambiguos que los sistemas con autonomía más simple, o sistemas automatizados, que sólo pueden manejar los problemas que se han previsto [51].

Más allá del tamaño, forma y tipo de dron y de su misión, todos los drones deben contar con los siguientes componentes de vuelo autónomo: estimación de estado, control, cartografía y planificación. Para la estimación de estado los drones cuentan con sensores o sistemas de sensores que proporcionan una estimación del vector de estado del vehículo. El vector de estado normalmente se compone de tres coordenadas de posición, tres componentes del vector de velocidad, y entre tres y nueve parámetros que describen la altitud del vehículo. Los sistemas de control y orientación son los que les permiten a los drones maniobrar de una manera consistente con su misión, manipulando sus motores u otros componentes para pasar de un estado a otro.

Estos vehículos deben contar con alguna capacidad básica de cartografiar el ambiente que los rodea. Si no conocen sus alrededores son incapaces de razonar sobre su ambiente y planificar trayectorias seguras dentro del mismo.

Finalmente, dado un conjunto de obstáculos, datos del ambiente y un destino u otros requerimientos, el vehículo debe ser capaz de calcular una trayectoria, un camino seguro para ir de un punto a otro y completar su misión.[52]

2.3.1. Controladores de vuelo y pilotos automáticos

No existe una definición exacta, pero se suele entender por piloto automático a un sistema que permite que un dron vuele automáticamente hacia determinados puntos preconfigurados – también llamados puntos de ruta GPS (o en inglés: waypoints). Mientras que un controlador de vuelo es el dispositivo que mantiene a la aeronave estable. Sin embargo, muchas veces se utilizan estas dos palabras de forma intercambiable. Los controladores de vuelo modernos y sofisticados tienen partes en común en su arquitectura. Podemos dividir su funcionalidad en tres capas distintas, ilustradas en la Figura 2.1

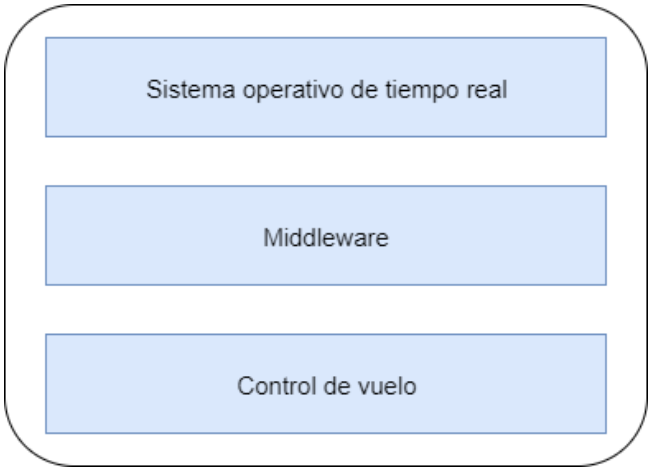


Figura 2.1: Arquitectura de un controlador de vuelo moderno

Capas de los controladores de vuelo

Capa 1: Sistema operativo de tiempo real
El sistema operativo de tiempo real es la columna vertebral del firmware de vuelo, proporcionando abstracción de hardware básico y concurrencia. Los sistemas de tiempo real son críticos para el rendimiento y la seguridad del control de vuelo, ya que garantizan que las tareas de control de vuelo se completarán en un tiempo determinado y son esenciales para la seguridad y el rendimiento de los UAV.
Capa 2: Middleware
El middleware es una colección de herramientas, controladores y bibliotecas que se relacionan con el control de vuelo. Contiene controladores de dispositivo para controlar sensores y otros periféricos. También contiene bibliotecas de control de vuelo tales como protocolos *Radio Control* (RC), utilidades matemáticas y filtros de control.
Capa 3: Control de vuelo
La capa de control de vuelo es el cerebro de la operación. Esta capa contiene todas las rutinas de comando y control. Estimación del estado, control de vuelo, calibración del sistema, telemetría, control del motor, y otros aspectos de control de vuelo residen en esta capa.

Algunos controladores de vuelo

A continuación se da una reseña de los sistemas de controladores de vuelo más utilizados.

LibrePilot

El proyecto LibrePilot de código abierto fue fundado en julio de 2015. Se centra en la investigación y desarrollo de software y hardware para ser utilizado en una variedad de aplicaciones, incluyendo el control y la estabilización de vehículos, vehículos autónomos no tripulados y robótica.

Uno de los objetivos principales del proyecto es proporcionar un ambiente abierto y colaborativo [53].

LibrePilot encuentra sus raíces en el proyecto OpenPilot y los miembros fundadores son colaboradores de larga data en ese proyecto [54]. No es tan popular como los otros controladores de vuelo, por lo que no es tan fácil encontrar hardware especializado.

ArduPilot

ArduPilot fue desarrollado por la comunidad DIYDrones [55] y se basaba en la plataforma Arduino, por eso el nombre comienza con "Ardu" [56].

APM 2.5 y 2.6 son las ediciones más recientes (y finales) del tradicional controlador de vuelo ArduPilot. PX4 comenzó como un proyecto universitario por un equipo suizo que utilizó microcontroladores de ultima generación y procesamiento distribuido para aumentar enormemente las capacidades más allá de APM. DIYDrones y 3DRobotics lo adoptaron como base para su controlador de vuelo de última generación. El Pixhawk es el hardware para controlador de vuelo de última generación hecho en colaboración entre DIYDrones, 3DR y el equipo suizo original que desarrolló PX4 [57].

Paparazzi UAV

Paparazzi UAV es un proyecto de software y hardware de drones de código abierto que abarca sistemas de piloto automático para drones multirrotor, ala fija, helicópteros e híbridos. Paparazzi, al ser de código abierto, permiten desarrollar mas funcionalidades y mejoras al sistema. El uso y la mejora de Paparazzi es alentado por la comunidad. El proyecto incluye código fuente para los controladores aéreos y sistemas de tierra, así como los diseños de hardware para muchas partes del sistema de aviónica. El software de aviónica de Paparazzi está diseñado teniendo en cuenta la portabilidad, permitiendo el uso del sistema de UAV Paparazzi en muchas plataformas de hardware variadas. El software informático de la estación terrestre está escrito en un lenguaje funcional estáticamente tipificado para brindar un sistema más confiable y estable.

A diferencia de otros sistemas, Paparazzi UAV fue diseñado con el vuelo autónomo como enfoque principal y el vuelo manual como el secundario. Desde el principio, fue diseñado para poder controlar múltiples aeronaves dentro del mismo sistema. Paparazzi cuenta con un sistema de plan de vuelo dinámico que se define por los estados de la misión y puntos de ruta. Esto hace que sea fácil crear misiones completamente automatizadas muy complejas sin la intervención de los operadores.

Desde su fundación en 2003, las universidades y los equipos civiles han empleado a Paparazzi para ganar las mejores calificaciones en competiciones de robótica aérea de alto perfil en todo el mundo. Varios desarrolladores centrales del proyecto están afiliados a universidades e instituciones de investigación. Estos incluyen, ENAC University of Toulouse, Francia, MAVlab de TU-Delft en los Países Bajos y AggieAir de Utah State University en los Estados Unidos.

Actualmente, varios proveedores están produciendo y vendiendo controladores de vuelo Paparazzi y accesorios populares, lo que hace que el sistema sea fácil y asequible para todos. Algunas de las compañías fueron fundadas por desarrolladores centrales del proyecto UAV de Paparazzi co-

mo una manera de proporcionar hardware bien probado e integrado a la comunidad. Por ejemplo, 1BitSquared está desarrollando hardware de aviónica y proporcionando servicios de consultoría para la integración de sistemas, el desarrollo de software y el desarrollo de hardware personalizado de drones micro [58].

De software y hardware propietario

Las empresas más conocidas que fabrican drones son DJI y Parrot, ya que 3DR ya no se dedica a la venta, sino a brindar servicios relacionados con drones. Ambas empresas cuentan con un SDK para poder programar sus drones, por ejemplo para que sigan determinadas coordenadas y sacar fotografías o vídeos [59].

2.3.2. Protocolos de comunicación

Los drones de DJI y Parrot cuentan con protocolos de comunicación propietarios, mientras que las demás plataformas (ArduPilot, APM, PX4) utilizan MAVLink. MAVLink, cuyo nombre completo es *Micro Air Vehicle Communication Protocol* es un protocolo diseñado para establecer comunicación con los UAV. Permite empaquetar estructuras sobre canales seriales y enviarlas a una estación de control en tierra. Además brinda a los desarrolladores la libertad de crear mensajes y añadirles códigos específicos [60].

2.4. Drones en Uruguay

En Uruguay el uso de drones es cada vez más frecuente, contando entre sus usuarios al Ministerio del Interior, emprendimientos agropecuarios y de producciones audiovisuales, así como aficionados [61]. La Dirección Nacional de Aviación Civil e Infraestructura Aeronáutica (Dinacia) resolvió reglamentar el uso de los drones. Se obliga a los usuarios a registrar ante el organismo aquellos dispositivos cuya masa supere los 25 kilogramos. Si los drones se utilizan con fines comerciales tanto el dispositivo como su piloto deben estar registrados y la actividad debe ser bajo una empresa registrada para trabajos aéreos. La normativa establece también que todos los usuarios de drones deben pedir autorización para sobrevolar eventos multitudinarios o áreas pobladas. Además se restringe o prohíbe el sobrevuelo de zonas de tráfico aéreo (aeropuertos y aeródromos) [62]. Todo vuelo debe ser realizado manteniendo contacto visual con el dron, como se esquematiza en la Figura 2.2, a menos que se consiga una autorización explicita de Dinacia.

2.4.1. Empresas destacadas

Existen muchas empresas que realizan trabajos con drones. La mayoría son empresas que utilizan los drones con fines agropecuarios o para realizar tomas aéreas que no serían posibles, o serían muy costosas de otra manera. En este documento se hace mención a dos empresas en particular por

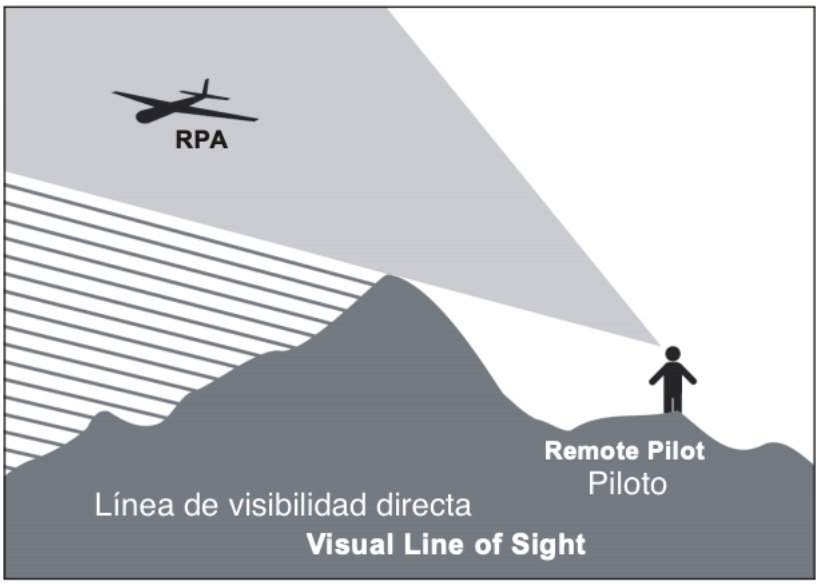


Figura 2.2: Línea de visibilidad directa

haber obtenido apoyo de la Agencia Nacional de Investigación e Innovación (ANII) y por lo tanto demostrando ser dignas de tal mención.

Dronfies Labs

Dronfies Labs es una empresa dedicada 100 % al desarrollo de software para drones y operaciones aéreas. Se centra en el desarrollo de software altamente escalable aplicado a los drones del fabricante DJI [63]. Entre sus trabajos se encuentra una aplicación de entretenimiento con el nombre Dronfies, una experiencia de entrega con drones, y SENTINEL: Solución de Inspección Remota y Búsqueda y Rescate para bomberos y rescatadores. A su vez son los proveedores tecnológicos de la segunda empresa mencionada, Mi Halcón.

Dronfies es una aplicación para celulares para drones DJI que permite a sus pilotos enviar instantáneamente los contenidos que generan. Los contenidos son subidos a internet y luego una notificación por sms o email es enviada a su destinatario final con un enlace a los contenidos. A su vez la aplicación permite a sus usuarios realizar tomas aéreas fácilmente con vuelos automáticos. La empresa ofrece capacitación y una franquicia con materiales para aprender a utilizar los drones, junto con la aplicación de Dronfies, en acciones de marketing, turismo, y eventos con el fin de generar ingresos extra [64].

Mi Halcón

Mi Halcón es un sistema para supervisión aérea de propiedades. Mi Halcón instrumenta vuelos automáticos para supervisar diferentes puntos predefinidos de las propiedades, y realizan imágenes y vídeo que son transmitidos mediante una aplicación móvil. La solución incluye una capa de usabilidad que permite a los guardias de seguridad sin entrenamiento operar con seguridad los drones y una capa de información donde el propietario del activo y la agencia de seguridad pueden verificar las imágenes en tiempo real [65].

La empresa es la primera en obtener vuelos más allá de la línea de visibilidad directa (beyond VLOS). En febrero logró obtener el primer permiso otorgado por DINACIA en el país, para realizar pruebas en la zona de Ciudad de la Costa utilizando como base operativa una empresa de seguridad de la zona.

(a) Delfly

(b) Hubsan x4

(c) Bebop 2

(d) Phantom 4

(e) ebee

(f) Raven

(g) Scaneagle

Tabla 2.2: Imágenes de drones comúnmente usados

Capítulo 3

Algoritmos

Este capitulo expone diferentes tipos de algoritmos utilizados para resolver problemas de planificación. La primera sección explica por qué hay ciertos problemas que son difíciles de resolver, exponiendo sobre los principios de resolución mediante heurísticas. La segunda sección describe qué son los algoritmos de planificación y muestra algunos ejemplos de los problemas que resuelven. La tercer sección explica qué son los algoritmos evolutivos, cuales son los mecanismos por los cuales se llega a soluciones adecuadas y como son construidos. La cuarta sección es un resumen sobre la programación orientada a agentes.

3.1. Heurísticas

3.1.1. ¿Por qué algunos problemas son difíciles de resolver?

Los problemas en el mundo real son difíciles de resolver por varias razones:

- El número de posibles soluciones en el espacio de búsqueda es tan grande que prohíbe una búsqueda exhaustiva de la mejor respuesta.
- El problema es tan complicado que para llegar a una respuesta, hay que usar modelos tan simplificados del problema que cualquier resultado es esencialmente inútil.
- La función de evaluación propuesta que describe la calidad de una solución es ruidosa o varía con el tiempo, por lo que no solo requiere una solución sino toda una serie de soluciones.
- Las posibles soluciones están tan limitadas que la construcción de una respuesta factible es difícil, en especial buscar una solución óptima.

Naturalmente, esta lista podría ampliarse para incluir muchos otros posibles obstáculos. Por ejemplo, se podría incluir el ruido asociado a observaciones y mediciones, incertidumbre sobre la

información dada, y las dificultades planteadas por problemas que tienen objetivos múltiples y posiblemente en conflicto (que pueden requerir un conjunto de soluciones en lugar de una sola solución). Pero la lista anterior es suficiente para tener un entendimiento básico. Para resolver un problema, hay que entender el problema, por lo que en esta sección se analiza cada uno de estas razones.[66]

El tamaño del espacio de búsqueda

Uno de los problemas elementales de la lógica es el problema de satisfacibilidad booleana (también llamado SAT). El problema SAT es el problema de saber si, dada una expresión booleana con variables y sin cuantificadores, hay alguna asignación de valores para sus variables que hace que la expresión sea verdadera. Por ejemplo, considerando el siguiente problema de 100 variables expresado en su forma normal conjuntiva:

$$F(x) = (x_{17} \vee \neg x_{37} \vee x_{73}) \wedge (\neg x_{11} \vee \neg x_{56}) \wedge ... \wedge (x_2 \vee x_{43} \vee \neg x_{77} \vee \neg x_{89} \vee \neg x_{97})$$

El reto es encontrar la asignación de valor para cada una de las variables x_i, para $i = 1, ..., 100$ tal que $F(x) = V$. Se puede utilizar 1 como valor de verdad, y 0 como falso.

Independientemente del problema que se plantee, siempre es útil considerar el espacio de las posibles soluciones. Aquí, cualquier cadena binaria de longitud 100 constituye una solución potencial al problema. Hay dos opciones para cada variable, sobre 100 variables, esto genera 2^{100} posibilidades. Por lo que el tamaño del espacio de búsqueda E es $|E| = 2^{100} \approx 10^{30}$, lo cual es un número enorme. Intentar cada una de alternativas esta fuera de alcance. Si hubiera una computadora que puede comprobar 1000 opciones por segundo, y se hubiera comenzado a usar en el inicio de los tiempos, hace 15 billones de años, justo en el Big Bang, la computadora habría examinado menos de uno por ciento de las posibilidades hasta ahora.

Además, la elección de qué función de evaluación a utilizar no es muy clara. Lo ideal es que la función de evaluación brinde orientación sobre la calidad de la solución propuesta. Las soluciones que están más cerca de la respuesta correcta deberían dar mejores evaluaciones que las que están más lejos. Pero aquí, todo lo que se tiene es que $F(x)$ que puede evaluar a verdadero o falso. Si se evalúa una cadena x y $F(x)$ devuelve verdadero, entonces se encontró la respuesta. Pero ¿y si $F(x)$ devuelve falso?. Además, casi todas las cadenas posibles de 0 y 1 que se prueban son probablemente falsas, así que, ¿es posible distinguir entre soluciones potenciales "mejores" y "peores"? Si se utiliza una búsqueda enumerativa, no importaría porque simplemente habría que probar cada posibilidad. Pero si se quiere que la función de evaluación ayude a encontrar las mejores soluciones más rápido que la enumeración, es necesario algo más que "correcto" o "incorrecto". La forma en que se podría lograr eso para el problema del SAT no está clara de inmediato.

Otros ejemplos ilustrativos de problemas altamente estudiados con espacios de búsqueda enormes son el problema del vendedor ambulante (TSP) y algunos problemas de programación no lineales (NLP). Es evidente que algunos problemas que parecen simples al principio pueden ofrecer desafíos importantes simplemente debido a la cantidad de soluciones alternativas. Los medios para idear formas de evaluar estas soluciones no siempre son claros.

Modelado del problema

Cada vez se resuelve un problema hay que darse cuenta de que en realidad solo se está encontrando la solución a un modelo del problema. Todos los modelos son una simplificación del mundo real, de lo contrario serían tan complejos y difíciles de manejar como el entorno natural en sí. El proceso de resolución de problemas consta de dos pasos generales separados: (1) crear un modelo del problema y (2) usar ese modelo para generar una solución:

$$Problema \Longrightarrow Modelo \Longrightarrow Solución$$

La solución es solo una solución en términos del modelo. Si el modelo tiene un alto grado de fidelidad, se puede tener más confianza en que la solución será útil. En contraste, si el modelo tiene demasiados supuestos no cumplidos y aproximaciones, la solución puede carecer de sentido, o no servir para nada.

Hay dos enfoques posibles. El primero utiliza un modelo aproximado, $Modelo_a$, de un problema, y luego encuentra la solución precisa $Solución_p$ para este modelo aproximado:

$$Problema \Longrightarrow Modelo_a \Longrightarrow Solución_p(Modelo_a)$$

El segundo enfoque utiliza un modelo preciso, $Modelo_p$, del problema, y luego encuentra una solución aproximada, $Solución_a(Modelo_p)$, para este modelo preciso:

$$Problema \Longrightarrow Modelo_p \Longrightarrow Solución_a(Modelo_p)$$

Esta es la segunda fuente de dificultades en la resolución de problemas: es difícil obtener una solución precisa a un problema porque o bien hay que aproximar el modelo o aproximar la solución.

Cambios en el tiempo

Como si las dificultades anteriores no fueran suficientes, los problemas del mundo real a menudo presentan otro conjunto de dificultades: cambian. Cambian antes de ser modelados, cambian mientras se deriva una solución, y cambian después de ejecutar una solución.

Por ejemplo en el problema del vendedor ambulante el tiempo de viaje depende de muchos factores. Con suerte el vendedor perdía tener todos los semáforos verdes en el camino. O puede que tenga mala suerte y no solo llegue a las luces rojas sino que también quede atrapado detrás de un camión que se mueve lentamente. Peor aún, podría tener una llanta desinflada, lo que agregaría una cantidad significativa de tiempo a su viaje. Todas estas posibilidades, y un número inimaginable de otros resultados, como el clima, las condiciones de la carretera, los accidentes de tráfico, los vehículos de emergencia que requieren que se detenga, un tren en las vías que cruzan su camino, etc., pueden describirse bajo el encabezado de ruido o aleatoriedad. Una solución a este problema podría ser calcular un promedio de los valores variables. Pero cualquier decisión tomada en base a este tiempo promedio no toma en cuenta la variabilidad del tiempo, y esto puede ser mucho más importante para tomar buenas decisiones.

La posibilidad aleatoria no es la única fuente de cambio en los problemas del mundo real. A veces también hay problemas puramente deterministas. Es sabido, por ejemplo, que viajar en horas pico llevará más tiempo en cada ciudad que viajar a medianoche. Es posible que no se sepa exactamente cuánto tiempo tomará, es un evento aleatorio, pero se sabe que existe una tendencia a que en la hora pico se paralice el progreso en el tráfico. Ese sesgo es un patrón regular y predecible, y debe considerarse o, de lo contrario, el modelo podría no corresponder lo suficiente a la realidad y las soluciones con ese modelo no serán útiles. En el peor de los casos, una ruta particular entre dos ciudades podría estar disponible solo durante ciertas horas del día, y no estará disponible de otra manera. Actuar como si la ruta no disponible aún fuera una opción llevaría a una solución no viable.

La situación, sin embargo, es aún más compleja que esto. Las variaciones anteriores pueden ocurrir en función de cambios ambientales o eventos incontrolables, pero ninguno de ellos conspiró en contra de la solución. Desafortunadamente, en el mundo real, a menudo es el caso de que otras entidades intentan anular una solución de la competencia, y esto requiere que actualizar continuamente el modelo y anticipar las acciones de las otras entidades.

Restricciones

Casi todos los problemas prácticos presentan restricciones, y si se violan las restricciones, la solución no se puede implementar. En principio, uno podría pensar que los problemas restrictivos serian más fáciles. Después de todo, el espacio de búsqueda es más pequeño y, por lo tanto, hay menos posibilidades que considerar. Eso es cierto, pero para buscar soluciones mejores debe haber una manera de pasar de una solución a la siguiente. Son necesarios operadores que actúen sobre soluciones factibles para generar nuevas soluciones factibles mejores que las ya encontradas. Es aquí donde la geometría del espacio de búsqueda se vuelve difícil.

Por ejemplo, supongamos hay que resolver el problema de hacer un calendario para todas las clases de un semestre de una universidad. Primero, hay que hacer una lista de todos los cursos que se ofrecen. Se necesita una lista de todos los estudiantes asignados a cada clase, y el profesor asignado. En tercer lugar, es necesaria una lista de aulas disponibles, teniendo en cuenta el tamaño y las instalaciones que cada uno ofrece (por ejemplo, una pizarra, un vídeo proyector, equipo de laboratorio, etc.). Hay tres restricciones difíciles:

- Cada clase debe asignarse a una sala disponible que tenga suficientes asientos para cada estudiante asignado y tenga las instalaciones necesarias para el tipo de asignatura (por ejemplo, un laboratorio de química debe tener vasos, quemadores Bunsen, productos químicos apropiados, etc.).
- Los estudiantes que están matriculados en más de una clase no pueden tener sus clases a la misma hora el mismo día.
- No se puede asignar a los profesores para que enseñen cursos que se superponen en el tiempo.

Esas son las restricciones difíciles. Estas son las cosas que deben satisfacerse absolutamente para tener una solución viable. Con lo presentado hasta ahora, cualquier asignación que cumpla con las

restricciones resolvería el problema. Entonces, esto significa que la tarea es bastante similar al problema de SAT: hay que encontrar una asignación de clases (en comparación con las variables booleanas) de manera que una función de evaluación general devuelva un valor de verdadero. Cualquier cosa que viole las restricciones significa que la función de evaluación devuelve un valor de falso. Pero esto solo no proporciona información suficiente para guiar la búsqueda de una solución viable.

Se podría emplear alguna estrategia que pueda proporcionar esta información adicional. Por ejemplo, se podría juzgar la calidad de la solución no solo por si satisface o no las restricciones, sino por aquellas asignaciones que no cumplen con las restricciones, se podría calcular el número de veces que se violan las restricciones (por ejemplo, cada vez que a un estudiante se le asignan dos clases que se son al mismo tiempo aumentamos un contador). Esto daría una medida cuantitativa de cuán pobres son nuestras soluciones no viables, y podría ser útil para guiar la búsqueda hacia soluciones cada vez mejores, minimizando el número de violaciones de restricciones. Se podrían aplicar diferentes operadores para reasignar cursos a aulas, profesores a cursos, etc., y con el tiempo generar una solución que cumpla con las restricciones disponibles.

Pero luego están las restricciones blandas, las cosas que se esperan lograr pero que no son obligatorias. Éstas incluyen:

- Los cursos que se reúnen dos veces por semana deben asignarse preferentemente a los lunes y los miércoles o los martes y los jueves. No es deseable tener estos cursos en días consecutivos o con dos o más días intermedios.
- Los cursos que se reúnen tres veces por semana deben asignarse preferentemente a los lunes, miércoles y viernes. No se desean otras asignaciones.
- Si más de una sala satisface los requisitos para un curso y está disponible a la hora designada, el curso debe asignarse a la sala con la capacidad más cercana al tamaño de la clase (esto significa que no se utilizan salas grandes para clases pequeñas) , mejorando así la participación de los estudiantes).

Ciertamente se podrían imaginar muchas otras restricciones blandas. Cualquier asignación que cumpla con las restricciones difíciles es factible, pero no necesariamente óptima a la luz de las restricciones blandas.

Primero, hay que cuantificar cada una de las restricciones blandas en términos matemáticos para poder evaluar dos asignaciones y decidir si una es mejor que la otra. A continuación, se debe poder modificar una solución factible y generar otra solución factible que satisfaga mejor las restricciones blandas. El primer problema se puede resolver penalizando las soluciones si estas no cumplen con las restricciones. Pero existe el problema de determinar cual de las restricciones es más importante. Este tipo de decisiones deben ser cuantificadas en la función de evaluación, lo cual no es fácil.

Incluso después de que todas estas restricciones blandas se hayan cuantificado, aún queda el problema de buscar la mejor asignación: la solución que es factible y minimiza la función de evaluación para las restricciones blandas. Es posible que si se encuentra una solución factible que no funciona muy bien con respecto a las restricciones blandas y se aplican algunos operadores para mejorar la

situación con respecto a las restricciones blandas, al hacerlo, la solución deje de ser factible. Se podría elegir descartar la solución ya que no es factible, o ver si se puede reparar para generar una solución factible que aún maneje bien las restricciones blandas. De cualquier manera, esta es típicamente una tarea difícil. Sería incluso mejor idear operadores de variación que nunca corrompan una solución factible mientras siguen buscando vigorosamente el espacio de soluciones factibles para encontrar aquellos que manejen mejor las restricciones blandas. Esa es una buena aspiración, pero a menudo no es mucho más que una ilusión.

3.1.2. Conceptos básicos de modelado

Tres conceptos básicos son comunes en los algoritmos para la resolución de problemas. Independientemente de la técnica que se emplee, se deberá especificar: (1) la representación, (2) el objetivo y (3) la función de evaluación. La representación codifica soluciones candidatas alternativas para la manipulación, el objetivo describe el propósito que debe cumplirse y la función de evaluación devuelve un valor específico que indica la calidad de una solución (o mínimamente, permite una comparación de la calidad de dos soluciones alternativas). En esta sección se describen cada uno de estos aspectos de la resolución de problemas.

Representación

En el problema del SAT hay n variables que son bits lógicos, una forma obvia de representar una solución candidata es una cadena binaria de longitud n. Cada elemento en la cadena corresponde a una variable en el problema. El tamaño del espacio de búsqueda resultante bajo esta representación es 2^n, ya que hay 2^n cadenas binarias diferentes de longitud n. Cada punto en ese espacio de búsqueda es una solución factible.

Para cada problema, la representación de una solución potencial y su correspondiente interpretación determina el espacio de búsqueda y su tamaño. Este es un punto importante para reconocer: el tamaño del espacio de búsqueda no está determinado por el problema, está determinado por su representación y la manera en que se maneja la codificación.

Elegir el espacio de búsqueda correcto es de suma importancia. Si no se selecciona el dominio correcto para comenzar la búsqueda, se pueden agregar numerosas soluciones inviables o duplicadas a la lista de posibilidades, o puede que en realidad no se pueda encontrar la respuesta correcta.

Objetivo

Una vez que se define el espacio de búsqueda, uno tiene que decidir lo que está buscando. ¿Cuál es el objetivo del problema? Esta es una declaración matemática de la tarea a realizar. No es una función, sino una expresión. Para el SAT, el objetivo es encontrar el vector de bits de modo que se satisfaga la declaración booleana compuesta (es decir, que de verdadera).

Función de evaluación

El objetivo, sin embargo, no es lo mismo que la función de evaluación. Esto último es más típicamente un mapeo desde el espacio de posibles soluciones candidatas bajo la representación elegida hasta un conjunto de números (por ejemplo, los reales), donde a cada elemento del espacio de posibles soluciones se le asigna un valor numérico que indica su calidad. La función de evaluación permite comparar el valor de las soluciones alternativas. Algunas funciones de evaluación permiten hacer una clasificación de todas las soluciones posibles. Esto se llama una función de evaluación ordinal. Alternativamente, otras funciones de evaluación son numéricas y determinan no solo el orden de las soluciones en términos de la calidad percibida, sino también el grado de esa calidad.

En los problemas del mundo real, hay que elegir la función de evaluación, no viene gratis con el problema. Un criterio obvio para satisfacer es que cuando una solución cumple con el objetivo por completo, también debe tener la mejor evaluación. La función de evaluación no debe indicar que una solución que no cumple con el objetivo es mejor que una que cumple con el objetivo. Pero esto es solo un comienzo rudimentario para diseñar la función de evaluación.

Cuando se diseña la función de evaluación, se debe considerar que, para muchos problemas, las únicas soluciones de interés estarán en un subconjunto del espacio de búsqueda S. Solo interesan las soluciones factibles, es decir, soluciones que satisfagan las limitaciones específicas del problema. Este subconjunto constituye la parte factible del espacio de búsqueda F.

Definición del problema de búsqueda

Ahora podemos definir un problema de búsqueda. Dado un espacio de búsqueda S junto con su parte factible $F \subseteq S$, encuentre $x \in F$ tal que

$$eval(x) \leq eval(y),$$

para todo $y \in F$. Se debe tener en cuenta que aquí se está usando una función de evaluación para la cual las soluciones que devuelven valores más pequeños se consideran mejores (es decir, el problema es de minimización), pero para cualquier problema, se puede fácilmente utilizar una función de evaluación para la cual se favorecen valores más grandes, convirtiendo el problema de búsqueda en uno de maximización. No hay nada inherente en el problema original o en un modelo aproximado que exija que la función de evaluación se configure para la minimización o la maximización. De hecho, el objetivo en sí no aparece en ninguna parte en el problema de búsqueda definido anteriormente. Esta declaración de problema se podría usar con la misma facilidad para describir un TSP, un SAT o un NLP. La búsqueda en sí misma no sabe qué problema estás resolviendo. Todo lo que sabe es la información que proporciona en la función de evaluación, la representación que utiliza y la manera en que se hace el muestreo de las posibles soluciones. Si la función de evaluación no se corresponde con el objetivo, se estará buscando la respuesta correcta al problema incorrecto.

El punto x que satisface la condición anterior se denomina solución global. Encontrar una solución global de este tipo para un problema puede ser muy difícil. Pero a veces encontrar la mejor solución es más fácil cuando es posible concentrarse en un subconjunto relativamente pequeño del

espacio de búsqueda total (factible o tal vez también no factible). Menos alternativas hacen que los problemas sean mucho más fáciles. Esta es una observación fundamental que subyace a muchas técnicas de búsqueda.

3.2. Algoritmos de planificación

Planificación es un término que significa diferentes cosas para diferentes grupos de personas. La robótica aborda la automatización de sistemas mecánicos que tienen capacidades de detección, actuación y computación. Una necesidad fundamental en robótica es tener algoritmos que conviertan las especificaciones de tareas de alto nivel de los humanos en descripciones de bajo nivel de cómo moverse. Los términos planificación de movimiento y planificación de trayectoria se usan a menudo para este tipo de problemas. Una versión clásica de la planificación de movimiento a veces se conoce como el problema del movimiento del piano. Con un modelo preciso de diseño asistido por computadora (CAD) de una casa y un piano como entrada a un algoritmo, el algoritmo debe determinar cómo mover el piano de una habitación a otra en la casa sin golpear nada. La planificación del movimiento del robot generalmente ignora la dinámica y otras restricciones y se enfoca principalmente en las traslaciones y rotaciones necesarias para mover el piano. La planificación de la trayectoria generalmente se refiere al problema de tomar la solución de un algoritmo de planificación de movimiento del robot y determinar cómo moverse a lo largo de la solución de una manera que respete las limitaciones mecánicas del robot.

La teoría del control ha estado históricamente relacionada con el diseño de entradas para sistemas físicos descritos por ecuaciones diferenciales. Estos podrían incluir sistemas mecánicos como automóviles o aviones, sistemas eléctricos como filtros de ruido o incluso sistemas que surgen en áreas tan diversas como la química, la economía y la sociología. Clásicamente, la teoría de control ha desarrollado políticas de retroalimentación, que permiten una respuesta adaptativa durante la ejecución, y se ha centrado en la estabilidad, lo que garantiza que la dinámica no haga que el sistema se salga de control. También se pone un gran énfasis en los criterios de optimización para minimizar el consumo de recursos, como la energía o el tiempo. En la literatura reciente sobre teoría de control, la planificación del movimiento a veces se refiere a la construcción de entradas para un sistema dinámico no lineal que lo conduce desde un estado inicial a un estado objetivo específico. Por ejemplo, en el contexto de este trabajo, se intenta operar un dron a control remoto o autocontrolado sobre un predio. Supongamos que nos gustaría que el dron dejara su ubicación de descanso actual y se detuviera en otra ubicación específica. ¿Se puede diseñar un algoritmo que calcule las entradas deseadas, incluso en un simulador ideal que ignora las incertidumbres que surgen de las imprecisiones del modelo? Es posible agregar otras consideraciones, como incertidumbres, comentarios y optimalidad, sin embargo, el problema ya es suficientemente desafiante sin estos.

En inteligencia artificial, los términos planificación y planificación de inteligencia artificial suelen referirse a problemas más discretos. En lugar de mover un piano a través de un espacio continuo, como en el problema de la planificación del movimiento del robot, la tarea podría ser resolver un rompecabezas, como un cubo de Rubik o un rompecabezas de azulejos deslizantes, o lograr una

tarea que se modele de forma discreta, por ejemplo: como construir una pila de bloques. Aunque tales problemas podrían modelarse con espacios continuos, parece natural definir un conjunto finito de acciones que se pueden aplicar a un conjunto discreto de estados y construir una solución al dar la secuencia apropiada de acciones. Históricamente, la planificación se ha considerado diferente de la resolución de problemas, sin embargo, la distinción parece haberse desvanecido en los últimos años.

Dada la amplia gama de problemas a los que se aplicó el término planificación en las comunidades de inteligencia artificial, teoría de control y robótica, podría preguntarse si tiene un significado específico. De lo contrario, casi cualquier cosa podría considerarse como un ejemplo de planificación.

Las preguntas naturales a esta altura son: ¿Qué es un plan? ¿Cómo se representa un plan? ¿Cómo se calcula? ¿Qué se supone que debe lograr? ¿Cómo se evalúa su calidad? ¿Quién o qué lo va a utilizar? Esta sección proporciona respuestas generales a estas preguntas. Respecto al usuario del plan, depende claramente de la aplicación. En la mayoría de las aplicaciones, un algoritmo ejecuta el plan, sin embargo, el usuario podría incluso ser un humano. El usuario del plan será referido frecuentemente como un robot o un tomador de decisiones. En inteligencia artificial y áreas relacionadas, se ha hecho popular en los últimos años el uso del término agente. La teoría del control usualmente se refiere al tomador de decisiones como un controlador. El plan en este contexto a veces se conoce como una política o ley de control. En un contexto de teoría de juegos, podría tener sentido referirse a los tomadores de decisiones como jugadores.[67]

3.2.1. Motivación y aplicaciones

¿Por qué estudiar algoritmos de planificación? Hay al menos dos buenas razones. Primero, es divertido intentar que las máquinas resuelvan problemas para los que incluso los humanos tienen grandes dificultades. Esto implica desafíos interesantes en el modelado de problemas de planificación, diseño de algoritmos eficientes y desarrollo de implementaciones sólidas. En segundo lugar, los algoritmos de planificación han logrado éxitos generalizados en varias industrias y disciplinas académicas, incluidas la robótica, la fabricación, el diseño de medicamentos y las aplicaciones aeroespaciales. El rápido crecimiento en los últimos años indica que muchas otras aplicaciones fascinantes pueden estar en el horizonte. Estos son tiempos emocionantes para estudiar algoritmos de planificación y contribuir a su desarrollo y uso.

Los algoritmos de **planificación discreta**, se pueden aplicar para resolver rompecabezas, como los que se muestran en la Figura 3.1. También son buenos en juegos como el ajedrez o el bridge [68].

Los rompecabezas en la Figura 3.1 pueden ser fácilmente discretizados debido a la regularidad y simetrías involucradas en el movimiento de las partes. La Figura 3.2 muestra un problema que carece de estas propiedades y requiere planificación en un espacio continuo. Tales problemas se resuelven utilizando las técnicas de **planificación de movimiento**. Este rompecabezas fue diseñado para frustrar a los humanos y los algoritmos de planificación de movimiento. Pero se puede resolver en unos minutos en una computadora personal estándar usando técnicas conocidas. Se han desarrollado muchos otros rompecabezas como puntos de referencia para evaluar algoritmos de planificación.

Si bien el problema en la Figura 3.2 puede parecer pura diversión y juegos, surgen problemas

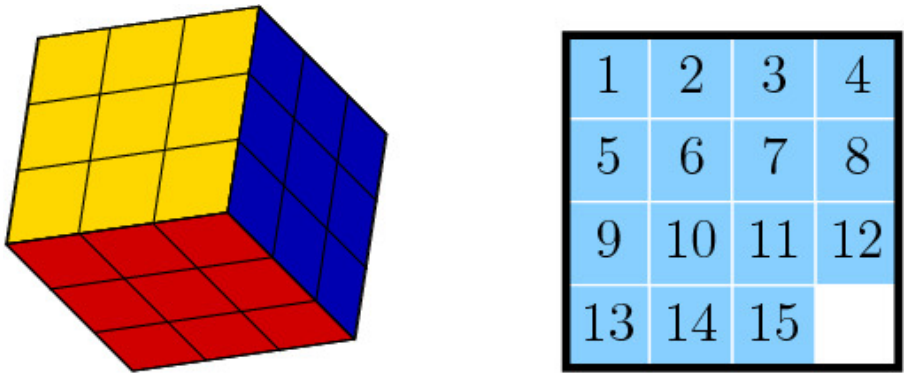


Figura 3.1: El cubo de Rubik (a), el rompecabezas de baldosas deslizantes (b) y otros rompecabezas relacionados son ejemplos de problemas de planificación discretos.

similares en aplicaciones importantes. Por ejemplo, problemas de ensamblaje automotriz para el cual se necesita software para determinar si una pieza se puede insertar (y quitar) a través de una cavidad. Tradicionalmente, este problema se resuelve construyendo modelos físicos. Esta parte costosa y lenta del proceso de diseño puede eliminarse virtualmente en el software al manipular directamente los modelos virtuales. Otras aplicaciones como el calculo automático de movimientos necesarios para que robots complejos realicen tareas ahorra tiempo y gastos enormes en el proceso de fabricación, permitiendo a los ingenieros especificar las tareas a alto nivel.

Una tarea más común para los robots móviles es solicitarles que naveguen en un ambiente interior, como se muestra en la Figura 3.3a. Se le puede pedir a un robot que realice tareas como construir un mapa del entorno, determinar su ubicación precisa dentro de un mapa o llegar a un lugar en particular. **Adquirir y manipular información de los sensores** es bastante difícil. La mayoría de los robots operan a pesar de grandes **incertidumbres**. En un extremo, puede parecer que tener muchos sensores es beneficioso porque podría permitir una estimación precisa del entorno y la posición y orientación del robot. Esta es la premisa de muchos sistemas existentes, como se muestra para el sistema de robot en la Figura 3.4, que construye un mapa de su entorno. Alternativamente, puede ser preferible desarrollar robots confiables y de bajo costo que logren tareas específicas con poca o ninguna detección.

Si hay varios robots, entonces surgen muchos problemas adicionales. ¿Cómo pueden comunicarse los robots? ¿Cómo se puede integrar su información? ¿Debería centralizarse o distribuirse su coordinación? ¿Cómo pueden evitarse las colisiones entre ellos? ¿Logran cada uno tareas independientes o están obligados a colaborar de alguna manera? Si compiten de alguna manera, entonces pueden aplicarse los conceptos de la **teoría de juegos**.

Una tarea importante para un robot móvil es jugar al **juego del escondite**. Imagina entrar en una cueva en completa oscuridad. Se le da una linterna y se le pide que busque a cualquier persona que pueda estar moviéndose, como se muestra en la Figura 3.3b. Surgen varias preguntas. ¿Existe alguna estrategia que garantice que encontraré a todos? Si no es así, ¿cuántos otros buscadores son necesa-

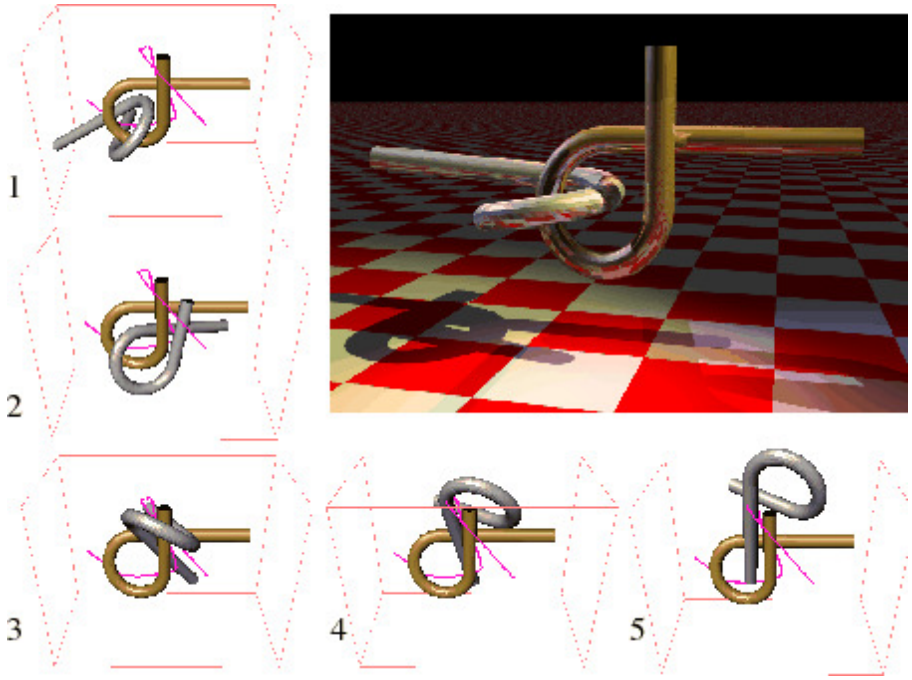


Figura 3.2: El objetivo de este rompecabezas es separar las dos barras. Este ejemplo se llama Alpha 1.0 Puzzle. Fue creado por Boris Yamrom y publicado como un punto de referencia de investigación por Nancy Amato en la Universidad de Texas A&M.

rios antes de poder completar esta tarea? ¿Dónde debo moverme después? ¿Puedo evitar explorar los mismos lugares varias veces? Este escenario surge en muchas aplicaciones de robótica. Los robots pueden integrarse en sistemas de vigilancia que utilizan robots móviles con varios tipos de sensores (movimiento, térmicos, cámaras, etc.). En escenarios que involucran múltiples robots con poca o ninguna comunicación, la estrategia podría ayudar a un robot a ubicar a otros. Un robot podría incluso intentar localizar otro que esté funcionando mal. Fuera de la robótica, se pueden desarrollar herramientas de software que ayuden a las personas a buscar o cubrir entornos complicados, para usos como la aplicación de la ley, búsqueda y rescate, limpieza de tóxicos y en el diseño arquitectónico de edificios seguros. El problema es extremadamente difícil porque el estado de la búsqueda debe calcularse cuidadosamente para evitar la búsqueda innecesaria en lugares ya visitados. Los conceptos de espacio de información se vuelven críticos para resolver el problema. La formulación y solución exitosas de tales problemas depende de la capacidad para manipular, simplificar y controlar el espacio de información. En algunos casos, existen soluciones elegantes, y en otros no parece haber ninguna esperanza de hacerlo de manera eficiente. Hay muchos problemas de investigación abiertos relacionados con los espacios de información y la detección bajo incertidumbre en general. Una solución algorítmica para el juego de escondite, es conocida como persecución-evasión basada en visibilidad [70, 71].

El problema en la Figura 3.3b puede parecer un videojuego. En el clásico de arcade Pacman, los

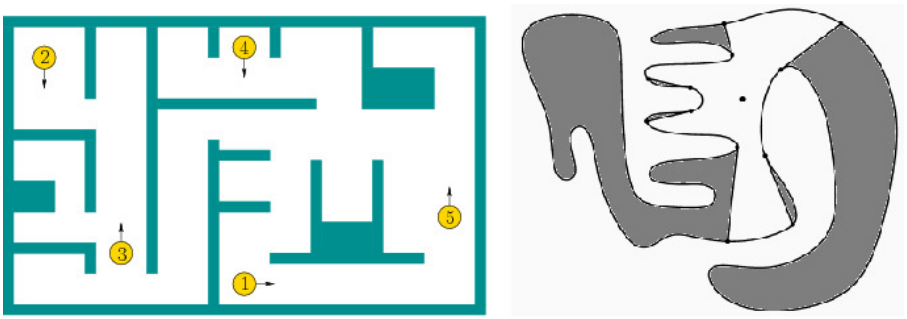


Figura 3.3: (a) Varios robots móviles intentan navegar con éxito en un ambiente interior evitando colisiones con las paredes y entre sí. (b) busqueda con una linterna en una cueva.

fantasmas están programados para buscar al jugador. Los videojuegos modernos involucran personajes parecidos a los humanos que exhiben un comportamiento mucho más sofisticado. Los algoritmos de planificación pueden permitir a los desarrolladores de juegos **programar comportamientos de personajes** a un nivel superior, con la expectativa de que el personaje puede determinar por sí mismo de una manera inteligente.

Más allá de los videojuegos, existe un interés más amplio en el desarrollo de seres humanos virtuales. En el campo de los gráficos de computadora, las **animaciones generadas por computadora** son de gran interés. A los animadores les gustaría desarrollar actores digitales que mantengan muchas características que distinguen a los actores humanos y, al mismo tiempo, poder diseñar movimientos para ellos a partir de descripciones de alto nivel. Es extremadamente tedioso y lleva mucho tiempo especificar todos los movimientos cuadro por cuadro. El desarrollo de algoritmos de planificación en este contexto se está expandiendo rápidamente.

Los problemas de planificación discutidos hasta ahora no han involucrado restricciones diferenciales. Por ejemplo, el problema de estacionar vehículos de movimiento lento. La mayoría de las personas tienen un poco de dificultad con el estacionamiento paralelo de un automóvil y mucha más dificultad para estacionar un camión con un remolque. ¿Qué hace que estos problemas sean tan desafiantes? Un automóvil está obligado a moverse en la dirección que apuntan las ruedas traseras. Manejar el coche alrededor de los obstáculos, por lo tanto, se vuelve un reto. Si las cuatro ruedas pudieran girar a cualquier orientación, este problema desaparecería. El término **planificación no holonómica** abarca problemas de estacionamiento y muchos otros.

Los algoritmos de planificación pueden ayudar a navegar helicópteros autónomos a través de obstáculos. También pueden calcular los empujes de una nave espacial para evitar colisiones alrededor de una estructura complicada, como una estación espacial. La planificación de la misión para naves espaciales interplanetarias, incluidas las velas solares, puede realizarse utilizando algoritmos de planificación [72].

Los algoritmos de planificación incluso afectan campos tan lejos de la robótica como la **biología computacional**. Dos problemas principales son el plegamiento de proteínas y el diseño de fármacos.

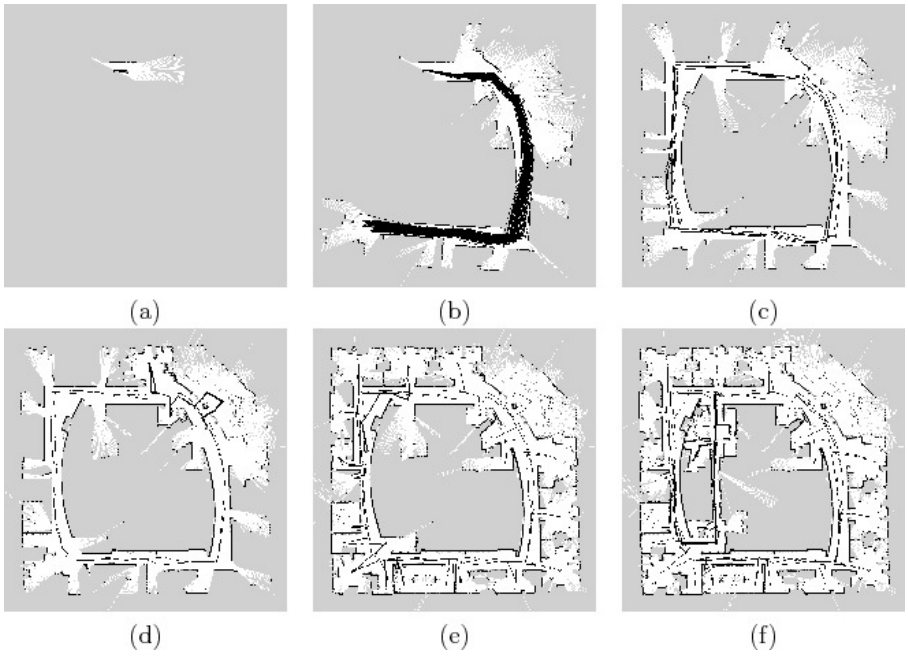


Figura 3.4: Un robot móvil puede construir de manera confiable un buen mapa de su entorno (aquí, el Laboratorio de Investigación Intel) mientras se localiza simultáneamente. Esto se logra utilizando sensores de escaneo láser y realizando cálculos bayesianos eficientes en el espacio de información [69].

En ambos casos, los científicos intentan explicar los comportamientos en los organismos por la forma en que interactúan las moléculas orgánicas grandes. Tales moléculas son generalmente flexibles. Las moléculas del fármaco son pequeñas, y las proteínas generalmente tienen miles de átomos. El problema de acoplamiento implica determinar si una molécula flexible puede insertarse en la cavidad de una proteína, al mismo tiempo que satisface otras restricciones, como mantener la energía baja. Una vez que los modelos geométricos se aplican a las moléculas, el problema se ve muy similar los problemas de ensamblaje mencionados anteriormente y se puede resolver mediante algoritmos de planificación de movimiento.

Los algoritmos de planificación se han aplicado a muchos más problemas que los que se mencionan aquí. En algunos casos, el trabajo ha pasado de modelado, a algoritmos teóricos, a software práctico que se utiliza en la industria. En otros casos, queda investigación sustancial para llevar los métodos de planificación a su máximo potencial.

3.2.2. Ingredientes básicos de la planificación

Aunque existen muchos algoritmos de planificación, hay varios ingredientes básicos que surgen en prácticamente todos. En esta sección se describen estos ingredientes.

Estado

Los problemas de planificación implican un espacio de estado que captura todas las situaciones posibles que podrían surgir. El estado podría, por ejemplo, representar la posición y la orientación de un robot, las ubicaciones de los azulejos en un rompecabezas o la posición y la velocidad de un helicóptero. Hay espacios de estado discretos (finitos, o infinitamente contables) y continuos (infinitamente infinitos). Un tema recurrente es que el espacio de estado suele estar representado implícitamente por un algoritmo de planificación. En la mayoría de las aplicaciones, el tamaño del espacio de estado (en términos de número de estados o complejidad combinatoria) es demasiado grande para ser representado explícitamente. Sin embargo, la definición del espacio estatal es un componente importante en la formulación de un problema de planificación y en el diseño y análisis de algoritmos que lo resuelven.

Tiempo

Todos los problemas de planificación implican una secuencia de decisiones que deben aplicarse en el tiempo. El tiempo se puede modelar explícitamente, como en un problema de conducir un auto lo más rápido posible a través de una carrera de obstáculos. Alternativamente, el tiempo puede ser implícito, simplemente reflejando el hecho de que las acciones deben seguir en sucesión, como en el caso de resolver un cubo de Rubik. El momento en particular no es importante, pero se debe mantener la secuencia correcta. Otro ejemplo de tiempo implícito es una solución al problema del movimiento del piano. La solución para mover el piano puede convertirse en una animación a lo largo del tiempo, pero la velocidad particular no se especifica en el plan. Como en el caso de los espacios de estado, el tiempo puede ser discreto o continuo.

Comportamiento

Un plan genera acciones que manipulan el estado. Los términos acciones y operadores son comunes en inteligencia artificial. En teoría del control y robótica, los términos relacionados son entradas y controles. En algún lugar de la formulación de una planificación, debe especificarse cómo cambia el estado cuando se aplican las acciones. Esto puede expresarse como una función de valores de estado para el caso de tiempo discreto o como una ecuación diferencial ordinaria para tiempo continuo. Para la mayoría de los problemas de planificación de movimiento, se evita la referencia explícita al tiempo especificando directamente una ruta a través de un espacio de estado continuo. Para algunos problemas, las acciones pueden ser elegidas por la naturaleza sin el control del tomador de decisiones, y podrían interferir con el resultado. Esto puede introducir incertidumbre en la previsibilidad del problema de planificación.

Estado inicial y objetivo

Un problema de planificación generalmente implica comenzar en algún estado inicial y tratar de llegar a un estado objetivo específico o cualquier estado en un conjunto de estados objetivo. Las

acciones se seleccionan de una manera que intenta hacer que esto suceda.

Un criterio

Esto codifica el resultado deseado de un plan en términos del estado y las acciones que se ejecutan. En general, existen dos tipos de criterios:

- Viabilidad: encontrar un plan que cause la llegada a un estado objetivo, independientemente de su eficiencia.
- Optimalidad: encontrar un plan factible que optimice el rendimiento de alguna manera cuidadosamente especificada, además de llegar a un estado objetivo.

Para la mayoría de los problemas, la viabilidad ya es lo suficientemente desafiante; lograr la optimalidad es considerablemente más difícil para la mayoría de los problemas. Por lo tanto, gran parte de la atención se centra en encontrar soluciones viables a los problemas, en lugar de soluciones óptimas. La mayoría de la literatura en robótica, teoría de control y campos relacionados se centra en la optimalidad, pero esto no es necesariamente importante para muchos problemas de interés. En muchas aplicaciones, es difícil incluso formular el criterio correcto para optimizar. Incluso si se puede formular un criterio deseable, puede ser imposible obtener un algoritmo práctico que calcule los planes óptimos. En tales casos, las soluciones factibles son ciertamente preferibles a no tener ninguna solución. Afortunadamente, para muchos algoritmos, las soluciones producidas no están muy lejos de ser óptimas en la práctica. Esto reduce parte de la motivación para encontrar soluciones óptimas. Sin embargo, para problemas que involucran incertidumbre probabilística, la optimización surge con más frecuencia. Las probabilidades se utilizan a menudo para obtener el mejor rendimiento en términos de costos esperados. La viabilidad a menudo se asocia con la realización de un análisis de incertidumbres en el peor de los casos.

Un plan

En general, un plan impone una estrategia o comportamiento específico a quien toma las decisiones. Un plan puede simplemente especificar una secuencia de acciones a tomar. Sin embargo, podría ser más complicado. Si es imposible predecir estados futuros, entonces el plan puede especificar acciones en función del estado. En este caso, independientemente de los estados futuros, se determina la acción apropiada. Incluso podría darse el caso de que el estado no se pueda medir. En este caso, la acción apropiada debe determinarse a partir de la información disponible hasta el momento actual. Esto generalmente se denomina como un estado de información en el que están condicionadas las acciones de un plan.

3.3. Planificación de trayectoria y movimiento

Existen distintas formas de abordar el problema de transformar la especificación de una tarea (proporcionada por humanos) en una descripción de bajo nivel adecuada para controlar los drones.

Un algoritmo de planificación de movimiento provee las transformaciones necesarias para mover un robot considerando todas las restricciones operativas. En la figura 3.5 se puede ver un esquema de las partes involucradas.

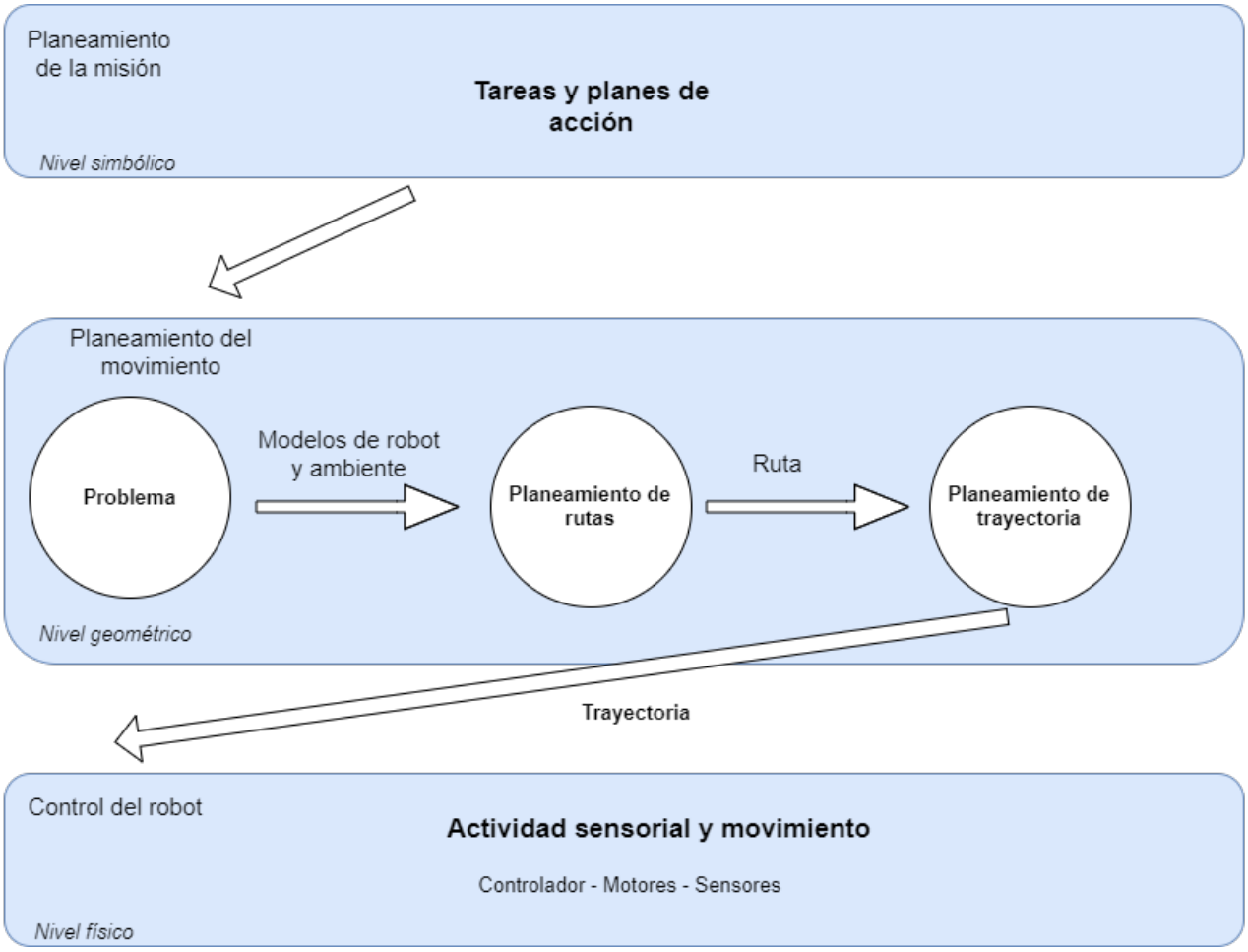


Figura 3.5: Planificación de movimiento en robots

En la realidad el problema es complejo. El mundo cambia constantemente, los robots actualizan su conocimiento sobre el medio ambiente y deben tomar nuevas decisiones. Por esto la planificación del movimiento es parte de un ciclo de replanificación. Los algoritmos fuera de línea se transforman en algoritmos de planificación en línea.

Este tipo de problema en 3 dimensiones es PSPACE-hard. Un problema clásico de la literatura es el problema de como trasladar un piano. [73] Existen algoritmos completos que abordan este problema, pero son demasiado complejos para ser prácticos. La investigación se ha centrado en algoritmos de aproximación que hacen un canje entre la completitud e integridad de la solución por mejor eficiencia del planificador. Los enfoques más exitosos son métodos de descomposición celular,

planificadores basados en el muestreo aleatorio (PRM, RRT por sus siglas en ingles) y planificadores probabilísticos basados en el muestreo óptimo (RRG, por sus siglas en ingles). [74]

Rapidly exploring random tree

Rapidly exploring random tree (RRT) es un algoritmo eficiente para buscar en un espacio no convexo de múltiples dimensiones mediante la construcción de un árbol. El árbol llena el espacio incrementalmente, eligiendo puntos del espacio de búsqueda de forma aleatoria y sesgada hacia lugares no explorados del problema. En el Algoritmo 1 se puede ver un pseudocódigo del algoritmo y en la figura 3.6 un ejemplo de aplicación de los primeros pasos en un problema genérico. [75]

Algoritmo 1 Esquema Algoritmo RRT

1: Entrada: q_{init}, numero de muestras n
2: Salida: $G = (V, E)$
3: $V \leftarrow \{q_{init}\}$
4: $E \leftarrow \emptyset$
5: **for** $i \leftarrow 1, n$ **do**
6: $\quad q_{rand} \leftarrow MuestraRandom$
7: $\quad q_{cercano} \leftarrow \text{CERCANO}(G = (V, E), q_{rand})$
8: $\quad q_{nuevo} \leftarrow \text{DIRIGIR}(q_{cercano}, q_{rand})$
9: $\quad$ **if** $\text{NOCOLISIONA}(q_{cercano}, q_{nuevo})$ **then**
10: $\quad\quad V \leftarrow V \cup \{q_{nuevo}\}$
11: $\quad\quad E \leftarrow E \cup \{(q_{cercano}, q_{nuevo})\}$
12: **return** $G = (V, E)$

Entre las principales características se encuentran:

- Explora el espacio rápidamente.
- Permite considerar restricciones dinámicas y kinodinámicas durante la expansión.
- Pobre rendimiento en problemas con espacios angostos.
- Proporciona rutas viables
- Existen variantes dependiendo de la técnica de muestreo utilizada

3.4. Algoritmos evolutivos

Muchas técnicas para resolver problemas se basan en una única solución como base para futuras exploraciones con cada iteración. O bien procesan soluciones completas en su totalidad, o construyen la solución final a partir de bloques de construcción más pequeños. Los algoritmos greedy, por

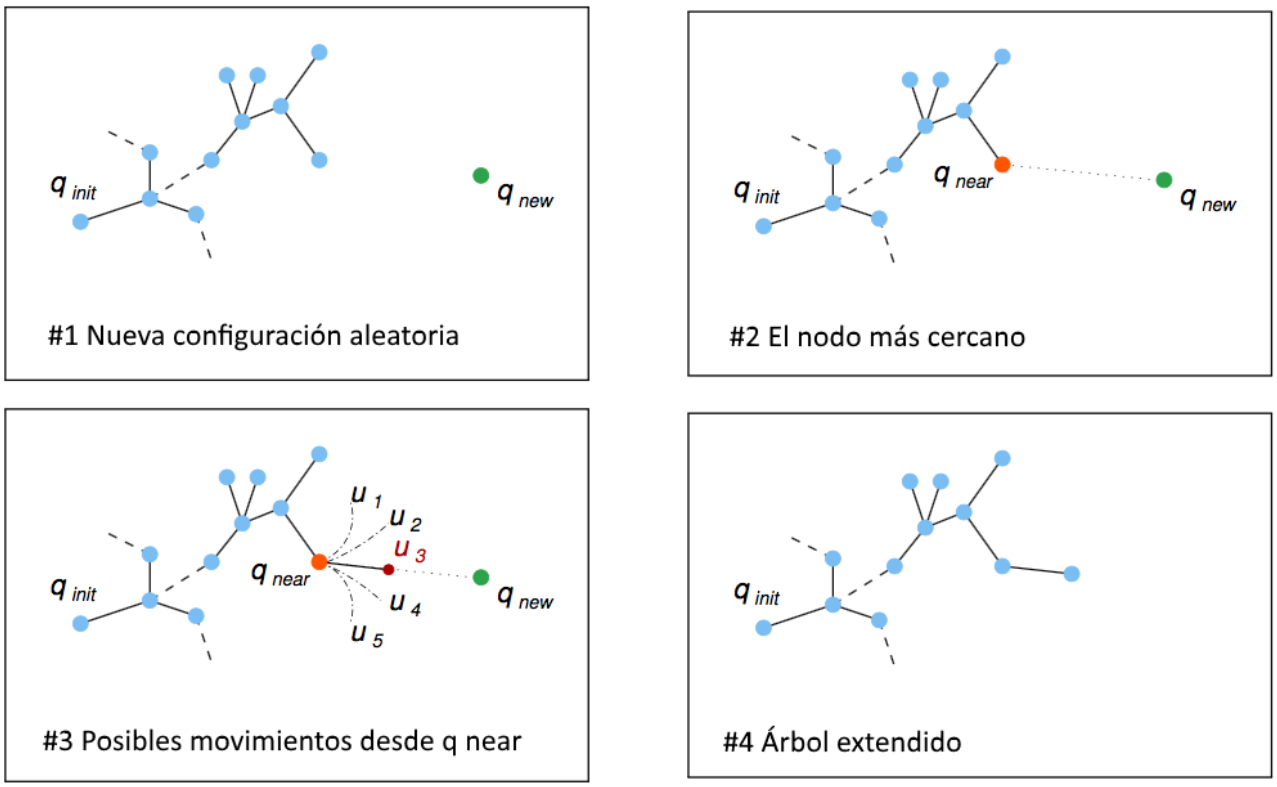


Figura 3.6: Ejemplo RRT

ejemplo, construyen soluciones de manera sucesiva para maximizar las mejoras disponibles localmente. La programación dinámica resuelve muchos subproblemas más pequeños antes de llegar a la solución completa final. Los métodos de ramificación y poda organizan el espacio de búsqueda en varios subespacios, luego buscan y eliminan algunos de estos de manera sistemática. Por el contrario, los métodos de búsqueda local, el algoritmo de recocido simulado (conocido en ingles como simulated annealing) y el proceso de búsqueda tabu completan las soluciones, y puede obtener una respuesta potencial (aunque probablemente no óptima) al detenerlos en cualquier iteración. Siempre tienen almacenada una única mejor solución actual que intentan mejorar en el siguiente paso. A pesar de estas diferencias, cada uno de estos algoritmos funciona o construye una sola solución a la vez. En general, el enfoque de mantener la mejor solución encontrada hasta el momento e intentar mejorarla es intuitivo, notablemente simple y, a menudo, bastante eficiente. Se pueden usar (1) reglas deterministas, por ejemplo, el algoritmo de escalada simple (conocido en ingles como hill climbing) usa la regla: si un vecino examinado es mejor, proceder a ese vecino y continuar buscando desde allí; de lo contrario, continuar buscando en el vecindario actual; (2) reglas probabilísticas, por ejemplo, el algoritmo de recocido simulado usa la regla: si un vecino examinado es mejor, aceptar esto como la nueva posición actual; de lo contrario, aceptar de forma probabilística esta nueva posición más débil o continuar buscando en el vecindario actual; o incluso (3) el historial de la búsqueda hasta

el momento, por ejemplo, la búsqueda tabu usa la regla: tomar el mejor vecino disponible, que no tiene que ser mejor que la solución actual, pero que no aparece en la memoria como un movimiento restringido o tabu [66].

¿Qué pasa si se abandona la idea de procesar una sola solución?. ¿Qué pasaría si en cambio se mantienen varias soluciones simultáneamente? Es decir, ¿qué pasaría si el algoritmo de búsqueda trabajara con una población de soluciones? Puede parecer que esta idea no proporciona nada realmente nuevo. Se puede procesar varias soluciones en paralelo si hay una computadora paralela disponible. Y si este fuera el caso, se puede implementar un método de búsqueda paralelo, como un recocido simulado paralelo o una búsqueda tabú paralela, donde cada procesador mantendría una solución única y cada procesador ejecutaría el mismo algoritmo en paralelo. Parecería que todo lo que se gana es la velocidad de cálculo: en lugar de ejecutar un algoritmo k veces para aumentar la probabilidad de llegar al óptimo global, una computadora con k procesadores completaría esta tarea en una sola ejecución, y por lo tanto en menos tiempo real. Pero hay un componente adicional que puede hacer que los algoritmos basados en poblaciones sean esencialmente diferentes de otros métodos de resolución de problemas: el concepto de competencia entre soluciones en una población.

Los Algoritmos Evolutivos (AE) son técnicas estocásticas que emulan el proceso de evolución natural de las especies para resolver problemas de optimización, búsqueda y aprendizaje [76]. Un proceso estocástico es aquel cuyo comportamiento es no determinista, en la medida que el subsiguiente estado del sistema está determinado tanto por las acciones predecibles del proceso como por elementos aleatorios. En los últimos 30 años, los AE han sido exitosamente aplicados para resolver problemas de optimización del mundo real en múltiples áreas de aplicación [77].

Un AE es una técnica iterativa (cada iteración se denomina generación) basada en emular el proceso de evolución natural de los seres vivos. En cada generación se aplican operadores estocásticos sobre un conjunto de individuos (la población). El esquema genérico de un AE que trabaja sobre una población P se puede ver en el Algoritmo 2.

Algoritmo 2 Esquema de un Algoritmo Evolutivo

```
1: t ← 0                                          ▷ contador de generación
2: INICIALIZAR(P(0))
3: EVALUAR(P(0))
4: while (no criterio de parada) do
5:     t ← t + 1
6:     P(t) ← SELECCIONAR(P(t − 1))
7:     ALTERAR(P(t))
8:     EVALUAR(P(t))
9: return mejor individuo hallado
```

El algoritmo evolutivo mantiene una población de individuos, $P(t) = x_i^t, \ldots, x_n^t$ para la iteración t. Cada individuo representa una solución potencial al problema en cuestión y, en cualquier algoritmo evolutivo, se implementa como una estructura de datos. Cada solución x_i^t; Se evalúa para dar

alguna medida de su $fitness$. Luego se forma una nueva población en la iteración $t+1$ seleccionando los individuos más aptos (el paso $seleccionar$). Algunos miembros de la nueva población sufren transformaciones (el paso $alterar$) por medio de operadores de variación para formar nuevas soluciones. Hay transformaciones únicas m_i, que crean nuevos individuos por un cambio en un solo individuo ($m_i : S \to S$), y transformaciones de orden superior c_j que crean nuevos individuos al combinar partes de varios (dos o más) individuos ($c_j : S \times \cdots \times S \to S$). El algoritmo se ejecuta hasta que se alcanza un criterio de detención predefinido, que puede incluir encontrar una solución adecuada, generar un número específico de soluciones candidatas o simplemente quedarse sin tiempo disponible.

La historia de la aplicación de este tipo de algoritmos se remonta a principios de la década de 1950 y, en realidad, fue inventada hasta diez veces por diferentes científicos, completamente independientes entre sí [78]. Cada procedimiento fue ligeramente diferente de los demás. Algunos de los investigadores en la década de 1960 le dieron a sus algoritmos nombres diferentes como algoritmos genéticos, estrategias de evolución o programación evolutiva, pero no todos eligieron bautizar sus invenciones. Con el tiempo, se desarrollaron técnicas más especializadas que se asociaron con estructuras de datos particulares, como los sistemas clasificadores y la programación genética. En la década de 1990, sin embargo, ha habido una considerable evidencia teórica y empírica de que ninguno de estos enfoques canónicos puede ofrecer algo que otro enfoque no pueda lograr. Además, como el interés en los algoritmos evolutivos ha aumentado rápidamente, las ideas se han tomado, intercambiado y modificado a través de todos estos enfoques. Como resultado, ya no existe ninguna base científica para discriminar entre los enfoques evolutivos. A la luz del estado actual de la técnica, se utiliza el término algoritmo evolutivo para describir cualquiera de los algoritmos basados en poblaciones que utilizan la variación y selección aleatoria.

3.4.1. Algoritmo genético

Basado en el esquema genérico de un AE que se muestra en el Algoritmo 2, los Algoritmos Genéticos (AG), establecidos por Holland (1975) [79], definen operadores de selección, recombinación y de mutación, y los aplican a la población de soluciones potenciales en cada generación. En una aplicación clásica de un AG, el operador de recombinación se utiliza para realizar la búsqueda (la explotación de las características de los individuos adecuados), mientras que la mutación se utiliza como el operador destinado a proporcionar diversidad para explorar diferentes zonas del espacio de búsqueda. La formulación más simple (llamado AG sencillo), realiza selección por rueda de ruleta, y utiliza el punto de cruce individual (SPX, por sus siglas en ingles) como operador de recombinación y un operador de mutación que modifica aleatoriamente posiciones seleccionadas en la codificación de la solución [80].

Conceptualmente, en la selección por ruleta, a cada miembro de la población se le asigna una sección de una rueda de ruleta imaginaria. Las secciones son de tamaño proporcional a la aptitud del individuo, de tal manera que el candidato más apto tenga la mayor porción de la rueda y el candidato menos apto tenga la más pequeña. La rueda es entonces girada y se selecciona el individuo asociado

con la sección ganadora. La rueda se hace girar tantas veces como sea necesario para seleccionar el conjunto completo de los padres para la próxima generación. En SPX se selecciona un punto de cruce en la representación de ambos padres. Todos los datos más allá de ese punto se intercambian entre las dos representaciones progenitoras. Los organismos resultantes son los hijos.

3.4.2. Algoritmo MOSES

El algoritmo MOSES [81] también está basado en el esquema genérico de un AE, pero utiliza solamente operadores de selección y mutación para la búsqueda de soluciones. Un esquema más detallado se puede ver en el Algoritmo 3. Sus creadores, Cercueil y Francois lo enmarcan dentro del conjunto de algoritmos de simulación Monte Carlo. Existen variaciones de este algoritmo siendo que en este trabajo se utiliza el denominado MOSES ordenado con reset. Ordenado por la forma en que se seleccionan los padres, ordenados según su fitness. Y con reset porque se vuelve a la probabilidad de mutación original una vez que se llega a un valor muy bajo. [82]

Algoritmo 3 Esquema de un Algoritmo MOSES

1: INICIALIZAR(P)
2: INICIALIZAR(T)
3: **while** not criterio de parada **do**
4: EVALUAR(P)
5: ORDENAR(P) ▷ Descendente según fitness
6: $p_{MUT} = e^{-1/T}$
7: Para cada $x_i \in P$ mutar con probabilidad p_{MUT}, si no muta reemplazar por $x_1 ... x_i$
8: $T = T * DecayFactor$
9: Si $T < ResetLimit$ entonces T = $T_{Inicial}$
10: **return** mejor individuo hallado

3.5. Programación orientada a agentes

La idea de la programación orientada a agentes (AOP, por sus siglas en ingles) fue iniciada por Shoham [83] como un nuevo paradigma de programación que combina el uso de las nociones mentalistas (como las creencias) para la programación (individual) de los agentes autónomos y una visión societal de la computación. En la década de los noventa, la mayor parte del trabajo se centró en unos pocos lenguajes que habían visto un trabajo teórico significativo, pero que tenían un uso limitado en la práctica, y sobre todo centrado en el desarrollo de agentes individuales, ya sea para un contexto de sistema multi-agente o no. Los trabajos en esta área a partir del 2000 cambiaron esta imagen sustancialmente ya que se produjeron muchos lenguajes de programación orientados a agentes basados en variados formalismos e inspirados por varios otros paradigmas de programación.

Muchos de estos lenguajes se desarrollaron seriamente, incluyendo plataformas de trabajo y herramientas de desarrollo. Esto llevó a que algunos de estos lenguajes tengan ahora bases de usuarios en crecimiento y sean utilizados en muchos cursos de Inteligencia Artificial (IA) y multi-agentes. En lugar de nombrar estos lenguajes individualmente, el lector interesado puede encontrar mayor información en [84] [85] donde muchos de estos lenguajes se han presentado en detalle.

La importante contribución de la programación orientada a agentes como un nuevo paradigma de programación fue proporcionar maneras de ayudar a los programadores a desarrollar sistemas autónomos. Por ejemplo, los lenguajes de programación orientados a agentes suelen tener construcciones de alto nivel que facilitan (en comparación con los lenguajes de programación tradicionales) el desarrollo de sistemas que están funcionando continuamente y reaccionando ante eventos que caracterizan cambios en los entornos dinámicos donde tales sistemas autónomos suelen funcionar. Los agentes deben asumir nuevas oportunidades o revisar los diferentes cursos de acción debido a los cambios en el ambiente. La programación orientada a agentes facilita el comportamiento de agentes que no sólo son reactivos sino también proactivos en el intento de lograr objetivos a largo plazo.

Las características de la programación orientada a agentes también hacen más fácil, en comparación con otros paradigmas, que los programadores puedan asegurar que los agentes se comportan de una manera que en la literatura de la programación orientada a agentes se denomina *racional*. Por ejemplo, si se adopta un curso de acción para lograr un objetivo (típicamente representado en el estado del agente explícitamente), si el agente se da cuenta de que la meta no se ha alcanzado, es de esperar que el agente tome medidas adicionales para lograr ese objetivo en nombre de su diseñador humano a menos que haya suficiente evidencia de que el objetivo ya no puede ser alcanzado o ya no es necesario.

Otra característica de la AOP, es que puede ser vista como una especialización de la programación orientada a objetos (OOP, por sus siglas en ingles). Un agente se define como una entidad cuyo estado está definido a partir de componentes mentales como creencias, capacidades, elecciones, compromisos, deseos, intenciones. Este estado mental puede describir a cualquier entidad y debe ser definido de forma coherente y útil, debe tener una semántica acorde al lenguaje de programación. En comparación con la OOP un agente sería un módulo, el estado mental sería el estado del módulo, estos estarían compuestos por creencias, capacidades y decisiones definidas con una sintaxis precisa para cada uno, con restricciones comunes entre sus contrapartes. Los agentes al ser usados para el

diseño de un sistema intercambian información, pedidos, ofertas, aceptan, declinan, compiten y se asisten mutuamente. Esto es parte de la definición de la teoría del acto de hablar que categoriza estas acciones y es utilizada en inteligencia artificial. Estos son los tipos de mensajes que intercambian los agentes, en la OOP no hay restricciones para los tipos de mensajes. Por otro lado los métodos en AOP deben ser consistentes para permitir la comunicación, mientras que en OOP no hay restricciones al respecto. Un factor importante en la AOP es el tiempo ya que el estado mental cambia con el transcurso del mismo.

En este trabajo se utilizaron dos plataformas de desarrollo, Jason y Jadex. Jason es una plataforma para el desarrollo de sistemas multi-agentes que incorpora un lenguaje de programación orientada a agentes. El lenguaje AgentSpeak, basado en la lógica e inspirado en *Belief-Desire-Intention* (BDI), inicialmente concebido por Rao [86], fue más tarde muy extendido en una serie de publicaciones de Bordini, Hübner, y colegas, para convertirlo en un lenguaje practico para la programación orientada a agentes. Estas extensiones llevaron a la variante de AgentSpeak que se puso a disposición en Jason [87]. La idea principal consiste en definir el *conocimiento* del programa en la forma de *planes* [88]. Los planes en AgentSpeak también se usan para describir respuestas a eventos, de esta forma es posible construir programas que no solo intentan cumplir con sus objetivos haciendo uso de la base de conocimientos provista sino que también pueden ser responsivos a los cambios en el ambiente. Los programas en AgentSpeak son llamados *agentes* haciendo hincapié en que son piezas activas de software, no proveyendo simples servicios sino tratando de cumplir con sus objetivos en su ambiente haciendo uso de su conocimiento. El framework no se preocupa de aspectos de bajo nivel (como sucede en los lenguajes procedurales) ni de proveer métodos (como típicamente se hace en lenguajes orientados a objetos) sino que se ocupa de la comunicación del conocimiento: como un agente puede comunicar sus creencias a otro, como un agente puede delegar uno de sus objetivos y su propio conocimiento. Los agentes son sistemas que existen en algún ambiente e interactúan con éste principalmente de dos formas: obteniendo información del ambiente mediante *sensores* y modificando el ambiente mediante *acciones*. La pregunta fundamental en torno al agente es como transformar las entradas obtenidas mediante los sensores en acciones concretas. Esto se logra en AgentSpeak mediante *planes*.

Jadex es un complemento a la plataforma de agentes JADE que busca salvar la brecha entre las plataformas que se concentran en la infraestructura de comunicación de agentes y las plataformas relacionadas con la representación de los conceptos de agentes internos. Sigue la arquitectura BDI, un modelo bien conocido para representar conceptos mentalistas en el diseño e implementación de sistemas. [89] Es parte de una plataforma llamada *Active Components* que tiene diversos componentes para el desarrollo de sistemas. Tiene una arquitectura orientada a servicios entre los componentes, con la ventaja de abstraer la lógica de negocios en agentes BDI y diagramas de flujo usando *Business Process Modelling Notation* (BPMN). Los componentes se comunican con servicios que cada uno de ellos describe. Los servicios se describen indicando que es lo que cada componente provee y sus requerimientos. Estos servicios utilizan la red lo que permite que puedan ejecutar de forma distribuida.

3.5.1. Modelo Belief-Desire-Intention

AgentSpeak (Jason en particular) y Jadex fueron inspirados, y su arquitectura y modelo de programación diseñados, en base a las ideas del modelo BDI. Los tres pilares del modelo se definen de la siguiente forma:

- Creencias (*Beliefs*), son la información que el agente tiene acerca del mundo. Esta información puede estar desactualizada o ser incorrecta.
- Deseos (*Desires*), consisten en los estados que el agente podría querer conquistar. La existencia de un deseo no significa que un agente vaya a actuar sobre el; sino que más bien este es un influenciador de las acciones del agente. Podemos ver los deseos como las distintas opciones disponibles para el agente.
- Intenciones (*Intentions*), son los planes que el agente ha decidido ejecutar y trabaja activamente por conseguir de todas las opciones disponibles.

3.5.2. Metodología TROPOS para desarrollo de sistemas AOP

Es una metodología estructurada en cinco fases que permiten representar el qué, el cómo y el por qué de un sistema AOP. Además abarca desde el modelado hasta la implementación del sistema, proporcionando mecanismos para definir dependencias y requisitos funcionales y no funcionales. Estas cinco fases del desarrollo se definen de la siguiente forma:

- Análisis de requisitos tempranos: consiste en identificar y analizar los actores del sistema y sus intenciones. Estos actores sociales dependen de otros para alcanzar sus metas, ejecutar planes, proveer recursos y ofrecer servicios.
- Análisis de requisitos tardíos: este análisis se enfoca en el desarrollo del sistema de software. En esta fase el modelo organizacional obtenido en la fase anterior se extiende para agregar el sistema como actor. Esto introduce nuevas dependencias entre los actores que definirán los requisitos funcionales y no funcionales.
- Diseño arquitectural: esta fase define la arquitectura del sistema global, definiendo subsistemas (actores) interconectados mediante flujos de control y datos (dependencias).
- Diseño detallado: en esta esta se definen las creencias, metas y capacidades de los agentes. Además se especifica la comunicación entre ellos.
- Diseño e implementación: según la herramienta a utilizar se basa en el diseño detallado paso a paso para la correspondencia entre la plataforma donde se implemente de forma natural.

Para esto existen dos diagramas que complementan la metodología. A continuación una breve definición de los componentes de estos diagramas.

Actor: entidad que tiene metas e intenciones dentro del sistema o conjunto organizacional. Un actor por ejemplo puede ser una persona o un sistema de software, así como un rol o una posición. En TROPOS las metas deben estar asociadas a actores que puedan satisfacerlas. La representación gráfica de un actor es un círculo y sus límites están representados por otro círculo de líneas punteadas. Si la vista del modelo no requiere presentar los elementos (metas, tareas o recursos) internos del actor es posible representar al actor como un círculo de línea continua.

Agente: es un caso particular de actor, algo más concreto. Tiene dependencias que aplican independientemente del rol que pueda desempeñar. Este concepto de agente es utilizado para referirse tanto a componentes de software como de hardware.

Rol: es una abstracción del comportamiento de un actor social dentro de un contexto o dominio. Puede tener dependencias asociadas que se aplican independientemente de quien juegue ese rol.

Posición: usualmente es un conjunto de roles asignados junto a un agente, es una abstracción intermedia entre un rol y un agente. Se dice que un agente ocupa una posición.

Asociación: define un conjunto de roles, posiciones y agentes, estos se interconectan mediante relaciones.

Capítulo 4

Presentación del problema y estado del arte

Este capítulo presenta el problema e introduce las principales características de los dos enfoques propuestos: planificación fuera de línea mediante uso de Algoritmos Evolutivos y planificación en línea con Programación Orientada a Agentes. Se presenta luego un conjunto de reseñas de trabajos relevantes, algunos de los cuales fueron utilizados como base para el desarrollo de los algoritmos y las heurísticas propuestas para el control del vuelo.

4.1. Presentación del problema

Este trabajo presenta un enfoque para la resolución del problema de búsqueda y vigilancia de objetivos mediante una flota de vehículos aéreos no tripulados. El problema que se intenta resolver consiste en la asignación de caminos a una flota de drones con el fin de maximizar la superficie explorada por los mismos, a la vez que se mantiene la conectividad entre los integrantes de la flota y la vigilancia y responsividad ante cambios en el entorno. De esto surgen los tres objetivos que se intenta satisfacer: exploración, conectividad y vigilancia. Es por las distintas características necesarias para cumplir con cada uno de los objetivos antes mencionados, que resulta conveniente dividir el problema en dos subproblemas. Por ejemplo, el objetivo de exploración puede ser satisfactoriamente resuelto de manera fuera de línea, ya que se conoce de antemano el área que los drones deben explorar. Esto hace que no se tengan restricciones acerca de la rapidez que debe tener la técnica propuesta para resolver el problema relacionado con este objetivo. Por otra parte, para el objetivo de vigilancia, se necesita tomar decisiones en tiempo real por lo cual la eficiencia de la técnica utilizada es crucial para su éxito. Otra diferencia importante radica en el tamaño y los requisitos que el software utilizado en cada caso requiere. Como fuere mencionado anteriormente, el algoritmo para el cálculo de rutas puede ejecutarse fuera de línea, por lo que no existen restricciones en cuanto a su tiempo de ejecución o limitantes de hardware. Por el contrario, el algoritmo para vigilancia debe ser rápido y capaz de correr en el hardware disponible en los agentes, el cual es limitado comparado a las opciones disponibles fuera de línea. El objetivo de conectividad se encuentra en un punto intermedio con respecto a los otros; es posible contemplar la conectividad entre los agentes al momento de calcular

las rutas de exploración (fuera de línea) pero se requiere tomar en consideración las modificaciones que los agentes en el campo realizan sobre sus rutas para la vigilancia en tiempo real. La Figura 4.1 muestra gráficamente donde se ubica cada objetivo desde el punto de vista de la disponibilidad de información y el tiempo de cómputo necesario para su resolución.

Por las razones expuestas anteriormente es que resulta conveniente trabajar en una división del problema original en dos subproblemas. Por un lado, el subproblema fuera de línea que consiste en calcular un conjunto de caminos para un grupo de drones de forma tal de maximizar la superficie explorada manteniendo, en la medida de lo posible, la conectividad. Detalles sobre este subproblema y la forma de resolución se encuentran en la sección 4.1.2. Por otro lado se tiene el subproblema en línea, para el cual se busca proveer a los drones de la capacidad de encontrar y vigilar objetivos en tiempo real. Una explicación más detallada sobre este subproblema se encuentra en la sección 4.1.3. Por más información sobre la formalización del problema ver el capítulo 5.

4.1.1. Conceptos generales

El enfoque combinado (*i.e.* la división en los dos subproblemas mencionados en la sección precedente) que se propone en este trabajo permite disponer de planes de vuelo eficientes y versátiles, teniendo a la planificación fuera de línea como un sólido punto de partida e incluyendo la flexibilidad para adaptarse a las situaciones cambiantes del ambiente proporcionada por la AOP. Los objetivos del enfoque, a grandes rasgos, son los siguientes: maximizar la superficie explorada, mantener la comunicación entre los agentes tanto como sea posible y vigilar a los objetivos. A los efectos de este proyecto se trabaja sobre una abstracción del problema en la que no se consideran aspectos físicos de los drones ni del vuelo. Por ejemplo, no se tiene en cuenta el radio de giro del dron o si existe viento o cualquier otro fenómeno climático. Se entiende que, si bien estos aspectos no son irrelevantes y su análisis debe ser incorporado para la implementación realista de este tipo de algoritmos, la presente propuesta aporta valor en el modelado y resolución del problema de búsqueda y vigilancia de objetivos mediante una flota de drones. El problema de convertir las rutas generadas en base a la abstracción mencionada, en rutas aplicables a un problema del mundo real, puede ser abordado como un problema independiente (por ejemplo, se podría aplicar un proceso de *path smoothing* con curvas de Dubins [90] para las rutas computadas).

Entrando en detalle acerca de los tres objetivos principales (exploración, conectividad y vigilancia), se considera que:

- La superficie explorada es el área de la unión de las superficies exploradas por cada dron en su recorrido en un tiempo arbitrario establecido. El área cubierta por un dron en un instante dado es la de la circunferencia definida por el *radio de cobertura* que lo tiene como centro.

- Mantener la comunicación entre los agentes implica que existe una distancia máxima a la que pueden encontrarse. Se considera que dos drones se comunican entre si cuando la distancia entre ellos es menor que el *radio de comunicación*.

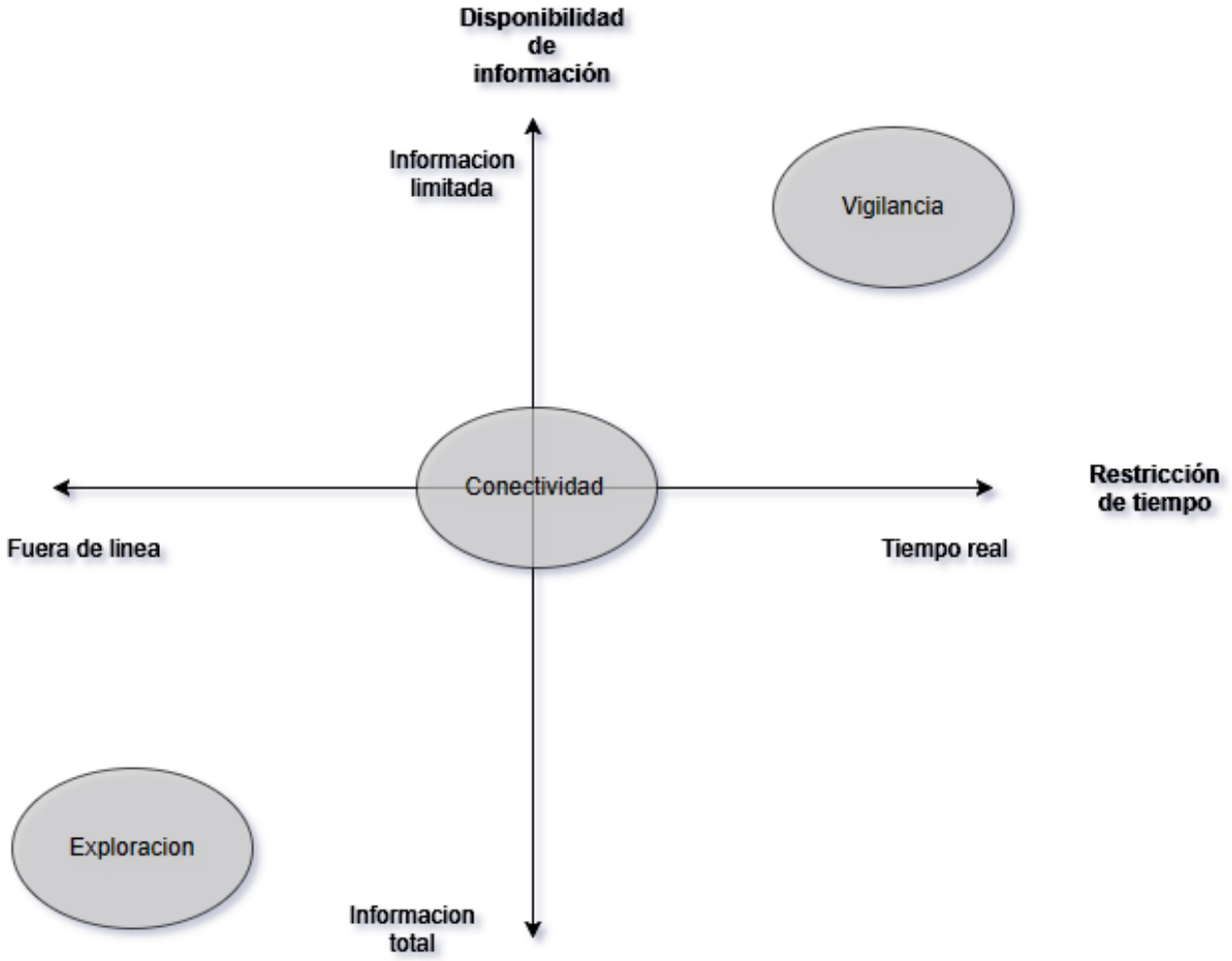


Figura 4.1: Disponibilidad de información y requerimiento de tiempo, para cada uno de los objetivos.

- En la versión en línea del problema, encontrar y vigilar a los objetivos implica que los drones pueden alterar su ruta precalculada con el objetivo de seguir entidades que aparezcan en su *radio de cobertura*.

4.1.2. Problema fuera de línea

El subproblema de planificación estática (fuera de línea) se aborda mediante algoritmos evolutivos para hallar una serie de puntos de vuelo para cada integrante de la flota, con el objetivo de lograr buenos valores de compromiso entre el tamaño del área explorada, la cercanía de los drones (*i.e.* conectividad) y la vigilancia de ciertos puntos de interés preestablecidos. Los datos de entrada para este problema son: una cierta área que se desea explorar, la cantidad de drones disponibles para

la misión, un conjunto de puntos de interés en el área a explorar y una base.

Las soluciones generadas por el algoritmo deben cumplir las siguientes características:

- Las rutas generadas deben comenzar y terminar en la base. Esto es una restricción dura, aquellas soluciones que no satisfagan esta restricción serán descartadas.
- La base es el punto en el que los drones pueden recargar sus baterías. En las soluciones válidas al problema, todos los drones deben tener una cantidad no nula de energía mientras estén fuera de la base.
- La conectividad entre los integrantes de la flota es una propiedad deseable de la solución, pero no mandatoria. Las soluciones con mejores niveles de conectividad serán promovidas a través de un mayor valor de fitness.

La formalización del problema fuera de línea se encuentra en la sección 5.1.1.

4.1.3. Problema en línea

El segundo algoritmo propuesto resuelve el subproblema de búsqueda y vigilancia de objetivos de forma autónoma y cooperativa aplicando AOP, partiendo de la planificación fuera de línea hallada por el algoritmo evolutivo. El algoritmo en línea toma como entrada el conjunto de rutas generadas por el algoritmo fuera de línea. La función del algoritmo en línea consiste en realizar las modificaciones necesarias a esas rutas precalculadas de manera de contemplar los cambios en el ambiente en tiempo real. El vuelo está marcado por los puntos definidos en la planificación fuera de línea y los drones inician con una carga completa de batería que se va descargando a medida que recorren el campo. Si la batería entra en un nivel muy bajo los drones deben regresar a la base para ser cargada. Cuando un dron detecta un intruso en su rango de visión la flota debe tomar una decisión sobre cual de los drones debe vigilarlo o cómo se determinará su seguimiento. Con el propósito de lograr una vigilancia cooperativa, los drones deben enviar mensajes a todos los drones de la flota que se encuentren en su rango de comunicación. Los drones deben esperar recibir información de los drones en su rango de comunicación y deben tener la capacidad de reaccionar a dichos mensajes.

Algunos ejemplos de cambios en el ambiente que podrían ocasionar un recálculo de rutas son:

- Aparición de intrusos en el campo de visión de alguno de los drones. Esto podría generar la necesidad de vigilancia y por ende la modificación de las rutas de uno o más drones destinados a esa tarea.
- Recepción de mensaje desde otro dron. Ya sea por baja batería o para pedir soporte en una tarea de vigilancia, un mensaje recibido puede ocasionar el recálculo de la ruta para el dron.
- Pérdida de conectividad. Un dron podría modificar su ruta para intentar recuperar la conectividad con el resto de la flota.

La formalización del problema en línea se encuentra en la sección 5.2.1.

4.2. Relevamiento del estado del arte

El análisis de la literatura relacionada permite identificar varias propuestas recientes sobre la aplicación de estrategias computacionales para la planificación de vuelo de una flota de drones. Los principales trabajos relacionados se describen a continuación. Una de las primeras propuestas de aplicación de inteligencia computacional para planificar el vuelo de drones fue presentada por Ponda et al. [91]. Los autores estudiaron métodos para diseñar trayectorias que permitan incrementar la cantidad de información provista por un conjunto de medidas obtenidas de un objetivo, realizadas por sensores a bordo, para determinar la posición de este. Se exploró la utilización de algoritmos de estimación para la localización de objetivos, incluyendo filtro extendido de Kalman y aplicando la matriz de información de Fisher y la cota de Cramér-Rao para evaluar el rendimiento de la estimación de medidas. Se identificó a la optimización de la trayectoria del dron como un factor de gran importancia para la localización de objetivos usando medidas basadas en imágenes, dado que los resultados de la estimación resultan severamente sesgados si las medidas no proveen información nueva, impidiendo alcanzar una estimación apropiada. El análisis experimental consideró dos escenarios principales: la identificación de puntos para realizar las mediciones considerando un número máximo de medidas y la optimización de trayectorias de un único dron con restricciones de movimiento. Ambos escenarios aplican un método de descenso de gradientes para la optimización, que presenta algunos inconvenientes propios de una técnica determinista (por ejemplo, tiende a agrupar medidas). Se plantea también un caso de estudio optimizando simultáneamente las trayectorias y la estimación de objetivos, para los casos de objetivo estático y objetivo que sigue una trayectoria predeterminada. Los resultados reportados muestran que las trayectorias calculadas permiten mejorar la estimación, recabando la misma cantidad de información que una estrategia no optimizada tan solo con la mitad de las medidas. El trabajo no plantea la aplicación a la planificación de vuelo de flotas de drones, aunque el primer escenario puede utilizarse para coordinar las medidas provistas por varios drones para mejorar las estimaciones de movimiento del objetivo.

Mufalli et al. [92] estudiaron el problema de selección de sensores y de planificación de rutas de drones en misiones de reconocimiento militar, que es una generalización del problema de orientación de equipos (*Team Orienteering Problem*, TOP por sus siglas en ingles). Los autores presentaron un modelo de programación matemática y una metodología de resolución exacta (utilizando CPLEX) que solo permitió resolver misiones muy simples. Para abordar misiones más realistas, los autores propusieron dos heurísticas, aumentadas mediante la técnica de generación de columnas. Los resultados obtenidos sobre escenarios de hasta 100×100 posiciones con hasta 100 objetivos y flotas de hasta 8 drones indicaron que las heurísticas fueron capaces de hallar buenas soluciones rápidamente. La generación de columnas mejoró la solución en muchas instancias, con un impacto mínimo en el tiempo de ejecución.

Schleich et al. [93] propusieron un enfoque descentralizado y localizado para controlar la movilidad en flotas de drones dirigidos desde una base. El problema abordado corresponde a la planificación de misiones de vigilancia con restricciones de conectividad entre los drones y la base. Se tiene un área que necesita ser monitoreada y una flota de drones encargada de patrullarla. La flota se despliega desde la estación base y todos los drones son equipados con módulos de comunicación (con la base y

con otros drones). La principal contribución del trabajo es el diseño de un modelo de movilidad, llamado *connected coverage model*, que es el algoritmo responsable de mover físicamente los drones para cumplir la misión de vigilancia. El modelo se compone de tres pasos secuenciales: la selección del vecindario con el que cada dron intentará mantenerse conectado, el cálculo de alternativas viables para la posición futura del dron y la construcción de un comportamiento basado en feromonas para elegir la mejor dirección dentro del conjunto de alternativas. El modelo de movilidad se compara con modelos *random mobility model*, que selecciona la dirección de forma aleatoria, y *pheromone-based mobility model* que selecciona la dirección a seguir según el nivel de feromonas registrado en un mapa de cuadrantes. Para comparar los modelos de movilidad se proponen métricas de calidad para cuantificar diferentes aspectos del proceso de cobertura del área (*i.e.*, velocidad, exhaustividad y equitatividad de la cobertura) y el mantenimiento de la conectividad. La comparación se realizó mediante 30 ejecuciones de una simulación implementada a medida usando una biblioteca de código abierto para representar grafos y un autopiloto que simula un UAV de ala fija con velocidad máxima y mínima y con un ratio de giro máximo. Los resultados numéricos obtenidos muestran que el mantenimiento de la conectividad resulta significativamente mejor que el de los otros modelos. El trabajo de Schleich et al. presenta un enfoque similar al que se propone en este trabajo, pero sin aplicar inteligencia computacional y sin el componente de navegación reactiva proporcionado por la programación orientada a agentes.

Oh et al. [94] presentaron un framework para planificar de modo integral la selección de rutas para coordinar drones aplicando conceptos de geometría diferencial. El algoritmo propuesto construye caminos seguros y factibles para visitar un conjunto de puntos (intersecciones de vías de tránsito y puntos notables) en misiones cooperativas de múltiples drones, de modo eficiente. Las trayectorias consideradas contemplan mantener la comunicación entre cada UAV y una estación de control terrestre que controla centralmente toda la misión. El problema se modela como una variante del problema del cartero chino y se plantean tres métodos para su resolución: i) un método exacto basado en programación líneal entera-mixta mediante una formulación de problema de la mochila multidimensional de múltiple elección para asignar caminos a los drones minimizando el tiempo total de vuelo; ii) una heurística de inserción de elementos cercanos para construir rutas agregando segmentos de vías de tránsito considerando su costo de inserción y iii) un modelo de negociación por subasta para resolver conflictos entre zonas visitadas por más de un UAV en su recorrido, para minimizar el costo de los planes de vuelo. La validación del framework se realizó considerando dos casos de estudio: la planificación de una misión de vigilancia y búsqueda sobre una red vial y la planificación de rutas para mantener la comunicación con una base de comando, respetando restricciones de velocidad de vuelo y evitando zonas de vuelo prohibidas (por ejemplo, aeropuertos, grandes estructuras, etc.). Para la planificación de misiones se trabajó sobre un mapa de 10×10 vértices, generado aleatoriamente y aplicando simulaciones Monte Carlo. La heurística de inserción obtuvo caminos 12 % más largos que los del método exacto, pero fue efectiva para el cálculo en línea de trayectorias, hallando caminos en menos de un segundo. El modelo cooperativo se analizó sobre un escenario base de 100×100km, con solo dos drones y una base central, considerando un radio de comunicación de 20 km, una zona de vuelo prohibida y una velocidad máxima para los drones de

50 m/s. Los principales resultados indicaron que el modelo de negociación fue efectivo para generar planes de vuelo para vigilar el escenario considerado y evitar las zonas prohibidas. Sin embargo, se perdieron las comunicaciones por un tiempo total de 1200 s, resultado que sugiere que para realizar exitosamente la misión es necesario contar con una flota mayor. El trabajo presenta un enfoque centralizado por la existencia de la base de control, por lo cual los planes de vuelo no son totalmente autónomos.

Bekmezci et al. [95] presentaron una heurística para resolver el problema de planificación *basic task allocation planning problem* con la restricción de que los agentes, en este caso drones, deben mantener una *Flying Ad Hoc Network*. El problema consiste en mantener la red ad-hoc mientras realizan tareas de forma que el beneficio global sea maximizado (o el costo total sea minimizado) dado un conjunto de drones y tareas a realizar. El planteo del problema asume que los drones son idénticos y navegan a una velocidad constante. Además se supone que todos los drones tienen suficiente energía para completar toda la misión, que al comenzar la misión la red de drones se encuentra conectada, que cada dron conoce su propia posición durante la misión con la ayuda de su GPS, que cada dron conoce también la ubicación exacta de los demás drones, que los drones pueden cambiar su orientación y movimiento rápidamente y que las alturas de los drones son ligeramente diferentes para evitar colisiones. El terreno se modela en dos dimensiones, todos los objetivos son predefinidos y conocidos por los drones y se considera que si la distancia euclidiana entre dos drones es menor que un cierto rango predefinido de comunicación, los dos drones están conectados directamente. Se presentan resultados de una simulación en MATLAB para estudiar el desempeño del algoritmo (evaluado por el tiempo de ejecución) y el comportamiento de la flota, variando el número de UAV, la cantidad de objetivos y el rango de comunicación. Los resultados indican que cuando el rango de comunicación es muy bajo el rendimiento del algoritmo se degrada para cualquier cantidad de drones. Sin embargo, en un cierto nivel, la contribución se vuelve muy limitada. A medida que la cantidad de drones aumenta, el tiempo requerido para la misión disminuye de forma significativa. El trabajo presenta una simulación interesante pero muy restringida por el amplio número de suposiciones del escenario planteado.

Shang et al. [96] propusieron un algoritmo genético híbrido con un algoritmo de colonia de hormigas (ACO, por sus siglas en ingles) para planificar misiones de una flota de drones. El problema se modela como un TOP, en el que los nodos corresponden a los objetivos, los arcos entre nodos representan la ruta de vuelo entre objetivos, y se toma en cuenta el beneficio por vigilar un objetivo y las limitaciones de combustible/energía de los drones son tratadas como el límite de longitud en el TOP. La idea básica del algoritmo es reemplazar los malos individuos de la población del algoritmo genético por nuevos individuos construidos mediante un algoritmo ACO. También se aplica un operador de recombinación que incluye Path Relinking para el cruzamiento. El algoritmo aplica una estrategia de partición de la población para mejorar la eficiencia de la evolución. Los resultados experimentales sugieren que el algoritmo híbrido propuesto puede resolver instancias de prueba de forma efectiva en un tiempo razonable. La comparativa con algoritmos existentes muestran que el algoritmo propuesto es competitivo y prometedor. Al comenzar el algoritmo, los parámetros, la información heurística y los rastros de feromonas son inicializados y una población de tamaño

predeterminado es generada. La población se divide en dos: un conjunto de individuos élite y un conjunto de individuos no-élite. En cada iteración, para cada individuo no-élite, se elige aleatoriamente un individuo élite para recombinarlos. Después del cruzamiento, la mutación y los procedimientos de búsqueda local, se genera un descendiente y se actualiza la población. Finalmente se actualizan los rastros de feromona de acuerdo a la regla establecida para tal fin. El algoritmo termina de iterar cuando se alcanza un número máximo de ciclos. El algoritmo propuesto fue estudiado en casos de prueba seleccionados, realizando 10 ejecuciones independientes, y comparando con varios algoritmos de la literatura: *Guided Local Search*, ACO, *Slow Variable Neighbourhood Search* y *Slow Path Relinking*. El análisis experimental indicó que el algoritmo propuesto fue capaz de obtener mejores resultados en varias de las instancias y obtener soluciones óptimas en la mayoría, sugiriendo que el algoritmo puede utilizarse en problemas de gran escala.

Han et al. [97] propusieron un sistema multi agente que integra cada dron para formar un equipo que puede colaborar para realizar misiones complejas con mejor eficiencia. El sistema trata a cada dron como un objeto de control. Para lograr un vuelo coordinado, implementar la cooperación para alcanzar la misión y aumentar la robustez del MAS, los autores proponen varios mecanismos: un mecanismo de rastreo para seguir objetivos, un mecanismo de *artificial potential field* [98], y un mecanismo de subasta para alocar y seleccionar misiones.

Los autores aseguran que se prueba, a través de simulaciones, que el sistema multi agente desarrollado aumenta el grado de éxito y robustez, mientras mantiene el vuelo coordinado y la factibilidad de la misión, pero no presentan experimentos reales que validen estas afirmaciones. El análisis de los trabajos relacionados permite identificar varias propuestas para el desarrollo de algoritmos y modelos de movilidad inteligentes que permitan optimizar los resultados de una flotilla de UAV en diferentes misiones. Estas propuestas ayudaron a modelar el problema e inspirar las soluciones desarrolladas. A su vez se presenta una implementación en el mundo real de una solución fuera de línea utilizando dos drones Parrot Bebop 2.

Capítulo 5

Metodologías de resolución

En este capítulo se muestran los aspectos técnicos fundamentales y los detalles relacionados a la implementación del proyecto tanto para la versión fuera de línea como para la versión en línea. Se desarrollan además las características y aspectos particulares de los frameworks utilizados. Finalmente se describe como fueron empleados y que consideraciones se tuvieron en cuenta para su utilización en el contexto del presente trabajo.

5.1. Planificación fuera de línea

5.1.1. Formulación matemática

A continuación se presenta la formulación matemática del problema de planificación de rutas de vuelo para drones en misiones de vigilancia. Dados los siguientes elementos:

- un conjunto de drones $U = \{u_1, \dots u_U\}$
- un período de tiempo de misión T discretizado uniformemente en s instantes de tiempo; $T =< t_1, t_2, \dots t_s >$
- una velocidad máxima de desplazamiento de los drones v_D
- un conjunto de objetivos a vigilar $O = \{o_1, \dots, o_O\}$
- una velocidad máxima para los objetivos v_O
- un vector P que define el beneficio asociado a cada uno de los objetivos $P =< p_1, p_2, \dots p_O >$, donde p_i indica el beneficio obtenido por vigilar al objetivo o_i
- una matriz OP de dimensiones $O \times s$ que indica la posición de los objetivos en cada instante de tiempo. OP_{ij} indica las coordenadas del objetivo O_i en el instante t_j.

- un radio de cobertura r_O
- un radio de comunicación r_C
- un área a explorar A, de dimensión $H \times W$, discretizada en cuadrados de lado r_O. El área A queda definida por la siguiente matriz:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1(W/r_O-1)} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2(W/r_O-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{(H/r_O-1)1} & a_{(H/r_O-1)2} & a_{(H/r_O-1)3} & \cdots & a_{(H/r_O-1)(W/r_O-1)} \end{bmatrix}$$

- una *base* B, definida por sus coordenadas (x_B, y_B), desde la cual parten los drones para realizar sus misiones. La base B se encuentra dentro del área a explorar, i.e., $(x_B, y_B) \in A$.

El problema consiste en hallar una planificación para la flota de drones, i.e., una función $p : U \times T \rightarrow A$, que maximice simultáneamente las funciones objetivo dadas por las Ecuaciones 5.1, 5.2, 5.3.

Se define la función de beneficio por tener visión de un objetivo como:

$$\delta(p) = \sum_{i=0}^{u-1} \sum_{j=0}^{t-1} \sum_{z=0}^{o-1} found(pos(u_i, t_j), pos(o_z, t_j)) \times p_z \tag{5.1}$$

donde:

$$found(pos1, pos2) = \begin{cases} 1 & \text{si } d(pos1, pos2) \leq r_o \\ 0 & \text{en caso contrario} \end{cases}$$

$$pos(u_i, t) = a_{i\lfloor t/s \rfloor} + vel(u_i, t) \times t$$
$$vel(u_i, t) = (a_{i(\lfloor t/s \rfloor + 1)} - a_{i\lfloor t/s \rfloor}) \div s$$

Se define la función de beneficio por formar una red *ad-hoc* como:

$$\gamma(p) = \sum_{i=0}^{u-1} \sum_{z=0}^{u-1} \sum_{j=0}^{t-1} connected(pos(u_i, t_j), pos(u_z, t_j)) \text{ con } i \neq z \tag{5.2}$$

donde:

$$connected(pos1, pos2) = \begin{cases} 1 & \text{si } d(pos1, pos2) \leq r_c \\ 0 & \text{en caso contrario} \end{cases}$$

La función de beneficio por explorar se define como:

$$\phi(p) = \sum_{i=0}^{u-1} \sum_{j=0}^{t/s-1} explored(p_{ij}) \times \frac{1}{{r_o}^2} \tag{5.3}$$

donde:

$$explored(pos) = \begin{cases} 1 & \text{si } a_{(pos.y/r_o)(pos.x/r_o)} = 1 \\ 0 & \text{en caso contrario} \end{cases}$$

Es de interés destacar que dada la definición de ϕ, el beneficio obtenido es inversamente proporcional al tamaño del área cubierta. A este indicador de rendimiento se lo conoce también como *Spatial Exploration Ratio* (SER).[99]

5.1.2. Diseño de la solución

El diseño de la solución implica bajar a tierra todo el análisis realizado en etapas anteriores de forma de poder crear una pieza de software concreta que resuelva los problemas planteados. Dos etapas claves en este proceso son la elección de un ambiente tecnológico especifico y el diseño de cada uno de los elementos que componen la solución en ese ambiente particular. La primer decisión que se debió tomar consistía en si desarrollar todo el aplicativo desde cero o utilizar alguno de los frameworks de algoritmos evolutivos disponibles. Se escogió el framework Watchmaker[100] porque es multiplataforma y orientado a objetos, tiene buena y clara documentación y es fácilmente extensible.

Watchmaker

El framework Watchmaker es una plataforma de alto nivel, orientada a objetos y desarrollada en el ecosistema Java que provee un mecanismo para llevar a la práctica los conceptos de algoritmos evolutivos que han sido desarrollados en los capítulos anteriores. Existen dos pre-requisitos fundamentales que van más allá del framework particular que se utilice: la capacidad de representar las soluciones candidatas y la existencia de un mecanismo que permita evaluar la calidad de las mismas. [100]

Componentes de Watchmaker

El componente fundamental del framework es el *Evolution Engine*. El framework provee varias implementaciones del mismo, pero el más usado (y el que fue utilizado en este trabajo) es el *GenerationalEvolutionEngine*. Para completar la implementación se deben proveer algunos otros artefactos, como ser una implementación de la interface *FitnessEvaluator*, la cual es específica de cada problema y determina la forma en que se calcula el fitness. Otros componentes necesarios incluyen, *CandidateFactory*, que determina la forma en que los individuos candidatos son generados y *EvolutionaryOperators*, que determina las operaciones que estan disponibles sobre la población.

Una vez determinados los componentes fundamentales, muchos de los cuales poseen implementaciones por defecto, el framework se encarga de hacer el trabajo pesado por nosotros, inicializando la población, aplicando los operadores evolutivos sobre ella, etc.

5.1.3. Diseño de los algoritmos evolutivos propuestos

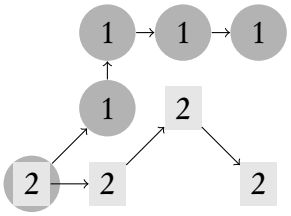
Esta sección presenta dos AE aplicados al problema: un AG tradicional, y un algoritmo MOSES en su variante ordenada con reset. Ambos métodos utilizan una inicialización y operadores especiales para buscar un buen patrón de exploración. Los detalles se presentan en las siguientes subsecciones, junto con la biblioteca de software utilizada para implementar los AE.

Representación de las soluciones

Los individuos codifican la posición de cada UAV en cada instante de tiempo. La representación elegida es la de una matriz de dimensiones $u \times s$ (*i.e.*, cantidad de UAVs $\times$ intervalos discretos de tiempo), donde cada elemento (i, j) de la matriz consiste en una dupla (x_{ij}, y_{ij}) que representa la posición de cada UAV en un sistema de coordenadas cartesiano. A modo de ejemplo, el siguiente individuo representa la secuencia de movimientos por parte de dos UAV que se ilustra en la Figura 5.1.

$$\begin{bmatrix} (0,0) & (1,1) & (1,2) & (2,2) & (3,2) \\ (0,0) & (1,0) & (1,0) & (2,1) & (3,0) \end{bmatrix}$$

Figura 5.1: Ejemplo de dos UAVs



5.1.4. Detalles de implementación

A continuación se presentan los detalles de implementación de los métodos estudiados, describiendo la inicialización, selección, recombinación y mutación de los algoritmos.

Función de fitness

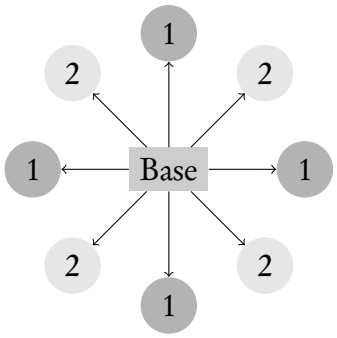
La función de fitness se define simplemente como la combinación líneal de las funciones de beneficio presentadas en la formulación matemática:

$$fitness(p) = \delta(p) + \gamma(p) + \phi(p)$$

Inicialización

Cada individuo de la población inicial se genera de acuerdo al procedimiento que se detalla a continuación. Se establece la ruta de cada UAV generando un camino que inicia en la posición de la base y se aleja hacia una determinada dirección a velocidad máxima. Esta dirección queda determinada por el angulo $i \times \frac{2\pi}{u} + j \times \frac{2\pi}{u \times Z}$, donde $i \in [0, u-1]$ es el número del UAV, $j \in [0, Z]$ es el número del individuo y Z la cantidad de individuos (tamaño de la población). De esta manera cada solución candidata es una rotación de otra. Por ejemplo, si se tienen cuatro UAV y una población de dos individuos ($Z = 2$), uno ira hacia el Este, otro hacia el Sur, otro hacia el Oeste y otro hacia el Norte (90 grados entre cada vector movimiento) para el primer individuo. Como se observa en la Figura 5.2, el segundo individuo corresponde a una rotación de 45 grados del primero. En el caso de que un UAV salga fuera del área de exploración, su trayectoria es corregida invirtiendo su sentido.

Figura 5.2: Ejemplo de inicialización



Selección

Se utiliza la técnica de Muestreo Estocástico Universal con Escalado Sigma para la selección. [101] Esta técnica resulta una alternativa a los métodos de selección basados en fitness más básicos como por ejemplo selección por Rueda de ruleta. Las ventajas de utilizar Escalado Sigma son básicamente dos:

- En la etapa inicial del algoritmo, ayuda a evitar la convergencia prematura causada por la dominancia de un grupo de individuos con fitness muy alto (en comparación con el resto de la población).
- En la etapa final, ayuda a amplificar pequeñas diferencias de fitness cuando el ritmo de mejora se ha enlentecido.

Explotación, cruzamiento

Para el cruzamiento se utiliza SPX. Luego de un cruzamiento puede que se deban arreglar los caminos de los UAV ya que es posible que no lleguen a una posición producto de un cruzamiento desde la posición anterior. Este procedimiento se detalla más adelante.

Exploración, mutación

Es utilizado un operador de mutación en el que se usa una variable aleatoria para determinar si se debe aplicar la mutación a un individuo o no. En el caso de que se deba aplicar se sortea un número aleatorio en el intervalo $[1, u \times \frac{t}{s}]$. Este número se utiliza para determinar la cantidad de puntos a mutar y cada punto se sortea aleatoriamente.

La mutación consiste en cambiar la dirección del UAV a la dirección perpendicular a la actual (se calcula la dirección a partir de la posición actual y la posición del siguiente paso de simulación), el sentido se elige aleatoriamente y el UAV viajará a velocidad máxima. Las soluciones mutadas deben ser corregidas ya que el UAV viajará hacia las mismas posiciones que antes, pero desde otro punto, procedimiento que se detalla en la sección de corrección.

Corrección de soluciones no factibles

Existen dos tipos de correcciones que pueden ser requeridas:

1. Un UAV tiene en su camino dos posiciones consecutivas que distan más de lo que puede recorrer en ese tiempo

2. Un UAV no retorna a la base

Para corregir las soluciones de tipo 1 se cambian las posiciones no alcanzables por la posición generada al dirigir al UAV a máxima velocidad en dirección a esa posición, desde la posición anterior. La corrección se realiza en orden de principio a fin.
Para el tipo 2 se determina cual es el paso de simulación más tardío en el que aún se puede volver a la base, y se cambia la trayectoria del UAV desde ese paso.

5.1.5. Criterio de parada

El criterio de parada establece qué condición debe cumplirse para dar por terminada la ejecución del algoritmo. Como los AE son técnicas no deterministas es de vital importancia establecer una condición de parada adecuada. Para el algoritmo con cruzamiento se utiliza la técnica de *Stagnation*, la cual establece que el algoritmo se detendrá una vez transcurrido un cierto número de generaciones sin mejora del mejor fitness. Para el caso del algoritmo MOSES, dado que las ejecuciones requieren

mucho tiempo computacional utilizando *Stagnation*, se utiliza esfuerzo prefijado. En la siguiente sección se presenta una elección informal para estos valores (número de generaciones sin mejora para *Stagnation* y número de generaciones para esfuerzo prefijado).

5.1.6. Diseño del algoritmo ávido

El algoritmo ávido envía cada UAV de la solución a seguir a el objetivo con su mismo ordinal. Si hay más UAV que objetivos, los sobrantes exploran el mapa de la misma manera que lo hace la población inicial de los algoritmos evolutivos.

5.2. Planificación en línea

En esta sección se describe la solución al problema de la vigilancia fuera de línea utilizando programación orientada a agentes para reaccionar ante cambios en el entorno o, más precisamente, en mantener la vigilancia de forma cooperativa al detectar un objetivo en un sistema multi agente.

5.2.1. Formulación

El problema consiste en dada una planificación fuera de línea, con todo lo que esto conlleva (*i.e.*, área, tiempo de vuelo, y las otras propiedades previamente definidas) implementar un algoritmo utilizando programación orientada a agentes que permita a los UAV reaccionar ante la aparición de objetivos mientras siguen la planificación fuera de línea.

Para formalizar un sistema orientado a agentes hay que definir las creencias, planes, acciones y metas. Por tratarse de un sistema multi agente, también se deben definir las interacciones presentes. Para la implementación se utilizó la metodología TROPOS que permite partir de una definición de requerimientos inicial y luego ir profundizando en estos requerimientos según las metas a cumplir. A continuación, se presentan los diagramas UML para el problema y la especificación de los componentes, los diagramas de requerimientos tempranos y tardíos en las Figuras 5.3 y 5.4, una visión global de los actores del sistemas y sus metas.

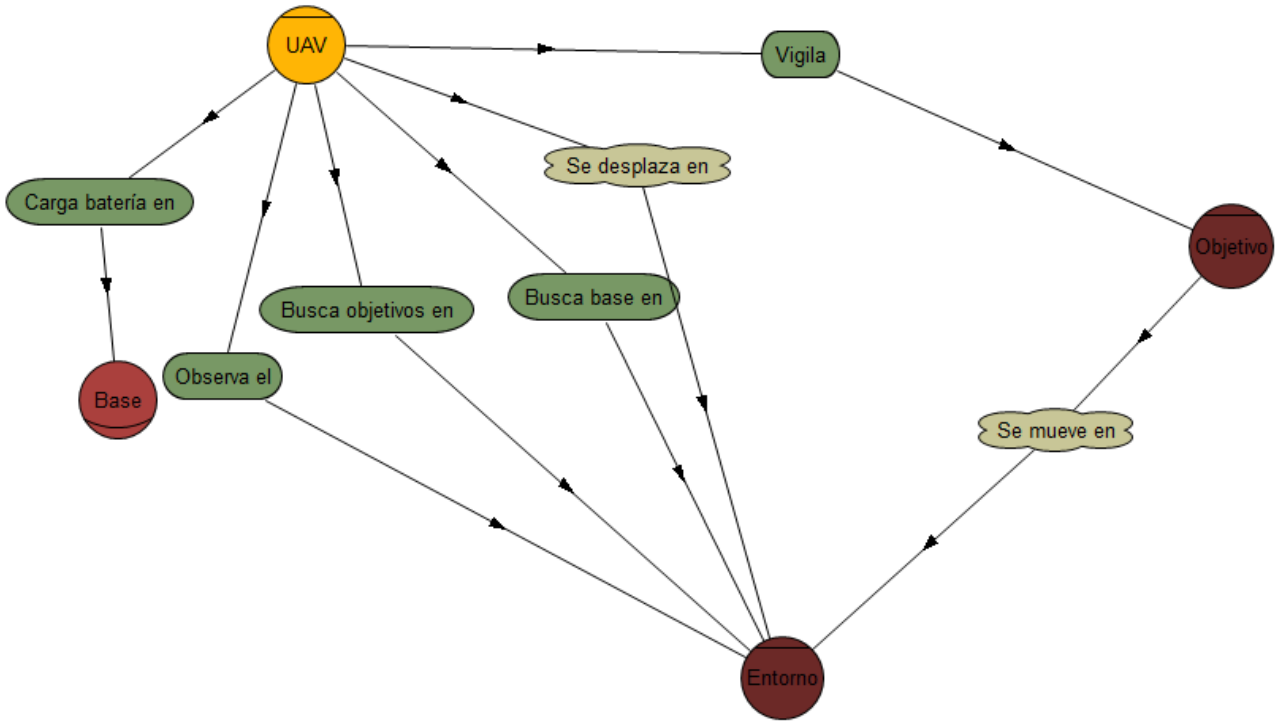


Figura 5.3: Diagrama de requerimientos tempranos

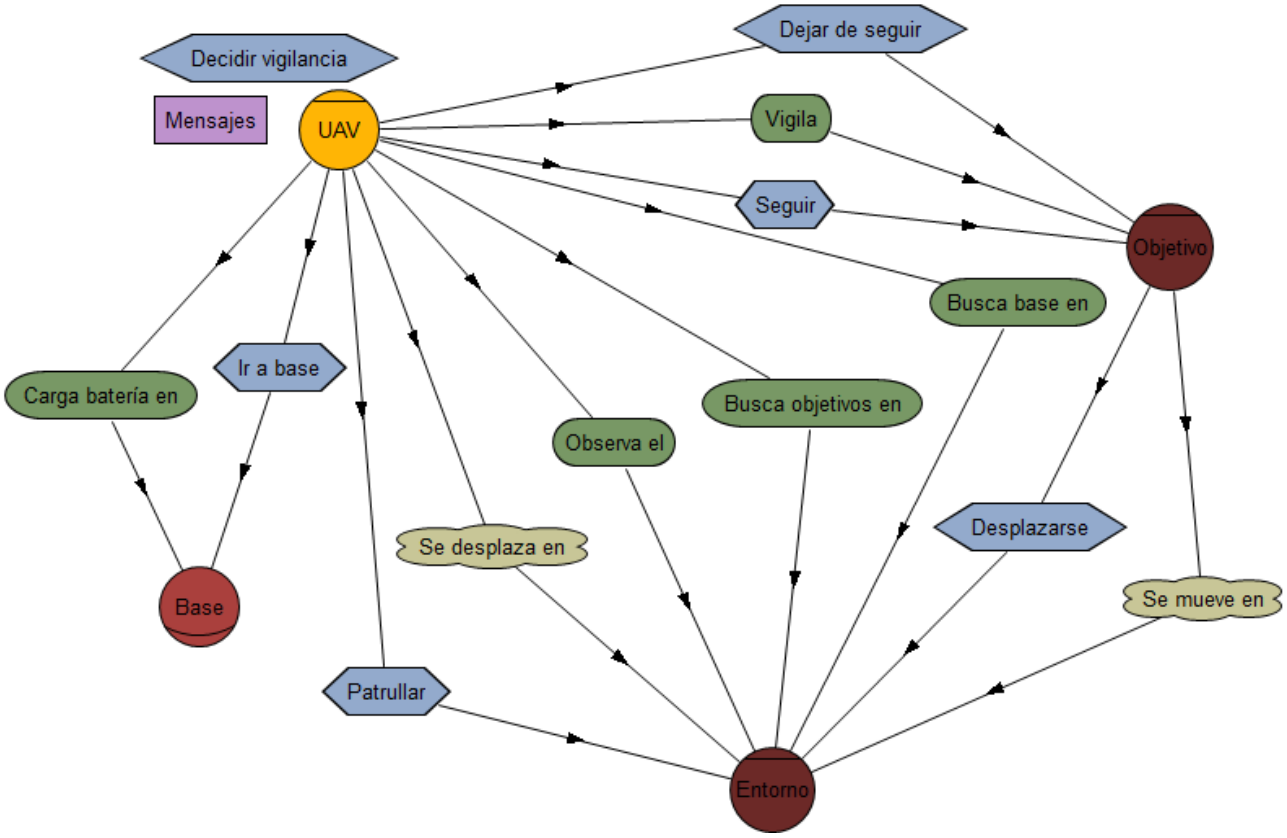


Figura 5.4: Diagrama de requerimientos tardíos

5.2.2. Agente UAV

Al ser definido como un agente BDI, se parte de las creencias que se definen en la Tabla 5.1. A partir de dichas creencias se definen metas con prioridades para cada deseo del UAV y las acciones que realiza, las mismas se pueden encontrar en la Tabla 5.2. Para lograr las metas se requiere de planes a seguir según el caso, para esto cada meta dispara un plan asociado. Cada plan puede disparar nuevas sub metas para lograr la meta principal, los detalles de estos planes se encuentran en la Tabla 5.3. Por otro lado cada UAV publica y provee servicios para el envío de mensajes personalizados. Estos contienen la información relevante para la coordinación. Los servicios se observan en la Tabla 5.4.

5.2.3. Agente Objetivo

El agente BDI que representa a los objetivos en la simulación. Tiene creencias básicas para el desplazamiento en el área, un subconjunto de las del agente UAV. Para la simulación se le atribuyen las creencias según se muestra en la Tabla 5.5. Al ser un agente que solo se desplaza en el área, tiene un conjunto mínimo de metas para su comportamiento; los detalles de las mismas se describen en la

Tabla 5.6. Los planes relacionados a las metas también son básicos para lograr simular un objetivo que se desplaza en el entorno, según se detalla en la Tabla 5.7. Este agente no intercambia mensajes específicos y por esto no define ningún servicio. En la implementación las capacidades de este agente son mayores al uso que se les dío, ya que puede llegar a tener un comportamiento similar al agente UAV. Por ejemplo un algoritmo para recorrer el área.

Tabla 5.1: Creencias agente UAV

Entorno	Esta creencia es necesaria solo para la simulación. Es una referencia al entorno virtual.
Objetivos	Referencia a objetivos dentro del rango de visión.
Bases	Referencia a bases visualizadas en el rango de visión.
UAVs	Referencia a UAV visualizados en el rango de visión.
Memoria	Posiciones visitadas en el área, utilizado para algoritmo de vuelo al terminar la patrulla.
Posición	Posición actual en el área.
Velocidad	Velocidad actual del UAV.
Visión	Rango de visión.
Conexión	Rango de red inalámbrica del UAV.
Batería	Carga actual de la batería.
Objetivo	Referencia al objetivo vigilado actualmente.
Patrulla	Posiciones del camino planificado fuera de línea.

5.2.4. Agente Entorno

Para la sincronización y comunicación básica del entorno con los demás agentes se le asoció un agente al entorno para el simulador. Este es el más simple de todos y sólo contiene el entorno virtual para la simulación. No tiene ninguna creencia, meta o plan, solo inicializar el entorno y la interfaz gráfica del simulador.

5.2.5. Diseño de la solución

Basado en la solución fuera de línea, se parte de esa planificación como base para los recorridos de vigilancia para detectar objetivos. Se toma como entrada una abstracción del ambiente para la toma de decisiones. Se utiliza en el envío de mensajes entre los UAV para la coordinación, la noción de distancias entre UAV y objetivos, vuelo a velocidad constante, consumo de batería directamente proporcional a la distancia recorrida y conexión entre UAV adyacentes. Este modelo de la realidad se

Tabla 5.2: Metas agente UAV

Mantener la batería cargada	De mayor prioridad para poder mantener el vuelo.
Buscar objetivos	Para reaccionar al visualizar un objetivo.
Realizar Patrulla	Control de recorrido de plan fuera de línea.
Seguir objetivo	Meta activada para la persecución.
Dejar de seguir objetivo	Meta para finalizar la persecución.
Ir a posición	Básica para moverse de una posición a otra.
Buscar base	Cuando se requiere recargar la batería se busca por una base.
Obtener visión	Actualiza la percepción del UAV con la información del entorno.
Memorizar posiciones	Mantiene una lista de las posiciones visitadas últimamente.

Tabla 5.3: Planes agente UAV

Recorrer lugar menos visitado recientemente	Plan para algoritmo de vuelo al terminar la patrulla.
Explorar mapa	Para cuando se requiere obtener información del entorno por parte del UAV.
Patrullar	Plan para recorrer camino planificado fuera de línea.
Seguir objetivo	Plan para seguir un objetivo.
Dejar de seguir	Plan para dejar de seguir un objetivo.
Cargar batería	Utilizado para cargar la batería.
Moverse a posición	Plan para desplazamiento de una posición a otra.
Memorizar	Para recordar las posiciones visitadas.
Ver	Plan para obtener percepción del entorno.

Tabla 5.4: Servicios agente UAV

Publicar	Envía mensaje a todos los UAV en el rango de conexión.
Suscribirse	Un UAV se suscribe para recibir mensajes al entrar en rango de conexión de otro UAV.

Tabla 5.5: Creencias agente objetivo

Entorno	Esta creencia es necesaria solo para la simulación. Es una referencia al entorno virtual.
Posición	Posición actual en el área.
Velocidad	Velocidad actual del objetivo.
Patrulla	Posiciones del camino a recorrer.

Tabla 5.6: Metas agente objetivo

Realizar Patrulla	Control de recorrido.
Ir a posición	Básica para moverse de una posición a otra.
Obtener visión	Actualiza la percepción del objetivo con la información del entorno.

Tabla 5.7: Planes agente objetivo

Recorrer	Plan para recorrer camino predefinido.
Moverse a posición	Plan para desplazamiento de una posición a otra.
Ver	Plan para obtener percepción del entorno.

implementó utilizando programación orientada a agentes que interactúan entre sí y con el entorno. De varias implementaciones de frameworks de programación orientada a agentes se eligió Jadex, por ser completamente realizado en Java, orientar los agentes a servicios, con la posibilidad de ser implantado en un caso real, además provee herramientas para control y simulación.

Jadex

En nuestro diseño se optó por ésta implementación del framework AOP de agentes BDI porque utiliza herramientas de Java como las anotaciones para la definición de agentes. Esto permite que toda la lógica que utiliza un agente se pueda definir en código Java sin el uso de lenguajes auxiliares para reglas lógicas como utilizan otras implementaciones del framework AOP, por ejemplo Jason. Basados en estos componentes de Jadex cada UAV es representado como un agente, esto en Jadex se define como una clase Java con anotaciones que representan el modelo BDI. De igual forma se definen los planes, metas y acciones. La posibilidad de definir estos componentes en clases separadas le da claridad al código. Por otro lado para el simulador se genera una interfaz gráfica que se acopla con las definiciones de los agentes simulando el entorno y las percepciones que tienen los UAV. Al ser Jadex una plataforma multi agente, solo se debe definir un agente UAV para que luego en el momento de ejecución, cada uno se configure de forma distinta según el camino asignado en la planificación fuera de línea. Cada objetivo en el simulador también es representado como un agente que solo se comunica con el entorno. El entorno tiene asociado un agente en la simulacion para poder interactuar con los demas agentes y es el encargado de simular el medio de comunicación de los UAV y las interacciones con los objetivos.

5.3. Implementación Vuelo Fuera de Línea

En esta sección se describe cómo se implementó el vuelo de los UAV utilizando la planificación fuera de línea.

5.3.1. Ambiente utilizado

Se utilizaron UAV Bebop 2 de Parrot, estos tienen una distribución linux para el manejo del UAV. Esto permitió compilar e instalar Python 3.7 y de esta forma utilizar la biblioteca pyparrot. Esta biblioteca permite el control de Bebop implementado el protocolo de comunicación de Parrot. Parrot brinda un simulador llamado Sphinx que permite emular completamente un Bebop 2 y su firmware. Utiliza Gazebo para la simulación de la interacción del Bebop con el ambiente y la interfaz gráfica. De esta forma se puede simular la ejecución de scripts Python utilizando pyparrot, la diferencia con la ejecución real es que los scripts correrán en el Bebop mismo en vez de utilizar una conexión de red.

5.3.2. Arquitectura de la implementación

Al utilizar la planificación fuera de línea, que produce una serie de coordenadas para cada UAV, el problema se reduce a seguir estas coordenadas para cada UAV. Ya que los Bebop constan de comunicación por red inalámbrica, y cada uno tiene un punto de acceso, es posible conectarse a cada uno de ellos teniendo una interfaz para cada uno. Aprovechando la ventaja de poder establecer una conexión se implementa un servidor http que escucha para recibir el camino que debe recorrer y provee servicios de información. Con esta implementación cada Bebop puede volar de forma independiente, no requiere comunicación constante a una base. Esta arquitectura de clientes-servidores permite que a futuro se pueda extender la comunicación entre los UAV para intercambiar información por ejemplo. Una opción podría ser que se interconectaran en una misma red dejando uno como punto de acceso por grupo de Bebop. De esa forma todos podrían en esa red enviar y recibir información.

5.3.3. Descripción de la implementación

Python permite crear de manera simple un servidor HTTP, en este se manejaron los mensajes POST para los comandos al Bebop y los métodos GET para obtener el estado. Utilizando pyparrot para el control del Bebop, y Coordinates para el manejo de coordenadas, se logra que cada Bebop recorra un camino. Los mensajes POST contienen en el cuerpo del mensaje los parametros para el vuelo o aterrizaje del Bebop en formato JSON. Los mensajes GET devuelven de forma cruda información del vuelo y el log del servidor. Para los métodos POST se agrego seguridad mediante un token para reducir el acceso a dar un comando a un Bebop. En cada Bebop el servidor debería ser levantado al inicio con su configuración correspondiente. En la configuración del servidor del Bebop deben estar valores básicos como el puerto donde escucha el servicios, el nombre que identifica al Bebop, el token esperado, el nivel de log y ruta de archivo de log. Para cada mensaje POST se crea un proceso para manejarlo, esto permite la concurrencia de pedidos al servidor. Además del servidor se creó una clase que maneja todo el control del Bebop según la planificación fuera de línea y la configuración de la misma.

5.3.4. Compilación de Python 3.7

Para esto se utilizó un Raspberry Pi 3 que consta de la misma arquitectura. Los pasos fueron los siguientes: apt install libffi-dev libbz2-dev liblzma-dev libsqlite3-dev libncurses5-dev libgdbm-dev zlib1g-dev libreadline-dev libssl-dev tk-dev build-essential libncursesw5-dev libc6-dev openssl git wget https://github.com/python/cpython/archive/v3.7.3.tar.gz tar -xvf v3.7.3.tar.gz cd cpython-3.7.3 ./configure –prefix=$HOME/.local –enable-optimizations make -j -l 4

5.3.5. Resultados

Se logro volar dos UAV Bebop 2 que siguieron exitosamente una misión sencilla.

Capítulo 6

Análisis experimental

Los algoritmos evolutivos conllevan un grado significativo de aleatoriedad como parte de su lógica por lo que pueden producir resultados diferentes en cada ejecución cuando se aplica a la misma instancia del problema. Por lo tanto, es importante evaluar la efectividad de los mismos mediante la recopilación de datos de un número suficientemente grande de ejecuciones. El uso riguroso de pruebas estadísticas es esencial para brindar apoyo a las conclusiones derivadas del análisis de dichos datos.

Muchas técnicas son evaluadas experimentando con varios casos de prueba. A menudo es difícil generalizar los resultados experimentales para hacer una afirmación sobre una técnica en particular. Sin embargo, es posible demostrar la utilidad de un nuevo método en varios casos cuidadosamente seleccionados, comparando el método con otras técnicas.

En el capítulo anterior se detallaron las técnicas utilizadas (algoritmo Greedy, algoritmo evolutivo y algoritmo MOSES) y en este capítulo se presentan las instancias del problema (o sea los casos de prueba) utilizados, los resultados obtenidos y comparaciones de los mismos.

6.1. Plataforma de ejecución y desarrollo

Por razones de reproducibilidad y evaluación de la eficiencia computacional, a continuación se detallan las características de la plataforma de ejecución y desarrollo de los algoritmos.

- Procesador: Intel Core i5
- Velocidad del procesador: 3 GHz
- Número de cores del procesador: 6
- Cache L2: 256 KB
- Cache L3: 9 MB

- RAM: 16 GB
- Disco duro de estado solido (SSD)
- Versión de Java: openjdk 12.0.1

6.2. Instancias del problema

Las configuraciones paramétricas de los algoritmos evolutivos se evalúan sobre una instancia de configuración (diferente al conjunto de problemas de validación), para evitar sesgos en la parametrización. A su vez la instancia es de menor tamaño para reducir la complejidad computacional, y así el tiempo requerido para el ajuste de parámetros. Los dos algoritmos evolutivos propuestos y el algoritmo Greedy son evaluados utilizando 3 instancias del problema. Las características de cada instancia de prueba, numeradas de 1 a 3, se muestran en la Tabla 6.1. En la misma tabla se encuentra la instancia de configuración paramétrica, con el numeral 0.

Tabla 6.1: Instancias de prueba

#	H *(m)*	W *(m)*	base	u	v *(m/s)*	t *(s)*	s	o	vo *(m/s)*	P	r_o	r_c
0	100	100	(50, 50)	5	2	100	10	2	1	[2, 4]	3	2
1	1000	1000	(300, 300)	5	10	1000	10	4	5	[1, 2, 3, 4]	5	5
2	1000	1000	(700, 700)	10	10	1000	10	4	5	[2, 2, 8, 8]	5	10
3	10000	10000	(5000, 5000)	5	10	2000	20	5	0.1	[1, 1, 1, 1, 10]	2	1000

Para la generación de los caminos de los objetivos se utilizó la técnica RRT. Como entradas de dicho algoritmo se utilizaron posiciones que fueron construidas a partir de la generación aleatoria de sus componentes (x e y). Las Figuras 6.1, 6.2, 6.3, 6.4 muestran en forma gráfica las instancias del problema, en particular se puede observar el tamaño del área, la posición inicial de los drones (la base) y el recorrido de los objetivos. En la Figura 6.4 no se ven los recorridos porque son cortos en relación al tamaño del área.

6.3. Planificación fuera de línea

Esta sección presenta los resultados de la configuración paramétrica de los dos algoritmos evolutivos, seguido de una sección específica de evaluación para cada algoritmo. Finalmente se realizan comparativas entre ellos.

6.3.1. Configuración paramétrica

Para la etapa de ajuste de parámetros se utiliza una instancia del problema pequeña, diferente a las utilizadas para la evaluación experimental de forma de evitar sesgos. Las características de esta

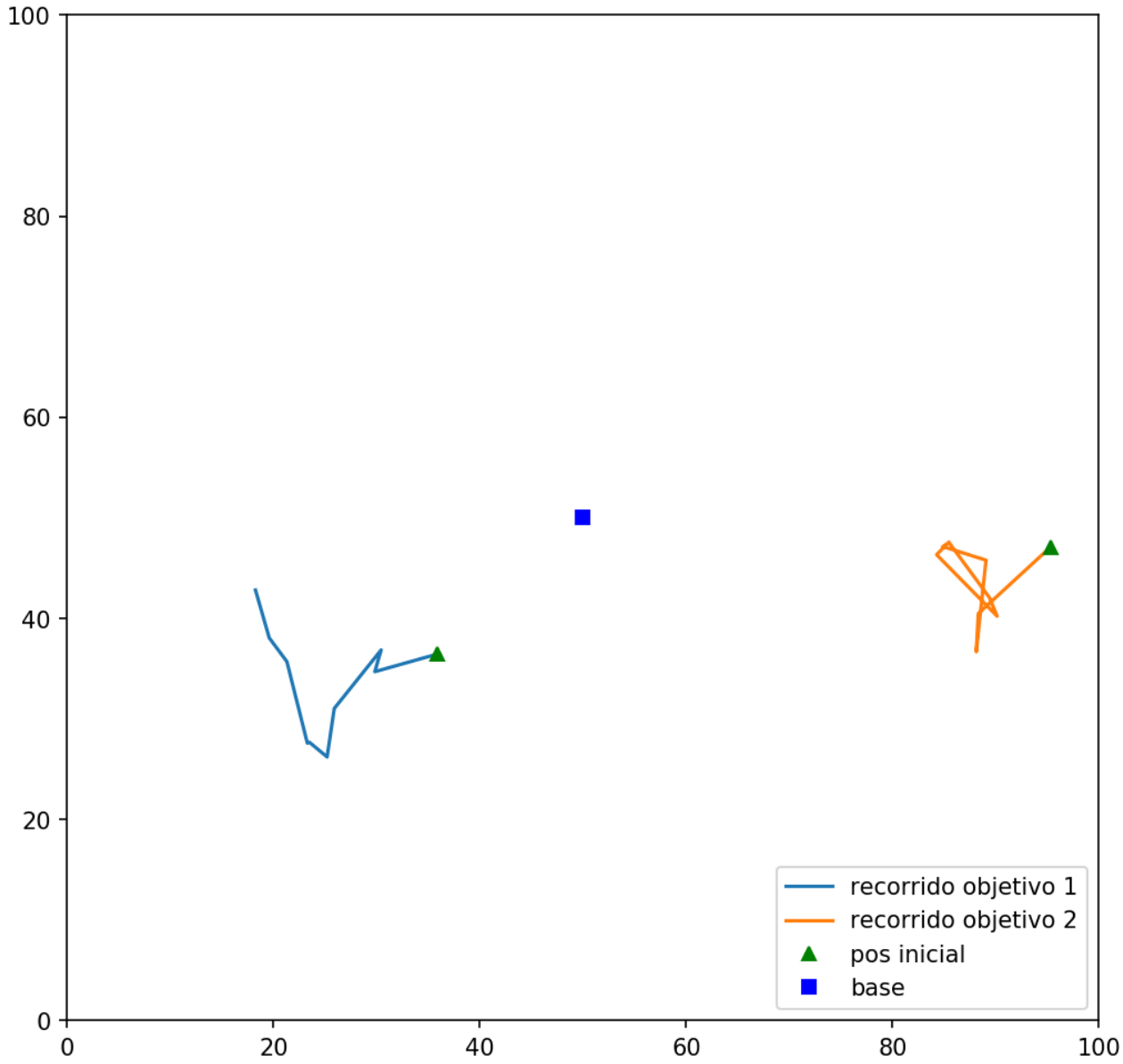


Figura 6.1: Problema 0

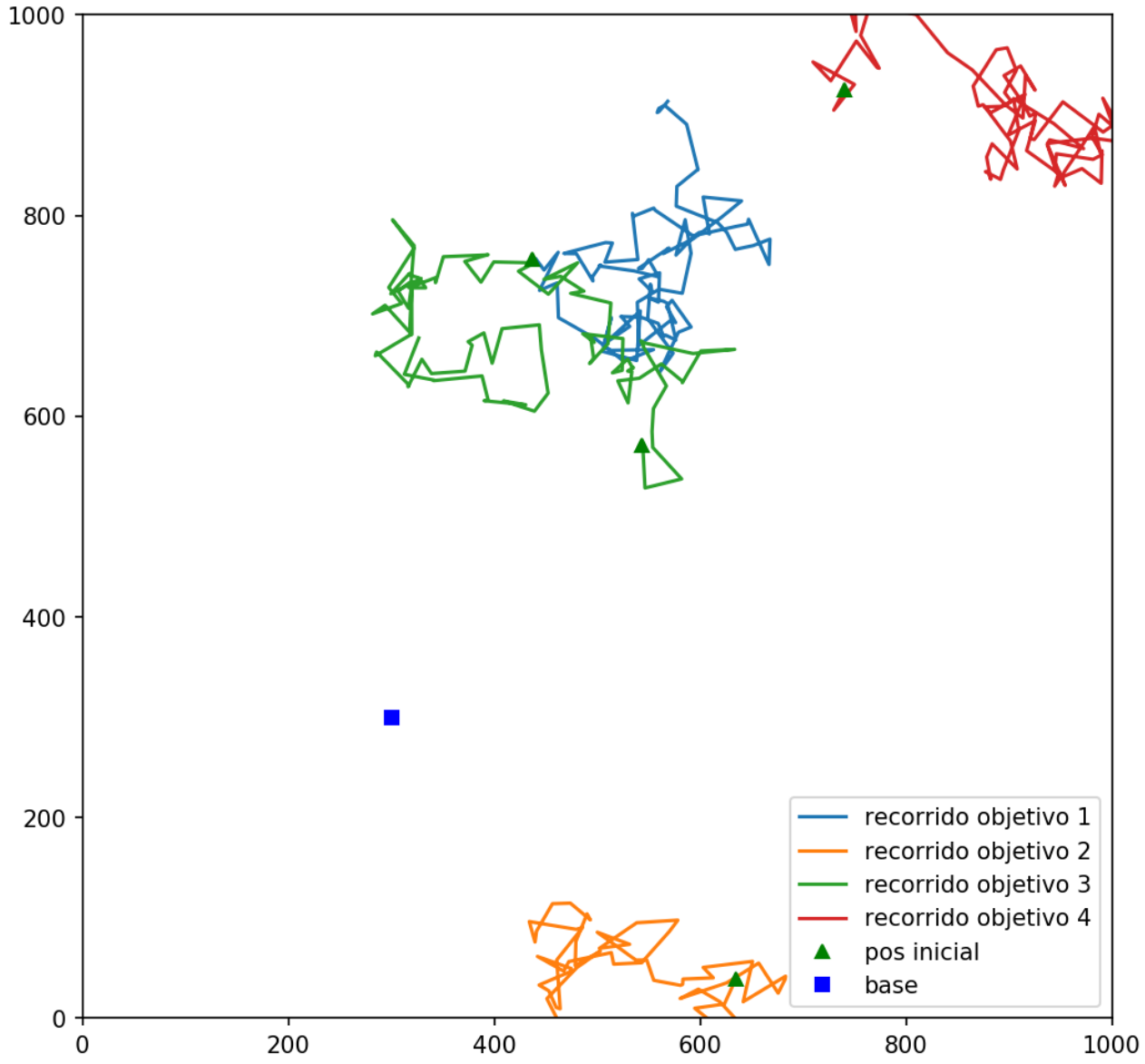


Figura 6.2: Problema 1

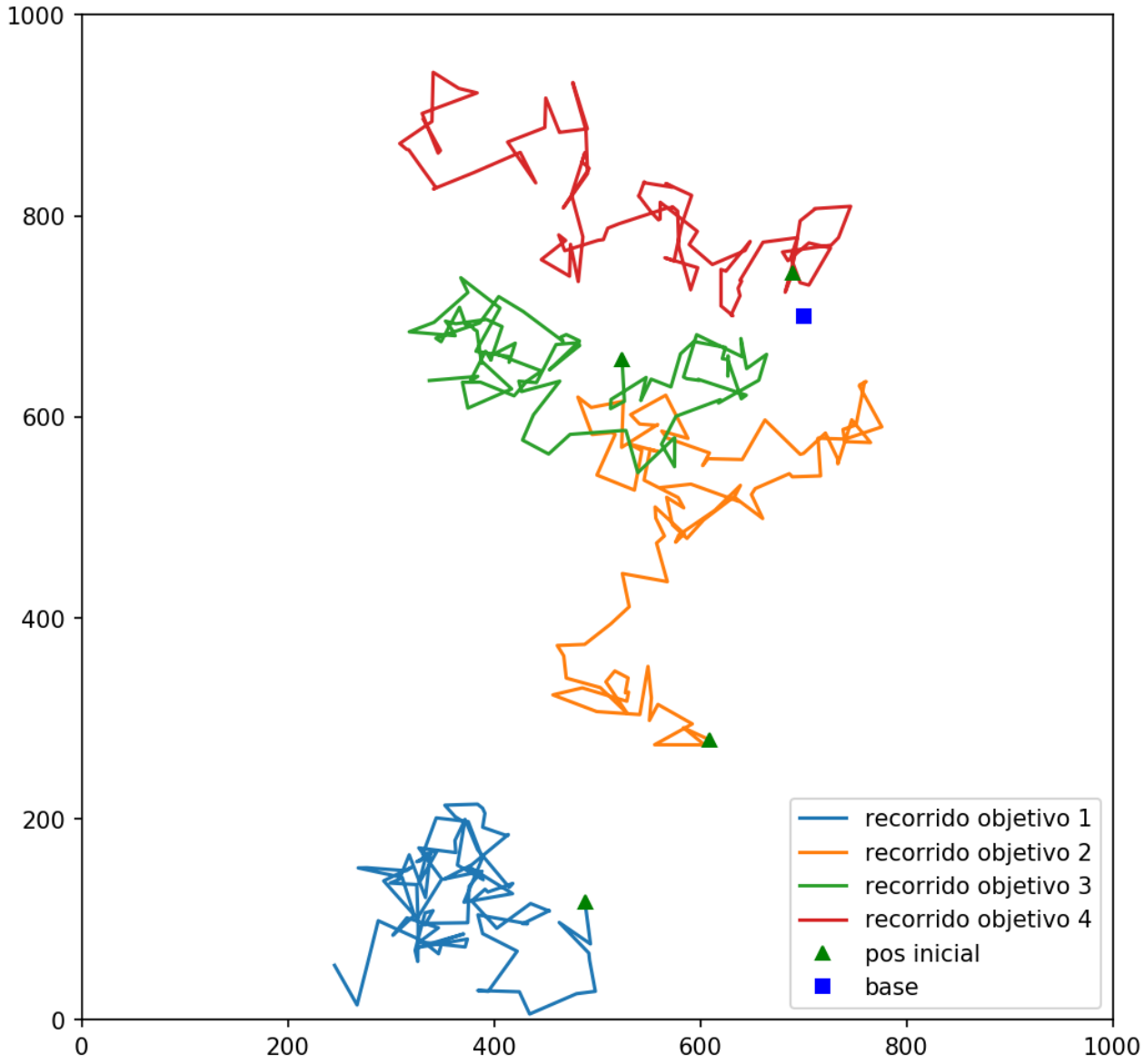


Figura 6.3: Problema 2

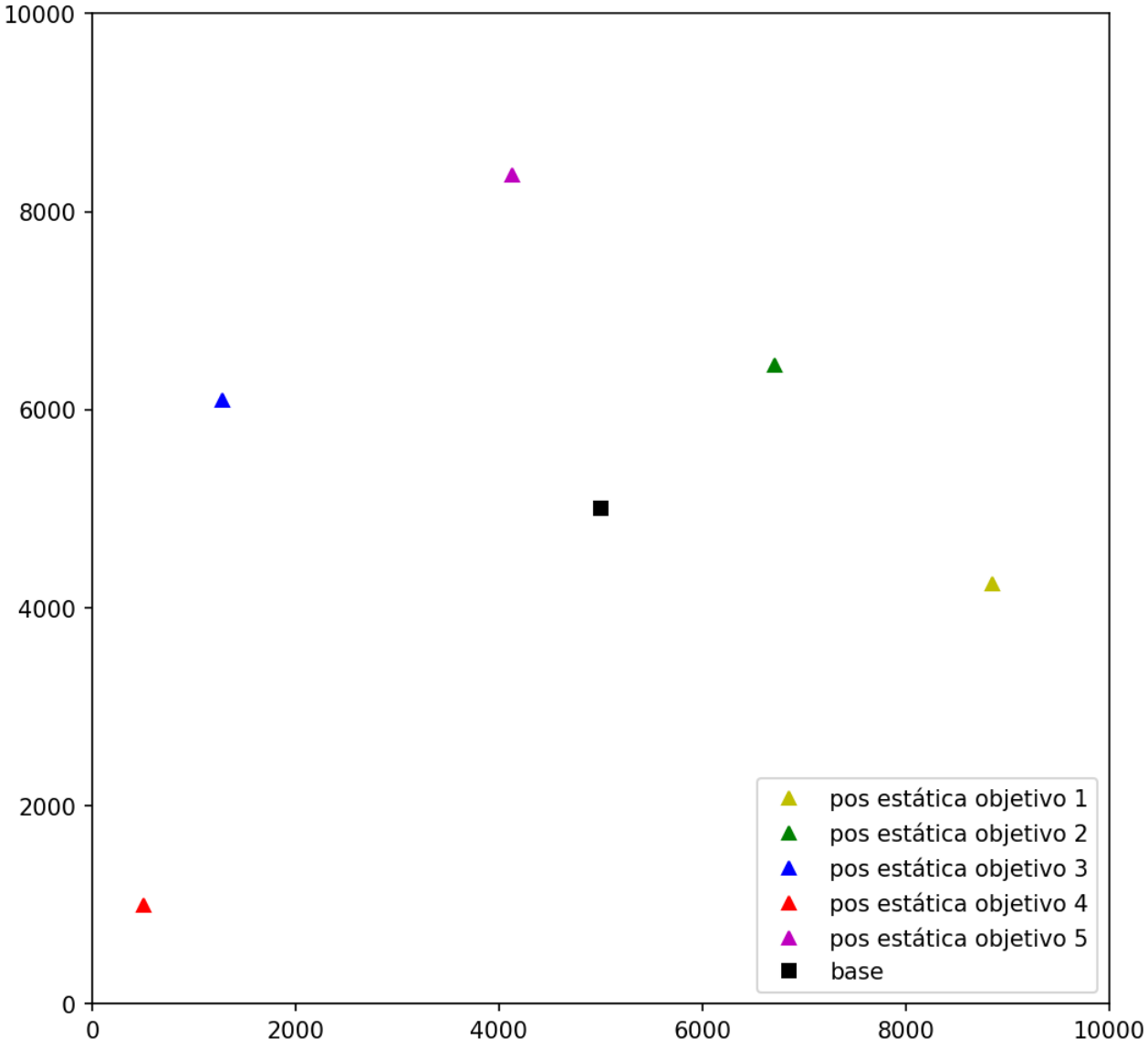


Figura 6.4: Problema 3

instancia se pueden ver en la primera fila de la Tabla 6.1 y en la Figura 6.1. Mediante experimentos preliminares se fijó el tamaño de la población en 200 individuos y el criterio de parada en 2000 generaciones sin mejorar el máximo valor de fitness de la población. Se realizó la etapa de configuración de los restantes parámetros utilizando la instancia de prueba mencionada anteriormente. Los parámetros estudiados fueron: p_C y p_M para EA; T y P_{min} para MOSES. Los valores candidatos para cada parámetro fueron los siguientes: $p_C \in \{0.5, 0.8, 1.0\}$; $p_M \in \{0.001, 0.1, 0.15\}$; $T \in \{1, 5, 100\}$ y $P_{min} \in \{0.1, 0.2, 0.3\}$.

Para cada combinación posible de valores de los parámetros estudiados se realizaron 40 ejecuciones independientes.

AE

Se realiza el test estadístico de Kolmogorov-Smirnov de una muestra para comparar los resultados con una distribución normal general con media y desviación estándar igual a la de los resultados, y por lo tanto determinar si los resultados siguen una distribución normal. Como se puede ver en los resultados de la Tabla 6.2, el p-valor obtenido no es menor a 0.05, por lo que no se rechaza la hipótesis nula de que los resultados fueron extraídos de una distribución normal. Tomando en cuenta este resultado, se asume que las distribuciones son aproximadamente normales, y se realiza un análisis de varianza (ANOVA) para determinar que existen diferencias. El resultado del test es un estadístico de 155.17 y un p-valor de 2.46×10^{-110}, por lo que se realiza el test de Tukey para identificar cuales son estadísticamente diferentes y cual es la mejor, los resultados se pueden ver el la Tabla 6.3. Como se puede ver en la Figura 6.5 la configuración $(p_M, p_C) = (0.15, 1.0)$ es la configuración que alcanza una media más alta, y por lo tanto, la configuración utilizada. Si bien no hay una diferencia estadística con la distribución de la configuración $(p_M, p_C) = (0.15, 0.8)$, la diferencia de tiempo de ejecución medio es marginal (0.59 segundos). En la Tabla 6.4 se presentan métricas relevantes para evaluar la calidad de la solución, y en la Figura 6.6 un análisis gráfico.

Tabla 6.2: Kolmogorov-Smirnov - AE

parámetros		kstest	
p_M	p_C	estadístico	p-valor
0.01	0.5	0.084	0.94
0.01	0.8	0.078	0.97
0.01	1.0	0.089	0.91
0.10	0.5	0.10	0.81
0.10	0.8	0.14	0.39
0.10	1.0	0.11	0.73
0.15	0.5	0.101	0.80
0.15	0.8	0.09	0.93
0.15	1.0	0.11	0.76

Tabla 6.3: Prueba de Turkey - AE

groupo 1	groupo 2	diferencia	p-adj	inferior	superior	rechaza
(0.01, 0.5)	(0.01, 0.8)	24.3542	0.9000	-42.2488	90.9572	No
(0.01, 0.5)	(0.01, 1.0)	54.8667	0.2026	-11.7363	121.4697	No
(0.01, 0.5)	(0.10, 0.5)	245.5500	0.0010	178.947	312.1530	Si
(0.01, 0.5)	(0.10, 0.8)	306.8583	0.0010	240.2553	373.4613	Si
(0.01, 0.5)	(0.10, 1.0)	370.6583	0.0010	304.0553	437.2613	Si
(0.01, 0.5)	(0.15, 0.5)	295.9958	0.0010	229.3928	362.5988	Si
(0.01, 0.5)	(0.15, 0.8)	451.9000	0.0010	385.2970	518.5030	Si
(0.01, 0.5)	(0.15, 1.0)	515.4375	0.0010	448.8345	582.0405	Si
(0.01, 0.8)	(0.01, 1.0)	30.5125	0.8814	-36.0905	97.1155	No
(0.01, 0.8)	(0.10, 0.5)	221.1958	0.0010	154.5928	287.7988	Si
(0.01, 0.8)	(0.10, 0.8)	282.5042	0.0010	215.9012	349.1072	Si
(0.01, 0.8)	(0.10, 1.0)	346.3042	0.0010	279.7012	412.9072	Si
(0.01, 0.8)	(0.15, 0.5)	271.6417	0.0010	205.0387	338.2447	Si
(0.01, 0.8)	(0.15, 0.8)	427.5458	0.0010	360.9428	494.1488	Si
(0.01, 0.8)	(0.15, 1.0)	491.0833	0.0010	424.4803	557.6863	Si
(0.01, 1.0)	(0.10, 0.5)	190.6833	0.0010	124.0803	257.2863	Si
(0.01, 1.0)	(0.10, 0.8)	251.9917	0.0010	185.3887	318.5947	Si
(0.01, 1.0)	(0.10, 1.0)	315.7917	0.0010	249.1887	382.3947	Si
(0.01, 1.0)	(0.15, 0.5)	241.1292	0.0010	174.5262	307.7322	Si
(0.01, 1.0)	(0.15, 0.8)	397.0333	0.0010	330.4303	463.6363	Si
(0.01, 1.0)	(0.15, 1.0)	460.5708	0.0010	393.9678	527.1738	Si
(0.10, 0.5)	(0.10, 0.8)	61.3083	0.0988	-5.2947	127.9113	No
(0.10, 0.5)	(0.10, 1.0)	125.1083	0.0010	58.5053	191.7113	Si
(0.10, 0.5)	(0.15, 0.5)	50.4458	0.3074	-16.1572	117.0488	No
(0.10, 0.5)	(0.15, 0.8)	206.3500	0.0010	139.747	272.953	Si
(0.10, 0.5)	(0.15, 1.0)	269.8875	0.0010	203.2845	336.4905	Si
(0.10, 0.8)	(0.10, 1.0)	63.8000	0.0728	-2.8030	130.403	No
(0.10, 0.8)	(0.15, 0.5)	-10.8625	0.9000	-77.4655	55.7405	No
(0.10, 0.8)	(0.15, 0.8)	145.0417	0.0010	78.4387	211.6447	Si
(0.10, 0.8)	(0.15, 1.0)	208.5792	0.0010	141.9762	275.1822	Si
(0.10, 1.0)	(0.15, 0.5)	-74.6625	0.0153	-141.2655	-8.0595	Si
(0.10, 1.0)	(0.15, 0.8)	81.2417	0.0052	14.6387	147.8447	Si
(0.10, 1.0)	(0.15, 1.0)	144.7792	0.0010	78.1762	211.3822	Si
(0.15, 0.5)	(0.15, 0.8)	155.9042	0.0010	89.3012	222.5072	Si
(0.15, 0.5)	(0.15, 1.0)	219.4417	0.0010	152.8387	286.0447	Si
(0.15, 0.8)	(0.15, 1.0)	63.5375	0.0753	-3.0655	130.1405	No

Figura 6.5: Medias por configuración: Fitness y Tiempo - AE

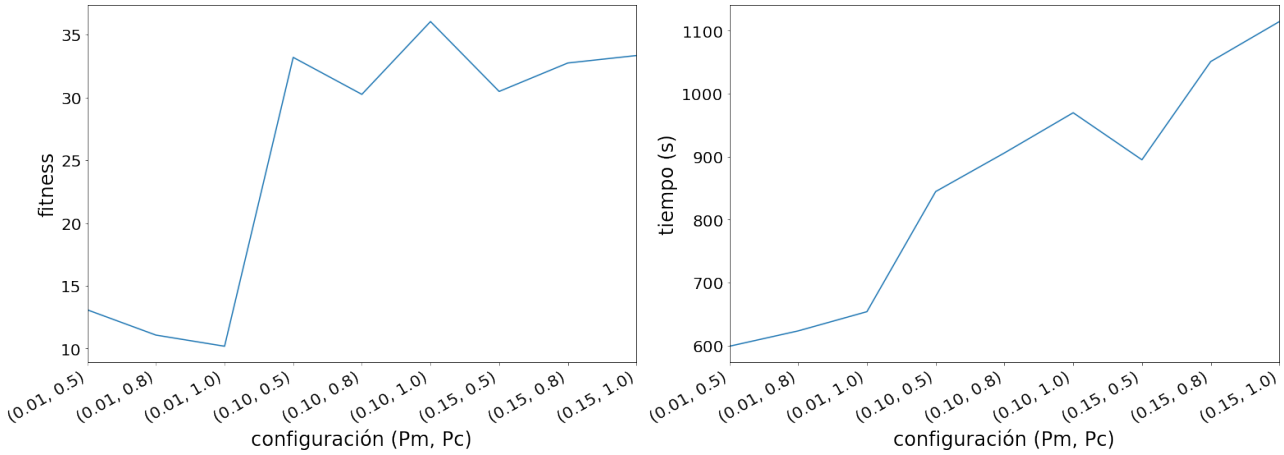
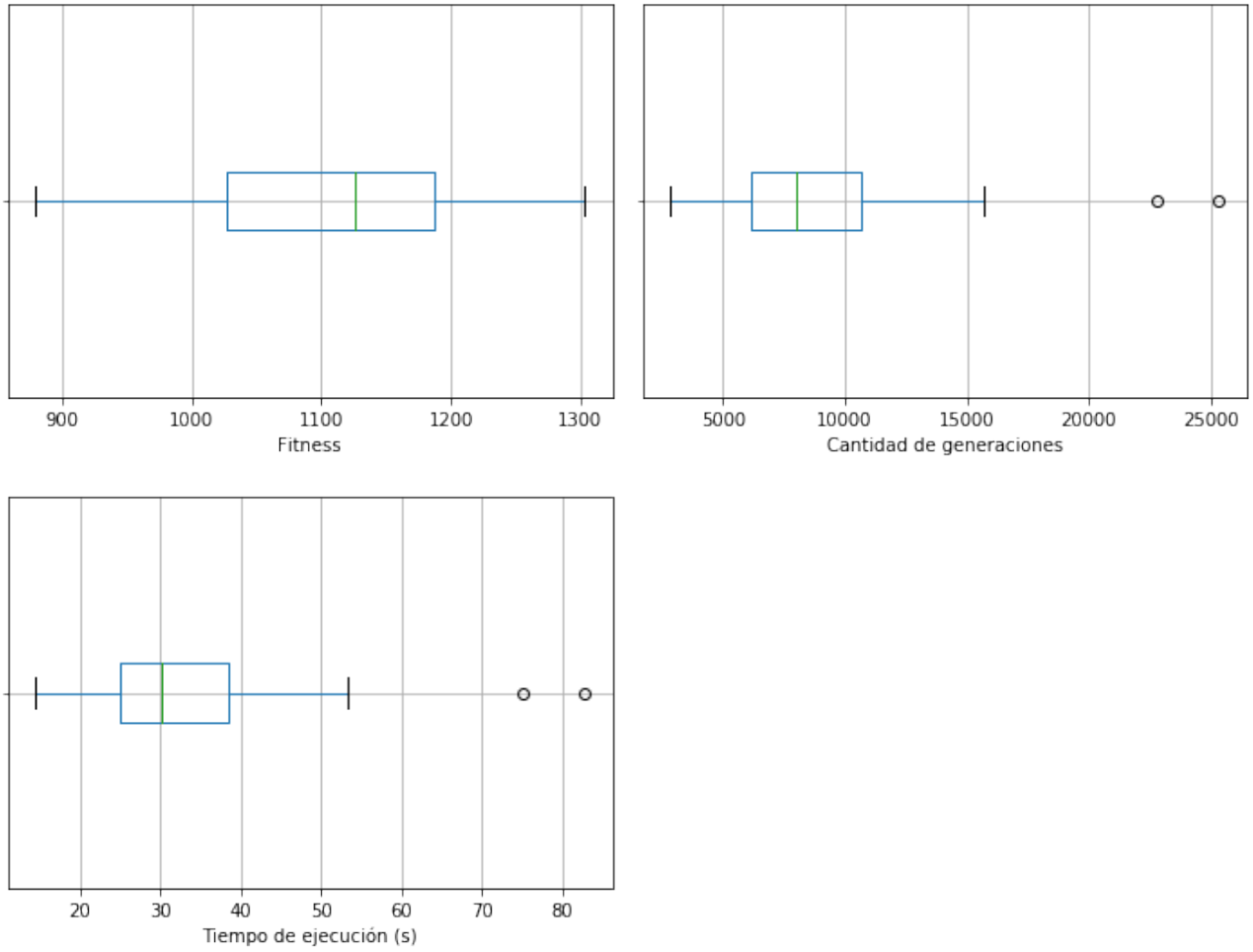


Tabla 6.4: Métricas - Problema 0, configuración seleccionada para el AE

	Fitness	Generaciones	Tiempo (s)
max	1303.50	25331	82.77
mean	1114.47	9020.05	33.34
std	101.95	4778.78	14.46

Figura 6.6: Boxplots de la configuración seleccionada para el AE



MOSES

Al igual que con el AE, se realiza el test estadístico de Kolmogorov-Smirnov de una muestra para comparar los resultados con una distribución normal general con media y desviación estándar igual a la de los resultados, y por lo tanto determinar si los resultados siguen una distribución normal. Como se puede ver en los resultados de la Tabla 6.5, el p-valor obtenido no es menor a 0.05, por lo que no se rechaza la hipótesis nula de que los resultados fueron extraídos de una distribución normal. Tomando en cuenta este resultado, se asume que las distribuciones son aproximadamente normales, y se realiza un análisis de varianza (ANOVA) para determinar que existen diferencias. El resultado del test es un estadístico de 128.43 y un p-valor de 2.13×10^{-99}, por lo que se realiza el test de Tukey para identificar cuales son estadísticamente diferentes y cual es la mejor, los resultados se pueden ver el la Tabla 6.6. Como se puede ver en la Figura 6.7 las configuraciones con $T = 1$ son las configuraciones que alcanzan una media más alta, y según la prueba de Turkey, no hay una diferencia estadística entre las mismas. Dado que la configuración con $P_{min} = 0.1$ tiene un tiempo de ejecución medio menor, es la configuración utilizada. En la Tabla 6.7 se presentan métricas relevantes para evaluar la calidad de la solución, y en la Figura 6.8 un análisis gráfico.

Tabla 6.5: Kolmogorov-Smirnov - MOSES

parámetros		kstest	
T	P_{min}	estadístico	p-valor
1	0.1	0.08	0.97
1	0.2	0.09	0.90
1	0.3	0.10	0.82
5	0.1	0.14	0.35
5	0.2	0.12	0.60
5	0.3	0.12	0.56
100	0.1	0.12	0.58
100	0.2	0.09	0.91
100	0.3	0.11	0.67

6.3.2. Comparación de resultados

Como en la configuración paramétrica, para la etapa de análisis experimental y comparación de resultados se realizaron 40 ejecuciones independientes de los algoritmos evolutivos propuestos sobre cada instancia del problema. Adicionalmente, se realizó una ejecución del algoritmo Greedy sobre cada instancia de prueba. Ya que el algoritmo es determinista no es necesario ejecutarlo más de una vez.

Para cada instancia del problema se realiza el test estadístico Kolmogorov-Smirnov de una muestra para comparar los resultados con una distribución normal general con media y desviación están-

Tabla 6.6: Prueba de Turkey - MOSES

grupo 1	groupo 2	diferencia	p-adj	inferior	superior	rechaza
(1, 0.1)	(1, 0.2)	-4.8167	0.9000	-24.5469	14.9136	No
(1, 0.1)	(1, 0.3)	3.2417	0.9000	-16.4886	22.9719	No
(1, 0.1)	(100, 0.1)	-51.1958	0.0010	-70.9261	-31.4656	Si
(1, 0.1)	(100, 0.2)	-37.7708	0.0010	-57.5011	-18.0406	Si
(1, 0.1)	(100, 0.3)	-66.9667	0.0010	-86.6969	-47.2364	Si
(1, 0.1)	(5, 0.1)	-87.1625	0.0010	-106.8928	-67.4322	Si
(1, 0.1)	(5, 0.2)	-38.4167	0.0010	-58.1469	-18.6864	Si
(1, 0.1)	(5, 0.3)	-155.8792	0.0010	-175.6094	-136.1489	Si
(1, 0.2)	(1, 0.3)	8.0583	0.9000	-11.6719	27.7886	No
(1, 0.2)	(100, 0.1)	-46.3792	0.0010	-66.1094	-26.6489	Si
(1, 0.2)	(100, 0.2)	-32.9542	0.0010	-52.6844	-13.2239	Si
(1, 0.2)	(100, 0.3)	-62.1500	0.0010	-81.8803	-42.4197	Si
(1, 0.2)	(5, 0.1)	-82.3458	0.0010	-102.0761	-62.6156	Si
(1, 0.2)	(5, 0.2)	-33.6000	0.0010	-53.3303	-13.8697	Si
(1, 0.2)	(5, 0.3)	-151.0625	0.0010	-170.7928	-131.3322	Si
(1, 0.3)	(100, 0.1)	-54.4375	0.0010	-74.1678	-34.7072	Si
(1, 0.3)	(100, 0.2)	-41.0125	0.0010	-60.7428	-21.2822	Si
(1, 0.3)	(100, 0.3)	-70.2083	0.0010	-89.9386	-50.4781	Si
(1, 0.3)	(5, 0.1)	-90.4042	0.0010	-110.1344	-70.6739	Si
(1, 0.3)	(5, 0.2)	-41.6583	0.0010	-61.3886	-21.9281	Si
(1, 0.3)	(5, 0.3)	-159.1208	0.0010	-178.8511	-139.3906	Si
100, 0.1)	(100, 0.2)	13.4250	0.4612	-6.3053	33.1553	No
100, 0.1)	(100, 0.3)	-15.7708	0.2378	-35.5011	3.9594	No
100, 0.1)	(5, 0.1)	-35.9667	0.0010	-55.6969	-16.2364	Si
100, 0.1)	(5, 0.2)	12.7792	0.5250	-6.9511	32.5094	No
100, 0.1)	(5, 0.3)	-104.6833	0.0010	-124.4136	-84.9531	Si
100, 0.2)	(100, 0.3)	-29.1958	0.0010	-48.9261	-9.4656	Si
100, 0.2)	(5, 0.1)	-49.3917	0.0010	-69.1219	-29.6614	Si
100, 0.2)	(5, 0.2)	-0.6458	0.9000	-20.3761	19.0844	No
100, 0.2)	(5, 0.3)	-118.1083	0.0010	-137.8386	-98.3781	Si
100, 0.3)	(5, 0.1)	-20.1958	0.04030	-39.9261	-0.4656	Si
100, 0.3)	(5, 0.2)	28.5500	0.0010	8.8197	48.2803	Si
100, 0.3)	(5, 0.3)	-88.9125	0.0010	-108.6428	-69.1822	Si
(5, 0.1)	(5, 0.2)	48.7458	0.0010	29.0156	68.4761	Si
(5, 0.1)	(5, 0.3)	-68.7167	0.0010	-88.4469	-48.9864	Si
(5, 0.2)	(5, 0.3)	-117.4625	0.0010	-137.1928	-97.7322	Si

Figura 6.7: Medias por configuración: Fitness y Tiempo - MOSES

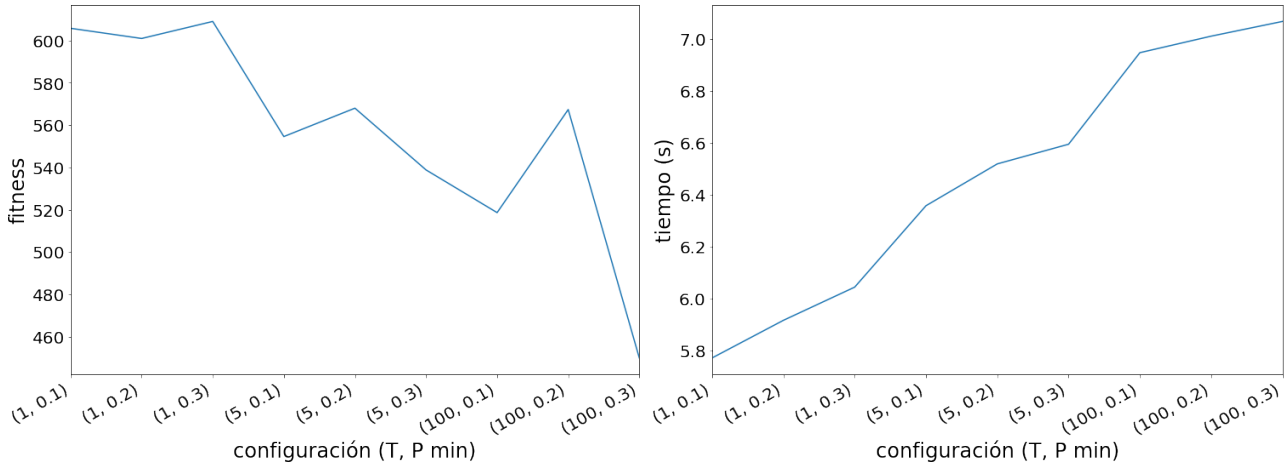
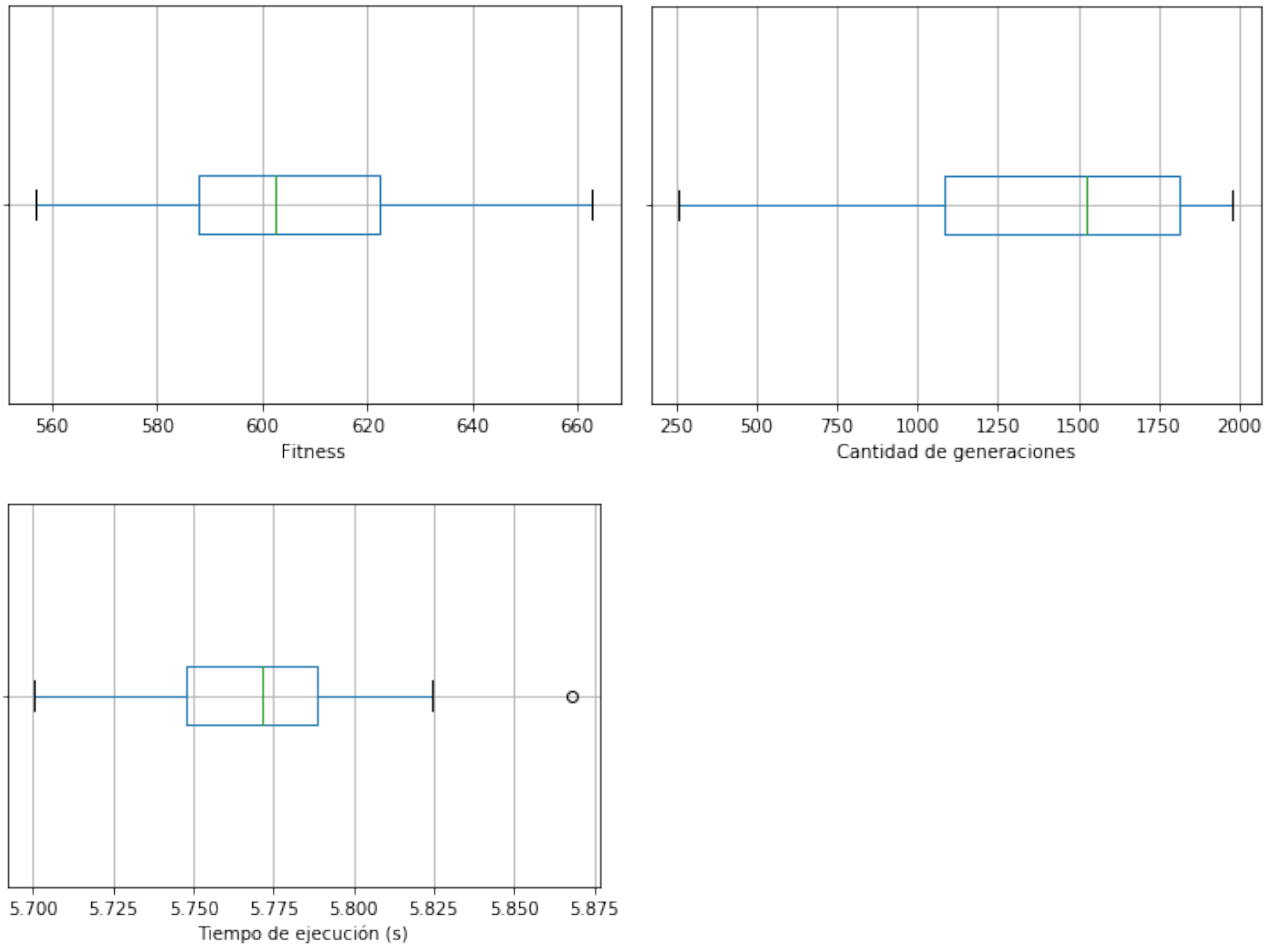


Tabla 6.7: Métricas - Problema 0, configuración seleccionada para MOSES

	Fitness	Generaciones	Tiempo (s)
max	662.83	1978.00	5.87
mean	605.78	1389.18	5.77
std	25.43	478.16	0.03

Figura 6.8: Boxplots de la configuración seleccionada para MOSES



dar igual a la de los resultados, y por lo tanto determinar si los resultados siguen una distribución normal. Como se puede ver en los resultados de la Tabla 6.8, el p-valor obtenido no es menor a 0.05, por lo que no se rechaza la hipótesis nula de que los resultados fueron extraídos de una distribución normal. Tomando en cuenta estos resultados, se asume que las distribuciones son aproximadamente normales, y se realiza un análisis de varianza (ANOVA) para determinar que existen diferencias entre los algoritmos AE y MOSES para las muestras de cada instancia del problema. Los resultados del test para cada instancia del problema fueron un p-valor de 3.28×10^{-29}, 3.94×10^{-46}, 3.59×10^{-27} respectivamente, por lo que se asume que existe una diferencia estadística significativa. En la Tabla 6.9 se presentan métricas relevantes para evaluar la calidad de las soluciones, y en la Figura 6.9 un análisis gráfico.

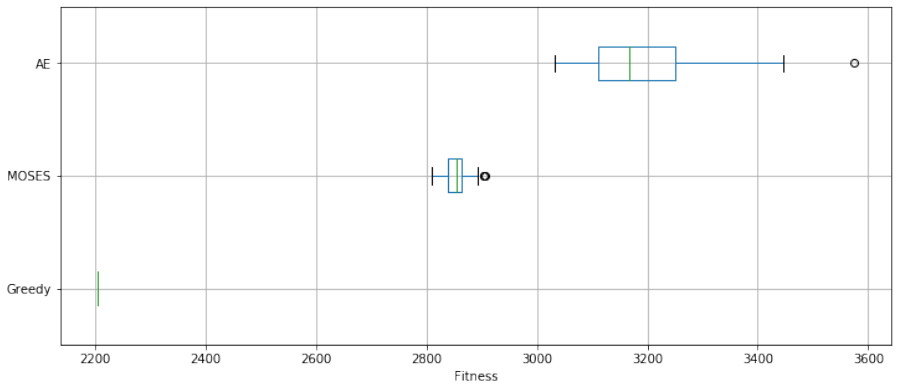
Tabla 6.8: Kolmogorov-Smirnov - AE, MOSES

kstest	instancia #1		instancia #2		instancia #3	
	AE	MOSES	AE	MOSES	AE	MOSES
estadístico	0.10	0.12	0.10	0.10	0.08	0.06
p-valor	0.81	0.55	0.82	0.80	0.90	0.99

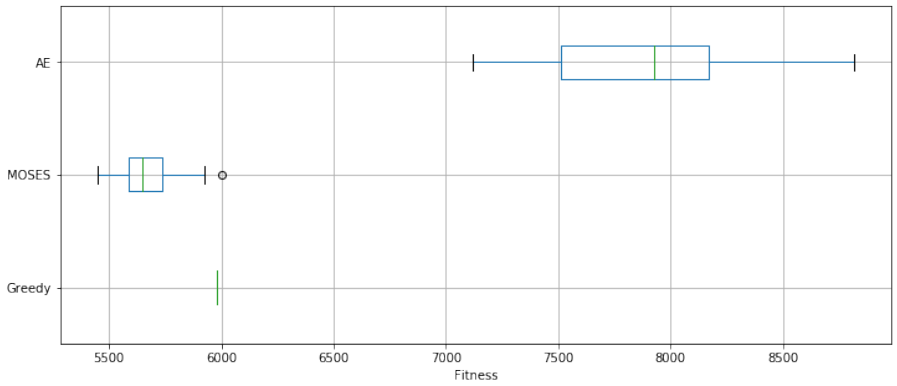
Tabla 6.9: Resultados experimentales: AE vs. MOSES vs. Greedy

	instancia #1			instancia #2			instancia #3		
	AE	MOSES	Greedy	AE	MOSES	Greedy	AE	MOSES	Greedy
max(F)	3573.70	2906.40	2205.00	8816.50	6000.50	5978.40	47018.00	46939.25	17088.00
mean(F)	3187.10	2854.55	-	7879.93	5658.44	-	46973.85	46818.71	-
std(F)	114.32	22.95	-	422.06	125.02	-	17.41	56.05	-
max(gen)	32340.00	1997.00	-	31576.00	1998.00	-	3751.00	1998.00	-
mean(gen)	10843.20	1948.60	-	14830.05	1975.13	-	1681.20	1909.38	-
std(gen)	7179.38	47.96	-	6004.51	33.88	-	837.29	83.33	-
max(T) (s.)	1793.14	107.34	-	4877.15	298.92	-	711.99	247.45	-
mean(T) (s.)	686.37	106.53	0.01	2472.84	296.26	0.14	430.60	244.21	0.02
std(T)	381.77	0.40	-	874.83	1.48	-	98.66	1.67	-

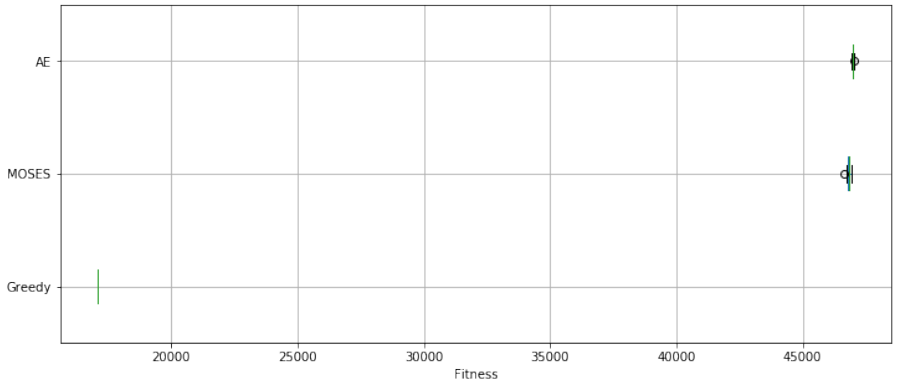
Figura 6.9: Fitness alcanzados por algoritmo



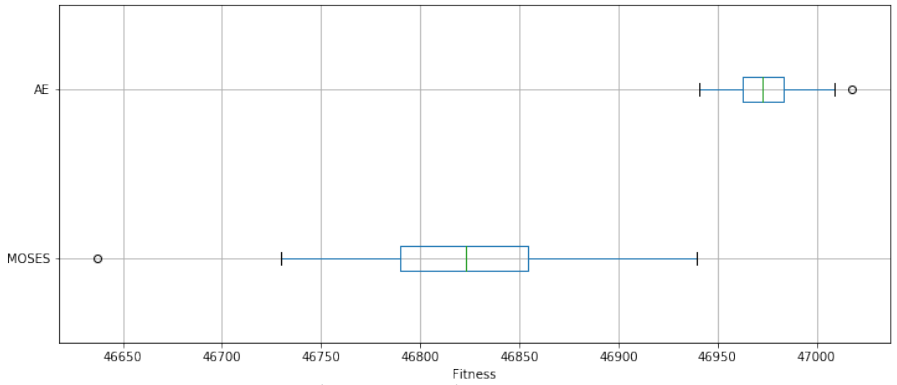
(a) Instancia 1



(b) Instancia 2



(c) Instancia 3



(d) Instancia 3 solo AE y MOSES

6.4. Planificación en línea

Esta sección explica la solución para el vuelo sincronizado en línea. La plataforma de ejecución es el UAV mismo y la comunicación entre los que componen la flota. Se definen las instancias pertinentes al problema del vuelo de la flota de UAV, y los componentes de las mismas. Asimismo las suposiciones utilizadas para el caso.

6.4.1. Instancias del problema

Una instancia está determinada por el entorno, los objetivos y los UAV. Cada instancia tiene solo un entorno, los parámetros para la simulación son las cantidades de UAV y objetivos. Existen también otros elementos que pertenecen al entorno que se podrían tomar como relevantes a la instancia si se se cambiaran las prioridades, por ejemplo la eficiencia del uso de la batería en vez de la cobertura del área. La cantidad de UAVs y objetivos impacta directamente en la cooperación y la vigilancia, muchos objetivos pueden abrumar a la flota, y muchos UAVs pueden llevar a zonas con mayor vigilancia que la que ameritan. Por otro parte este problema es una extensión de la planificación fuera de línea. El punto de partida de este problema sería esta planificación previa y todas sus suposiciones. La solución apunta a mejorar la vigilancia y poder reaccionar a cambios dinámicamente.

Entorno y objetivos

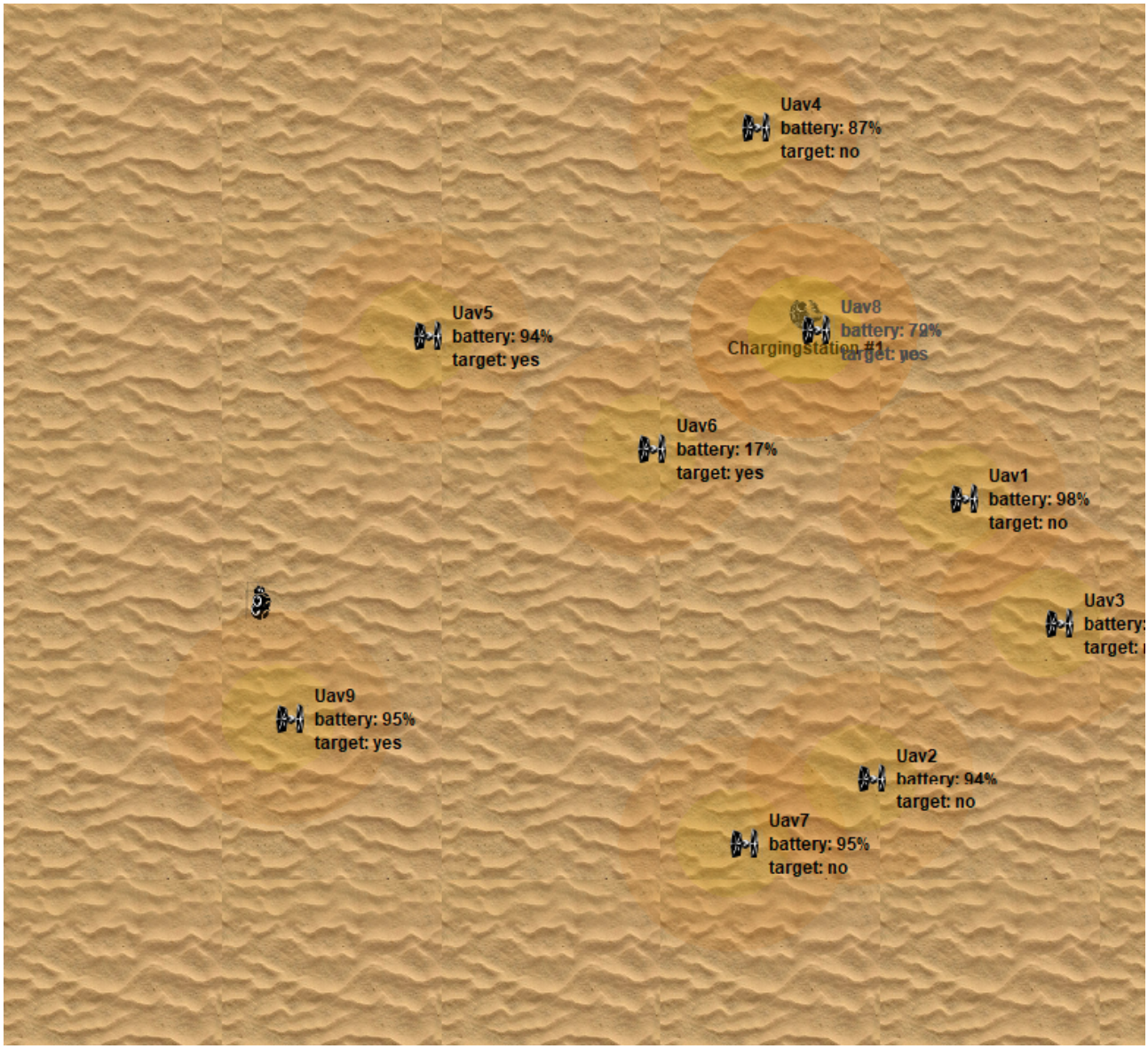
El entorno es un conjunto de elementos que definen la percepción que tiene el UAV de la realidad. Estos elementos quedan determinados por su posición, en cada instante de tiempo, dentro del entorno conocido. Los elementos relevantes del entorno son las bases de carga, el punto de partida de cada UAV y la posición inicial de cada objetivo. Por otra parte el entorno es la abstracción de un área delimitada en la que se debe realizar la vigilancia. En otras palabras el entorno es el mapa a vigilar donde se encuentran los objetivos que son los elementos relevantes para el problema. De los elementos del entorno solo la base de carga y el punto de partida no cambian de posición con el paso de tiempo. En una instancia pueden existir múltiples bases de carga y puntos de partida, pero se optó por tener solo una para cada instancia, determinadas por la posición en el mapa. Para los objetivos se les define un recorrido fijo en cada instancia. El área de interés se mantiene a lo largo de tiempo de una instancia, la dimensión del mapa no cambia con el tiempo.

En la Figura 6.10 se puede observar la vista del simulador donde se ve el entorno. El mapa es un desierto delimitado por el área de vigilancia. Cada dron tiene delimitado el rango de cobertura de conexión inalámbrica y el rango de alcance de vigilancia. Además el simulador marca el estado de la batería, si tiene objetivo a la vista y su nombre.

Auto-piloto

Se realizaron dos soluciones para el comportamiento de la flota, una en la que mantienen el plan fuera de línea, censa pero no toma acciones, y otro en la que reaccionan según lo que censa del

Figura 6.10: Vista del simulador



entorno. La base para la solución es el plan fuera de línea, lo que cambia es el comportamiento al percibir el entorno. El caso de solo seguir el plan fuera de línea implica un menor uso de recursos, con metas basadas en el cubrimiento y menor intercambio de información entre los UAVs. Es la versión más simple e influye directamente la relación entre la cantidad de objetivos a vigilar y la cantidad de UAVs. Por otro lado, en el comportamiento reactivo, cada UAV comparte su información para poder tomar decisiones que mejoren la vigilancia de la flota. Se prioriza la vigilancia entonces el foco esta en mantener en rango de visión a los objetivos. Esto implica una coordinación mucho mayor y uso activo de la información en cada instante de tiempo. Cada UAV toma decisiones con la información local que es compartida por el resto, ya que por más que los mensaje intercambiados son sincronizados la reacción debe ser en tiempo real. La localidad de la información se debe a que las comunicaciones entre los UAV se mantienen de forma local, se generan subgrupos interconectados. Los mensaje se envían cada vez que en la simulación se actualiza la percepción del entorno que recibe el UAV, pero solo si se tiene información relevante a compartir. Por ejemplo cuando se visualiza un objetivo dentro del rango de visión de un UAV se notifica a todos los UAV que estén dentro del rango de comunicación del UAV que visualizó el objetivo. Luego el UAV que este más cerca al objetivo y se desvíe menos de su próxima posición de patrulla notificará a todos dentro de su área de conexión que vigilara el objetivo avistado. De esta forma siempre hay al menos un UAV siguiendo los objetivos y se maximiza la vigilancia a lo largo del tiempo.

6.4.2. Calidad de soluciones

En esta sección proponemos métricas para evaluar la calidad de las soluciones en relación con los objetivos antes mencionados. Las métricas propuestas son en base al tiempo. Una enfocada en la cantidad de objetivos vigilados y otra enfocada en la conectividad entre los UAV. Estas métricas permiten comparar entre una planificación en línea, que ante la percepción del entorno, toma de decisiones cooperativas con otra que no lo hace.

Vigilancia

Se definió como el tiempo en la que se mantiene observado al menos un objetivo sobre el tiempo en que no se observan objetivos. Esta proporción refleja cuán eficiente es la vigilancia.

Conectividad

Siempre que exista conexión entre dos o más UAV, estos comparten información acerca de los objetivos y sus intenciones. Se propone usar la relación entre el tiempo en que permanecen conectados todos los UAV y el tiempo en el que no tienen conexión para determinar cuanto cooperan para tomar decisiones, como se explicó anteriormente se supone que siempre existe conexión entre UAVs que están dentro del rango de señal de red inalámbrica.

6.4.3. Comparación de resultados

A partir de la métricas definidas se obtuvieron datos de la simulación usando dos de los escenarios. Los escenarios son el resultado de la ejecución del algoritmo fuera de línea optimizado, para el problema 1 y el problema 2. Se comparan el vuelo autónomo con reacción y sin reacción.

Tabla 6.10: Resultados experimentales: Porcentaje de tiempo en que se observan objetivos

# vistos (%) por ejecución	problema #1		problema #2	
	Con reacción	Sin reacción	Con reacción	Sin reacción
4	77.15	0.54	47.38	1.28
3	20.04	0.00	36.67	0.26
2	1.76	0.54	8.81	34.10
1	1.05	98.92	7.14	64.36

Tabla 6.11: Resultados experimentales: Porcentaje de tiempo en que los dones están conectados

# conectados (%) por ejecución	problema #1		problema #2	
	Con reacción	Sin reacción	Con reacción	Sin reacción
10	-	-	88.43	89.08
9	-	-	7.44	6.69
8	-	-	1.65	2.06
7	-	-	0.62	0.72
6	-	-	0.62	0.41
5	98.34	99.48	0.31	0.31
4	1.46	0.41	0.21	0.41
3	0.00	0.10	0.62	0.10
2	0.10	0.00	0.10	0.21
1	0.10	0.00	0.00	0.00

Capítulo 7

Conclusiones y trabajo futuro

En el siguiente capítulo se exponen conclusiones referentes al proyecto en general, los resultados alcanzados, dificultades encontradas y se plantean mejoras y trabajos a futuro.

7.1. Conclusiones generales

El problema planteado por el proyecto abarcó muchas áreas de investigación, y como la tecnología de drones es relativamente moderna, la búsqueda de soluciones para el tipo de problema planteado llevo a temas muy diversos. Como observamos durante nuestra participación del curso Path Planning for Mobile Robots in Inspection, Surveillance, and Exploration Missions, dictado por el profesor Martin Saska en la Escuela de Ciencias Informáticas (ECI) de Buenos Aires, el problema puede ser planteado a varios niveles de abstracción, desde un bajo nivel donde se tiene en cuenta cualidades físicas del vuelo y los sensores de los drones, a un alto nivel de abstracción donde esos problemas no existen. Este aspecto dificultó la búsqueda de trabajos relacionados, así como al tomar la dicción de la forma correcta de modelar el problema.

Para el modelado del problema estático se tomo como insumo los resultados del proyecto de la asignatura Algoritmos Evolutivos. El modelado se realizó a un alto nivel de abstracción sin mucho conocimiento de robótica y drones. Dado que el modelado no es estándar, y el problema es bastante específico ya que tiene tres objetivos a optimizar (conectividad, vigilancia y exploración), no fue fácil encontrar algoritmos para comparar la solución, lo que llevó a implementar dos tipos de algoritmos evolutivos, y una versión ávida simple, para poder realizar una evaluación cruzada de las mismas. Se optó por utilizar el método de la suma ponderada para resolver el problema multi-objetivo, ya que la utilización de otro tipo de solución hubiera llevado más tiempo del disponible. El estudio contempla el uso de operadores evolutivos específicos para el problema, la calibración de los diferentes parámetros de los algoritmos y el análisis empírico de su vinculación con la calidad de resultados obtenidos sobre un conjunto de instancias de evaluación. Creemos que las instancias de evaluación son representativas de distintas clases de entornos y objetivos y fueron suficientes para una buena evaluación de las heurísticas. Fueron construidas de forma muy sencilla, eligiendo posiciones alea-

torias, pero buscando que los tres casos estudiados fueran diferentes. Se realizó una simulación del resultado del algoritmo evolutivo con cruzamiento sobre una instancia simplificada en el simulador CoppeliaSim. La simulación se presento en Ingeniería deMustra.

Para el problema en línea se eligió utilizar la programación orientada a agentes ya que en su momento nos visitó un estudiante de maestría de México con conocimientos de este paradigma de programación. Es más que nada utilizada en la robótica, en UAVs la limitante es el poder de cómputo ya que usualmente cuentan con hardware optimizado para alargar la duración de la carga de la batería lo cual reduce las capacidades de cómputo. Igual se opto por utilizar este paradigma de programación y se implemento una solución en el framework JADEX con el modelo BDI. Este framework permite desplegar la solución en dispositivos que ejecuten Java y además utiliza el protocolo FIPA para el intercambio de mensajes entre los agentes, además del protocolo mismo del framework. Teóricamente se podría ejecutar en distintos modelos de UAVs que permitan instalar Java ya que la arquitectura utilizada en la mayoría de los UAVs hoy en día es ARM y existen diversas versiones de Java para ARM. El framework cuenta con herramientas para implementar una simulación muy completa que permite observar el comportamiento de los UAVs en un entorno, el cual se implementó acorde a los parámetros del problema. Con este simulador se pueden obtener datos de la vigilancia y la conectividad. La vigilancia mejora debido a que pueden alterar el plan calculado fuera de línea cuando detectan un objetivo a la vista, de esta forma seguir a los objetivos identificados y mantenerlos vigilados por más tiempo. Eventos como baja batería o pérdida de visión del objetivo llevan a que los UAVs tomen decisiones según prioridades, por ejemplo, mantener la batería cargada tiene mayor prioridad que la vigilancia. Por otro lado la conectividad entre los UAVs no varia significativamente ya sea que estos reaccionen o no al observar un objetivo. En definitiva los resultados del algoritmo en línea, tablas 6.10 y 6.11, permiten observar que la vigilancia mejora ampliamente al usar la información de objetivos observados para tomar decisiones autónomas durante el vuelo. Además, se observa que el numero de UAVs conectados para cada misión se mantiene en valores similares para las versiones con y sin reacción, de donde se desprende que la versión reactiva suma valor a la solución.

Según los resultados podemos observar que el algoritmo evolutivo con cruzamiento es el que obtiene mejores resultados en las tres instancias del problema, aunque la diferencia con el algoritmo MOSES es baja (0.16 %) en la instancia 3 del problema. El mejor fitness alcanzado es aproximadamente 1.23 y 1.47 veces mejor que el del algoritmo MOSES en las instancias 1 y 2 respectivamente. Una posible razón podría ser que el algoritmo MOSES no es tan bueno haciendo que los UAVs sigan a los objetivos, ya que la instancia 3 tiene objetivos estáticos, pero se requieren más instancias de prueba para confirmarlo. El algoritmo MOSES compensa esa diferencia en fitness con su tiempo de ejecución máximo, que es aproximadamente 16.7, 16.32 y 2.88 veces más rápido en las instancias 1, 2 y 3, respectivamente. Por último el algoritmo Greedy tiene un tiempo de ejecución de menos de un segundo, lo que lo hace miles de veces más rápido, y en la instancia 2 es casi igual de bueno en fitness que el algoritmo MOSES (0.36 % de diferencia), pero siguiendo la comparación con el algoritmo MOSES, en la instancia 1 es aproximadamante 1.3 veces peor y en la instancia 3 es aproximadamente 2.75 veces peor. Lo que lo hace aproximadamente 1.62, 1.47, 2.75 veces peor que el algoritmo evolutivo con cruzamiento en las instancias 1, 2 y 3 respectivamente.

Ambos algoritmos evolutivos tienen una amplia gama de aplicabilidad y pueden ser ajustados para resolver variaciones del problema.

7.2. Trabajo futuro

Sobre las principales líneas de trabajo futuro se propone extender el conjunto de problemas de prueba para estudiar mejor el desempeño de los algoritmos en esta clase de problemas, así como compararlos con otras técnicas como PSO (Particle Swarm Optimization). Hacer más pruebas variando parámetros que no fueron estudiados formalmente como la cantidad de puntos del operador de cruzamiento, la variable aleatoria generadora de la cantidad de puntos a mutar en el operador de mutación, incorporación de elitismo, el factor de decaimiento, el número de generaciones y tamaño de población (que no fueron estudiados formalmente por requerir mucho tiempo de cómputo).

Se propone continuar trabajando en llevar la solución a la realidad, utilizando múltiples UAVs y una plataforma de carga para realizar vigilancias continuas. Esto permitirá obtener parámetros reales como tiempo de vuelo, duración de la batería, velocidad máxima, precisión de los UAVs, así como estudiar la conectividad entre los UAVs, y el nivel de dificultad a la hora de la planificación y ejecución de las rutas.

Bibliografía

[1] US Federal Aviation Administration. Aerospace Forecast Report Fiscal Years 2016 to 2036. https://www.faa.gov/news/updates/?newsId=85227. Accedido: 2019-15-12.

[2] Gabrielle Moore. The Commercial Use of Drones in Modern Society. http://pitt.edu/~gym4/trends.html. Accedido: 2019-15-12.

[3] International Civil Aaviation Organization. Unmanned Aircraft Systems (UAS). http://www.icao.int/Meetings/UAS/Documents/Circular%20328_en.pdf. Accedido: 2019-15-12.

[4] R.J. Wood, S. Avadhanula, M. Menon, and R.S. Fearing. Microrobotics using composite materials: the micromechanical flying insect thorax. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. IEEE.

[5] Avid Llc. Honeywell T-Hawk Micro Air Vehicle (MAV), 2019. Accedido: 2019-15-12.

[6] Washington, DC: US Department of Defence. Unmanned systems integrated roadmap. http://archive.defense.gov/pubs/dod-usrm-2013.pdf. Accedido: 2019-15-12.

[7] Inspectie Leefomgeving en Transport ILT/Luchtvaart – Civil Aviation Authority. Remotely piloted aircraft systems (RPAS) in The Netherlands. http://cstwiki.wtb.tue.nl/images/Informatiebulletin-lichte-onbemande-luchtvaart.pdf. Accedido: 2019-15-12.

[8] David Hambling. Longer-Lasting Drones Powered by Fuel Cells. https://www.popularmechanics.com/military/a8956/longer-lasting-drones-powered-by-fuel-cells-15425554/. Accedido: 2019-15-12.

[9] BBC News Services. Google buys solar-powered drone maker Titan Aerospace. https://www.bbc.com/news/business-27029443. Accedido: 2019-15-12.

[10] Juliette Garside. Facebook buys UK maker of solar-powered drones to expand internet. https://www.theguardian.com/technology/2014/mar/28/facebook-buys-uk-maker-solar-powered-drones-internet. Accedido: 2019-15-12.

[11] Laboratory Delft University of Technology. Delfly Drones – Micro Air Vehicles. http://www.delfly.nl. Accedido: 2019-15-12.

[12] Hubsan Ltd. Hubsan Official Site. https://www.hubsan.com. Accedido: 2019-15-12.

[13] Parrot SA. Parrot Drones Official Site. https://www.parrot.com. Accedido: 2019-15-12.

[14] DJI Technology Ltd. DJI Drones Official Site. https://www.dji.com. Accedido: 2019-15-12.

[15] Parrot Group senseFly. eBee Drone. https://www.sensefly.com/drone/ebee-mapping-drone/. Accedido: 2019-15-12.

[16] Military Factory. AeroVironment RQ-11 Raven. https://www.militaryfactory.com/aircraft/detail.asp?aircraft_id=888. Accedido: 2019-15-12.

[17] AeroVironment Inc. UAS: RQ-11B Raven. http://www.avinc.com/uas/view/raven. Accedido: 2019-15-12.

[18] Boeing. Scaneagle unmanned aerial vehicle. http://www.boeing.com/history/products/scaneagle-unmanned-aerial-vehicle.page. Accedido: 2019-15-12.

[19] Brandon C. Welsh and David P. Farrington. Effects of closed circuit television surveillance on crime. *Campbell Systematic Reviews*, 4(1):1–73, 2008.

[20] Harriet Alexander. Dutch police catches cannabis growers. https://www.telegraph.co.uk/news/worldnews/europe/netherlands/11402633/Dutch-police-catch-cannabis-growers-after-spotting-snow-free-roof.html. Accedido: 2019-15-12.

[21] C William R Webster. CCTV policy in the UK: reconsidering the evidence base. *Surveillance and Society*, 6(1):10–22, 2009.

[22] Jelle Brands, Tim Schwanen, and Irina Van Aalst. Fear of crime and affective ambiguities in the night-time economy. *Urban Studies*, 52(3):439–455, 2015.

[23] Lucas Matney. YC-backed Sterblue aims to enable smarter drone inspections. https://techcrunch.com/2018/08/17/yc-backed-sterblue-aims-to-enable-smarter-drone-inspections/. Accedido: 2019-01-05.

[24] EU Commission et al. Communication from the commission to the european parliament and the council: A new era for aviation—opening the aviation market to the civil use of remotely piloted aircraft systems in a safe and sustainable manner, 2014.

[25] R. K. Rhodes. UAS as an Inventory Tool: A Photogrammetric Approach to Volume Estimation. https://scholarworks.uark.edu/etd/2424. Accedido: 2019-15-12.

[26] European RPAS Steering Group. Roadmap for the integration of civil remotely-piloted aircraft systems into the European Aviation System. http://ec.europa.eu/DocsRoom/documents/10484/attachments/1/translations/en/renditions/native. Accedido: 2019-15-12.

[27] Sandi Doughton. Wash. state scientists using drones to spy on nature. https://phys.org/news/2013-07-state-scientists-drones-spy-nature.html. Accedido: 2019-15-12.

[28] Rory Carroll and agencies. US border patrol drone crashes off California coast. https://www.theguardian.com/world/2014/jan/28/us-border-patrol-drone-crashes-california-mexico. Accedido: 2019-15-12.

[29] Deutsche Welle. European Parliament approves Eurosur border surveillance. https://www.dw.com/en/european-parliament-approves-eurosur-border-surveillance/a-17149625. Accedido: 2019-01-05.

[30] Brian Fung. The FAA is letting Hollywood use drones. https://www.washingtonpost.com/news/the-switch/wp/2014/09/25/the-faa-is-letting-hollywood-use-drones-and-more-exemptions-are-on-the-way/?utm_term=.d44ae6e488a7. Accedido: 2019-01-05.

[31] Jeff Beckham. Drones will transform sports photography: once the FAA gets out of the way. https://www.wired.com/2014/09/drones-will-transform-sports-photographyonce-faa-gets-way/. Accedido: 2019-01-05.

[32] Brendan Richardson. Drones could revolutionize weather forecasts, but must overcome safety concerns. https://www.washingtonpost.com/news/capital-weather-gang/wp/2014/04/25/drones-could-revolutionize-weather-forecasts-but-must-overcome-safety-concerns/?utm_term=.a8214ad3d584. Accedido: 2019-01-05.

[33] Euronews. Drone-assisted archeology. https://www.euronews.com/2013/10/15/drone-assisted-archeology. Accedido: 2019-01-05.

[34] Michael Parker. Drones offer 360 degree vision for oil-hunting geologists. The Conversation, 24 Jan 2014. http://theconversation.com/drones-offer-360-vision-for-oil-hunting-geologists-22022. Accedido: 2019-01-05.

[35] Julianne Pepitone. Domino's tests drone pizza delivery. https://money.cnn.com/2013/06/04/technology/innovation/dominos-pizza-drone. Accedido: 2019-15-12.

[36] Daily Mail Reporter. Not just pie in the sky! Russian pizzeria claims to be the first to offer deliveries by helicopter drone. https://www.dailymail.co.uk/news/article-2669157/Not-just-pie-sky-Russian-pizzeria-claims-offer-deliveries-helicopter-drone.html. Accedido: 2019-15-12.

[37] Kelsey D. Atherton. Shanghai Company Claims It Delivered Cakes With Drones. https://www.popsci.com/technology/article/2013-07/shanghai-cake-company-claims-drone-delivery. Accedido: 2019-03-23.

[38] Carlos Felipe. Dronfies en SENAI. https://mundoconectado.com.br/noticias/v/6057/empresa-uruguaia-dronfies-faz-primeiros-testes-de-drones-para-entregas-no-brasil. Accedido: 2019-15-12.

[39] Gregory S. McNeal. Six Things You Should Know About Amazon's Drones. https://www.forbes.com/sites/gregorymcneal/2014/07/11/six-things-you-need-to-know-about-amazons-drones/#11a9c5f36777. Accedido: 2019-15-12.

[40] Michael Reilly. Uber's Ad-Toting Drones Are Heckling Drivers Stuck in Traffic. https://www.technologyreview.com/s/602662/ubers-ad-toting-drones-are-heckling-drivers-stuck-in-traffic/. Accedido: 2019-15-12.

[41] Jane Wells. Could the next wildfire be fought by a drone? https://www.cnbc.com/2014/11/20/the-next-wildfire-could-be-fought-by-a-drone.html. Accedido: 2019-15-12.

[42] Rachel L Finn and David Wright. Unmanned aircraft systems: Surveillance, ethics and privacy in civil applications. *Computer Law & Security Review*, 28(2):184–194, 2012.

[43] John W. Whitehead. Drones Over America, Tyranny at Home. https://archive.org/details/DronesOverAmericaTyrannyAtHome. Accedido: 2019-15-12.

[44] Paul Scharre. Why You Shouldn't Fear "Slaughterbots". https://spectrum.ieee.org/automaton/robotics/military-robots/why-you-shouldnt-fear-slaughterbots. Accedido: 2019-15-12.

[45] Evan Ackerman. Drones Help Rid Galapagos Island of Invasive Rats. https://spectrum.ieee.org/automaton/robotics/drones/drones-help-rid-galapagos-island-of-invasive-rats. Accedido: 2019-15-12.

[46] Jason Koebler. Drones will revolutionize farming first, not delivery. https://motherboard.vice.com/en_us/article/539zyn/drones-will-revolutionize-farming-first-not-delivery. Accedido: 2019-15-12.

[47] Huffington Post. Drones Could Revolutionize Agriculture, Farmers Say. http://flcitrusmutual.com/news/hp_drones_121413.aspx. Accedido: 2019-15-12.

[48] Trevor Mogg. Disney wants to stick projectors on drones and fly them in its parks. https://web.archive.org/web/20160528072440/https://www.i-drone.com.au/projector-drone/. Accedido: 2019-15-12.

[49] Alan Phillips. Spaxel Drones "Draw" Three-Dimensional Figures in Midair. https://dronelife.com/2014/09/26/spaxel-drones-draw-three-dimensional-figures-midair/. Accedido: 2019-15-12.

[50] Intel technologies. Drone Light Shows. https://www.intel.com/content/www/us/en/technology-innovation/aerial-technology-light-show.html. Accedido: 2019-15-12.

[51] Chad R. Frost. Challenges and Opportunities for Autonomous Systems in Space. https://ti.arc.nasa.gov/publications/2030/download. Accedido: 2019-15-12.

[52] Gabriel Hugh Elkaim, Fidelis Adhika Pradipta Lie, and Demoz Gebre-Egziabher. Cognitive networking for uav swarms. In Kimon P. Valavanis and George J. Vachtsevanos, editors, *Handbook of Unmanned Aerial Vehicles*, pages 749–780. Springer Netherlands, 2015.

[53] Librepailot. Librepilot Official Site. https://www.librepilot.org. Accedido: 2019-15-12.

[54] DIYDrones. Openpilot. https://diydrones.com/page/openpilot-1. Accedido: 2019-15-12.

[55] DIYDrones. DIYDrones Official Site. http://diydrones.com/. Accedido: 2019-15-12.

[56] Ardupilot. Ardupilot Official Site. http://ardupilot.org/. Accedido: 2019-15-12.

[57] Pixhawk. Pixhawk official site. https://pixhawk.org/. Accedido: 2019-15-12.

[58] G. Hattenberger, M. Bronz, and M. Gorraz. Using the Paparazzi UAV System for Scientific Research. 2014.

[59] DJI Technology Ltd. DJI Developer Documentation. https://developer.dji.com/mobile-sdk/documentation/. Accedido: 2019-15-12.

[60] Dronecode Project. MAVLink Micro Air Vehicle Communication Protocol. https://mavlink.io/en/. Accedido: 2019-15-12.

[61] El País. Bayardi anunció que apelarán a drones para vigilar la frontera y combatir el narcotráfico. https://www.elpais.com.uy/informacion/politica/bayardi-anuncio-apelaran-drones-vigilar-frontera-combatir-narcotrafico.html. Accedido: 2019-15-12.

[62] El País. Qué puede hacer y qué no un drone en Uruguay. http://web.archive.org/web/20170430011322/http://www.elpais.com.uy/informacion/drones-normativa-volar-uruguay-dinacia.html. Accedido: 2019-15-12.

[63] Dronfies Labs. Página oficial de Dronfies Labs. https://www.dronfieslabs.com. Accedido: 2019-15-12.

[64] Dronfies. Página oficial de Dronfies. https://www.dronfies.com/en/home/. Accedido: 2019-15-12.

[65] Mi Halcón S.A. Página oficial de Mi Halcón. http://mihalcon.com. Accedido: 2019-15-12.

[66] Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics*. Springer Berlin Heidelberg, enlarged 2nd edition, 2004.

[67] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 1st edition, 2006. Available at http://planning.cs.uiuc.edu/.

[68] Stephen J. J. Smith, Dana S. Nau, and Thomas A. Throop. Computer bridge - a big win for ai planning. *AI Magazine*, 19:93–106, 1998.

[69] D. Hahnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*. IEEE.

[70] Steven M Lavalle, David Lin, Leonidas J. Guibas, Jean Claude Latombe, and Rajeev Motwani. Finding an unpredictable target in a workspace with obstacles. *Proceedings - IEEE International Conference on Robotics and Automation*, 1:737–742, 1997.

[71] Ichiro Suzuki and Masafumi Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, 21(5):863–888, 1992.

[72] John W Hartmann. *Counter-intuitive behavior in locally optimal solar sail escape trajectories.* PhD thesis, University of Illinois at Urbana-Champaign, 2005.

[73] John H. Reif. Complexity of the mover's problem and generalizations. In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*, pages 421–427. IEEE Computer Society, 1979.

[74] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[75] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. *Search Algorithms*, 1998.

[76] Thomas Back, David B Fogel, and Zbigniew Michalewicz. Handbook of evolutionary computation. *Evolutionary Computation*, 1997.

[77] S Nesmachnow. Metaheuristics as soft computing techniques for efficient optimization. *Encyclopedia of Information Science and Technology, M. Khosrow-Pour, Ed. IGI Global*, pages 1–10, 2014.

[78] *Evolutionary Computation: The Fossil Record*. Wiley-IEEE Press, 1998.

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

[80] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing , Inc., Boston, MA, USA, 1st edition, 1989.

[81] O. Francois. An evolutionary strategy for global minimization and its markov chain analysis. *IEEE Transactions on Evolutionary Computation*, 2(3):77–90, 1998.

[82] Alain Cercueil and Olivier François. Monte carlo simulation and population-based optimization. In *Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, IEEE*, pages 191–198. Press, 2001.

[83] Yoav Shoham. Agent-oriented programming. *Artificial intelligence*, 60(1):51–92, 1993.

[84] Rafael H Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni. *Multi-Agent Programming:: Languages, Tools and Applications*. Springer Science & Business Media, 2009.

[85] Michael Fisher, Rafael H Bordini, Benjamin Hirsch, and Paolo Torroni. Computational logics and agents: a road map of current technologies and future trends. *Computational Intelligence*, 23(1):61–91, 2007.

[86] Anand S Rao. Agentspeak (l): Bdi agents speak out in a logical computable language. In *European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 42–55. Springer, 1996.

[87] Rafael H Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. John Wiley & Sons, 2007.

[88] Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley-Interscience, 2007.

[89] Lars Braubach, Winfried Lamersdorf, and Alexander Pokahr. Jadex: Implementing a bdi-infrastructure for jade agents, 2003.

[90] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[91] Sameera Ponda, Richard Kolacinski, and Emilio Frazzoli. Trajectory optimization for target localization using small unmanned aerial vehicles. In *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics (AIAA), 2009.

[92] Frank Mufalli, Rajan Batta, and Rakesh Nagi. Simultaneous sensor selection and routing of unmanned aerial vehicles for complex mission plans. *Computers & Operations Research*, 39(11):2787–2799, 2012.

[93] Julien Schleich, Athithyaa Panchapakesan, Grégoire Danoy, and Pascal Bouvry. UAV fleet area coverage with network connectivity constraint. In *Proceedings of the 11th ACM international symposium on Mobility management and wireless access*. Association for Computing Machinery (ACM), 2013.

[94] Hyondong Oh, Hyo-Sang Shin, Seungkeun Kim, Antonios Tsourdos, and Brian A. White. Cooperative mission and path planning for a team of UAVs. In *Handbook of Unmanned Aerial Vehicles*, pages 1509–1545. Springer Nature, 2014.

[95] Ilker Bekmezci, Murat Ermis, and Sezgin Kaplan. Connected multi UAV task planning for flying ad hoc networks. In *2014 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. Institute of Electrical and Electronics Engineers (IEEE), 2014.

[96] Ke Shang, Stephen Karungaru, Zuren Feng, Liangjun Ke, and Kenji Terada. A GA-ACO hybrid algorithm for the multi-UAV mission planning problem. In *2014 14th International Symposium on Communications and Information Technologies (ISCIT)*. Institute of Electrical and Electronics Engineers (IEEE), 2014.

[97] Jian Han, Changhong Wang, and Guoxing Yi. UAV robust strategy control based on MAS. *Abstract and Applied Analysis*, 2014:1–7, 2014.

[98] Qidan Zhu, Yongjie Yan, and Zhuoyi Xing. Robot path planning based on artificial potential field approach with simulated annealing. In *Sixth International Conference on Intelligent Systems Design and Applications*. IEEE, 2006.

[99] Christian Wietfeld and Kai Daniel. Cognitive networking for uav swarms. In Kimon P. Valavanis and George J. Vachtsevanos, editors, *Handbook of Unmanned Aerial Vehicles*, pages 749–780. Springer Netherlands, 2015.

[100] Daniel W. Dyer. Watchmaker Framework for Evolutionary Computation. `http://watchmaker.uncommons.org`. Accedido: 2019-15-12.

[101] Daniel W. Dyer. Sigma scaling. `http://watchmaker.uncommons.org/manual/ch03s05.html`. Accedido: 2019-04-07.