UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Diseño y validación de una herramienta de optimización para la selección de componentes de drones de carreras

Trabajo Fin de Grado

Grado en Ingeniería de Organización Industrial

AUTOR/A: Fiesco Muñoz, Juan Pablo

Tutor/a: Esteso Alvarez, Ana

Director/a Experimental: CASTIBLANCO QUINTERO, JOSE MANUEL

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN ORGANIZACIÓN INDUSTRIAL

DISEÑO Y VALIDACIÓN DE UNA HERRAMIENTA DE OPTIMIZACIÓN PARA LA SELECCIÓN DE COMPONENTES DE DRONES DE CARRERAS

AUTOR: JUAN PABLO FIESCO MUÑOZ

TUTORA: ANA ESTESO ÁLVAREZ
COTUTOR: JOSE MANUEL CASTIBLANCO QUINTERO

Curso Académico: 2021-22

AGRADECIMIENTO

“ Agradecer el gran apoyo de mi tutora y cotutor y la gran dedicación que tienen a sus áreas. Agradecer a mi pareja y amigos por el apoyo moral. ¡Esto no habría sido posible sin vosotros! ”

RESUMEN

La selección de componentes para el montaje de drones de carreras suele ser una tarea altamente delicada, compleja y confusa para los pilotos. Esto es debido a la alta oferta de componentes con unos rangos de rendimiento bastante parecidos. Además, las características inherentes del comportamiento de estos aparatos implican que los pilotos deben poseer cierto conocimiento técnico para asegurar la seguridad de vuelo. La problemática reside en que si esta selección se realiza de forma aleatoria o sin unos conocimientos específicos, esto puede generar que los componentes se quemen o rompan a la hora de hacer el montaje. El presente TFG abordará la necesidad procedente de la toma de decisión sobre qué componentes de drones de carreras seleccionar, a través de la optimización, más concretamente la programación lineal entera mixta (PLEM). Así, el objetivo general es el diseño, desarrollo y validación de una herramienta basada en PLEM que seleccione los componentes, como el tipo de estructura geométrica (airframe) y sus dimensiones, los motores y las hélices, considerando sus condiciones de operación y rendimiento bajo unas condiciones de seguridad establecidas. Drone UPV perteneciente al programa de Generación Espontánea (GE) de la Universitat Politécnica de València ha participado en la definición y validación de la herramienta aportando sus estudios sobre pruebas de vuelo, además de facilitar las preferencias de cuatro de sus pilotos para la validación de la herramienta. Asimismo, la herramienta tendrá en cuenta las preferencias y objetivos de los pilotos, relacionados con la maniobrabilidad, agilidad, empuje, potencia, amperaje y voltaje, así como sus conocimientos respecto a cambios en el firmware, para brindarle una opción óptima dentro de las posibilidades. El modelo se implementa en Pyomo, una extensión de Python que al ser un lenguaje de programación libre facilitará la futura utilización de la herramienta tanto para Drone UPV como para cualquier piloto de drones de carreras.

Palabras clave: optimización; programación lineal entera mixta; selección de componentes; drones de carreras, airframe.

RESUM

La selecció de components per al muntatge de Drons de carreres sol ser una tasca altament delicada, complexa i confusa. Això és degut a l'alta oferta de components i uns rangs de rendiment prou semblants. A més, les característiques inherents del comportament d'aquests aparells impliquen posseir un cert coneixement tècnic per a garantir la seguretat de vol. La problemàtica resideix en què si aquesta selecció es realitza de manera aleatòria o sense uns coneixements específics, això pot generar que els components es cremen o trenquen a l'hora de fer el muntatge. El present TFG abordarà la necessitat procedent de la presa de decisió sobre quins components de drons de carreres seleccionar, a través de l'optimització, més concretament la programació lineal sencera mixta (PLEM). Així, l'objectiu general és el disseny, desenvolupament i validació d'una eina basada en PLEM que seleccioni els components, com la mena d'estructura geomètrica (airframe) i les seues dimensions, els motors i les hèlixs, considerant les seues condicions d'operació i rendiment sota unes condicions de seguretat establertes. Drone UPV pertanyent al programa de Generació Espontània de la Universitat Politècnica de València ha participat en la definició i validació de l'eina aportant els seus estudis sobre proves de vol, a més de facilitar les preferències de quatre dels seus pilots per a la validació de l'eina. Així mateix, l'eina tindrà en compte les preferències i objectius dels pilots, relacionats amb la maniobrabilitat, agilitat, embranzida, potència, amperatge i voltatge, així com els seus coneixements respecte a canvis en el microprogramari, per a brindar-li una opció òptima dins de les possibilitats. El model s'implementa en Pyomo, una extensió de Python. Que a l'ésser un llenguatge de programació lliure facilitarà la futura utilització de l'eina tant per a Drone UPV com per a qualsevol pilot de drons de carreres.

Paraules clau: optimització; programació lineal sencera mixta; selecció de components; dron de carreres; airframe.

ABSTRACT

The selection of components for the assembly of racing drones used to be a delicate, complex and confusing task due to the high market demand and a similar range of efficiency. In addition, the intrinsic characteristics of this machine imply having some technical knowledge to ensure a safe flight. The problem is that if this selection is executed randomly or without some specific knowledge, this may cause components to burn or break easily at assembly time. This TFG will address the decision-making on which racing drone components to employ optimisation, more specifically, Mixed Integer Linear Programming (MILP). Thus, the general objective is the design, development and validation of a tool based on the selected components, such as the type of geometric structure (airframe) and its measurements, motors and propellers, considering their operating conditions and efficiency in the established safety conditions. Drone UPV, which belongs to the Spontaneous Generation programme of the Polytechnic University of Valencia, has participated in the definition and validation of this tool, supporting its studies on flight tests and providing the preferences of four pilots for the validation of the tool. The tool will also consider the pilots' preferences and objectives related to manoeuvrability, agility, thrust, power, amperage and voltage, as well as knowledge of firmware changes, to give them an optimal choice from among all the possibilities. The model is implemented in Pyomo. A Python is a free programming language that will allow the future use of the tool for both the Drone UPV drone and for any racing drone pilot.

Keywords: Optimization computer tool, Mixed Integer Linear Programming Model, selection of components, racing drone, airframe.

ÍNDICE

DOCUMENTOS CONTENIDOS EN EL TFG

ÍNDICE DE LA MEMORIA

ÍNDICE DEL PRESUPUESTO

ÍNDICE DE ANEXOS

MEMORIA

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Objeto

El objetivo general del presente Trabajo Final de Grado (TFG) es diseñar, desarrollar y validar una herramienta de optimización basada en un modelo de Programación Lineal Entera Mixta (PLEM) que permita la selección de componentes de drones de carreras atendiendo al rendimiento esperado y a las preferencias de los pilotos.

La necesidad de esta herramienta surge por la alta oferta y variedad de componentes disponibles en el mercado para el ensamblaje de drones de carreras cuyos rendimientos y coste económico son similares (Banggood, 2022; Aliexpress, 2022). Esto crea confusión entre los pilotos que deben seleccionar los componentes a emplear para ensamblar sus drones de carreras sin conocer cuál es el rendimiento real de dichos componentes, qué componentes son compatibles entre ellos, ni cuales se adecuarán mejor a su forma de pilotar y experiencia.

Para tratar este problema, se ha colaborado con el grupo Drone UPV perteneciente al programa de Generación Espontánea de la Universitat Politècnica de València (UPV). Entre las actividades de Drone UPV, se encuentra el desarrollo de prototipos que posteriormente emplean para participar en competiciones nacionales e internacionales de drones de carreras. Además, cuentan con un alto flujo de estudiantes que se inician en el ensamblaje de drones y que requieren una herramienta que facilite la selección de componentes según sus preferencias, los requerimientos del circuito en el que van a competir, y las características de rendimiento de los componentes disponibles.

Para ello, la herramienta propuesta en el presente TFG se nutre de estudios sobre el rendimiento de los componentes realizados por Drone UPV, y de las preferencias de los pilotos definidas a través de una interfaz de usuario amigable e intuitiva. La herramienta es validada a través de su aplicación a varios casos de estudio, y a cuatro casos reales definidos por pilotos de Drone UPV.

1.2. Motivación y Justificación

La realización del TFG se justifica en base a motivos académicos y profesionales. En relación con los motivos académicos, en el desarrollo del presente TFG se han empleado y se ha profundizado en conocimientos de diferentes asignaturas cursadas a lo largo del Grado en Ingeniería de Organización Industrial (GIOI).

A continuación, se mencionan las asignaturas más relevantes para el desarrollo del proyecto:

- Métodos Cuantitativos de Organización Industrial: Esta asignatura ha sido la principal fuente de conocimientos técnicos para el desarrollo del TFG. En ella se introduce la programación matemática y más concretamente la PLEM que ha sido empleada en el presente TFG para la formulación del modelo de optimización que soporta la selección de drones de carreras. Por otra parte, se han empleado los conocimientos derivados de la programación de heurísticas y metaheurísticas en el tema de optimización combinatoria para programar una interfaz sencilla e intuitiva que permite la traducción de las preferencias de los pilotos en datos de entrada para el modelo de optimización.

- Informática: En esta asignatura se aprende a programar en el lenguaje de programación C++ que es diferente al empleado en el presente TFG. Sin embargo, en ella se adquiere el pensamiento lógico necesario para programar. Esta habilidad ha facilitado la implementación del modelo de PLEM formulado en el lenguaje de programación Python, y más concretamente, en el paquete de optimización Pyomo. Esto ha permitido que en este TFG no solo se formule un modelo de PLEM, sino su implementación, validación, resolución, y obtención de resultados.
- Fundamentos de Organización de Empresa: Esta asignatura ofrece conocimientos acerca de la estructura de las empresas y su organigrama, lo que ha permitido el reconocimiento de las características del grupo Drone UPV, así como entender su organización. Cabe destacar que estos conocimientos también son trabajados en la asignatura Recursos Humanos en Empresas Industriales.
- Proyectos: En esta asignatura se aprende en qué consisten los proyectos, así como las diferentes etapas que hacen que un proyecto sea elaborado de forma exitosa. Estos conocimientos han permitido lograr una visión global del TFG teniendo en cuenta todas sus características desde su planificación hasta su fase de explotación, pasando por organización de tareas y elaboración del presupuesto. Ha sido muy útil a la hora de la organización y planificación de tareas para la consecución del presente proyecto.

A nivel profesional, el desarrollo del presente TFG supone un primer acercamiento a la toma de decisiones en las empresas y al desarrollo de herramientas de soporte a la toma de decisiones basadas en optimización. Además, el desarrollo de la herramienta propuesta para la selección de componentes de drones de carrera permitirá optimizar dicho proceso, reduciendo la posibilidad de desperdiciar componentes (por una selección errónea o incorrecto ensamblaje) y obteniendo la configuración optima del dron según los requisitos de los pilotos. Esta herramienta propuesta no solo podría ser utilizada por Drone UPV para aconsejar a pilotos sobre el dron a emplear, sino que su uso podría ser extendido a las propias empresas de venta de componentes de dron, pudiendo estas ofrecer un nuevo servicio de asesoramiento a sus clientes y cubriendo esta demanda.

1.3. Estructura del TFG

La memoria se compone de ocho capítulos. El primer capítulo muestra el objetivo del TFG justificando su realización desde el punto de vista académico y profesional, y establece la relación con las competencias transversales adquiridas durante el desarrollo del TFG. En el segundo capítulo se introduce al grupo de generación espontánea Drone UPV y sus principales actividades, así como la problemática que se les presenta a la hora de seleccionar los componentes para ensamblar los drones de carreras. En el tercer capítulo se realiza una aproximación teórica a la investigación operativa y a la programación matemática, que es escogida como el método a emplear para soportar el problema bajo estudio. Además, se incluye una relación de los principales softwares de optimización a poder ser empleados en este TFG y la elección del que finalmente se empleará. En el cuarto capitulo se formula el modelo de PLEM multiobjetivo para la selección de componentes de drones de carreras en el que se basará la herramienta desarrollada. En el capítulo cinco se define la herramienta de optimización ligada al modelo propuesto, haciendo hincapié en su arquitectura, las bases de datos empleadas, la interfaz desarrollada para recoger las preferencias de los pilotos y en la propia implementación del modelo. En el sexto capitulo se realiza la validación de la herramienta propuesta y se aplica a cuatro casos reales aportados por pilotos del propio grupo de Generación Espontánea Drone UPV. El séptimo capito remarca las principales conclusiones del TFG así como futuras líneas de actuación. Finalmente, el octavo capitulo muestra la bibliografía utilizada durante el desarrollo del TFG.

Durante el desarrollo del proyecto se ha utilizado la metodología del ciclo de ciclo PDCA *(Plan-Do-Check-Act)*. Los tres primeros capítulos pertenecen a la fase de planificación, los capítulos cuatro y cinco a la etapa hacer, el capítulo seis forma parte de la fase de verificar y el siete de la fase de actuar (Figura 1).

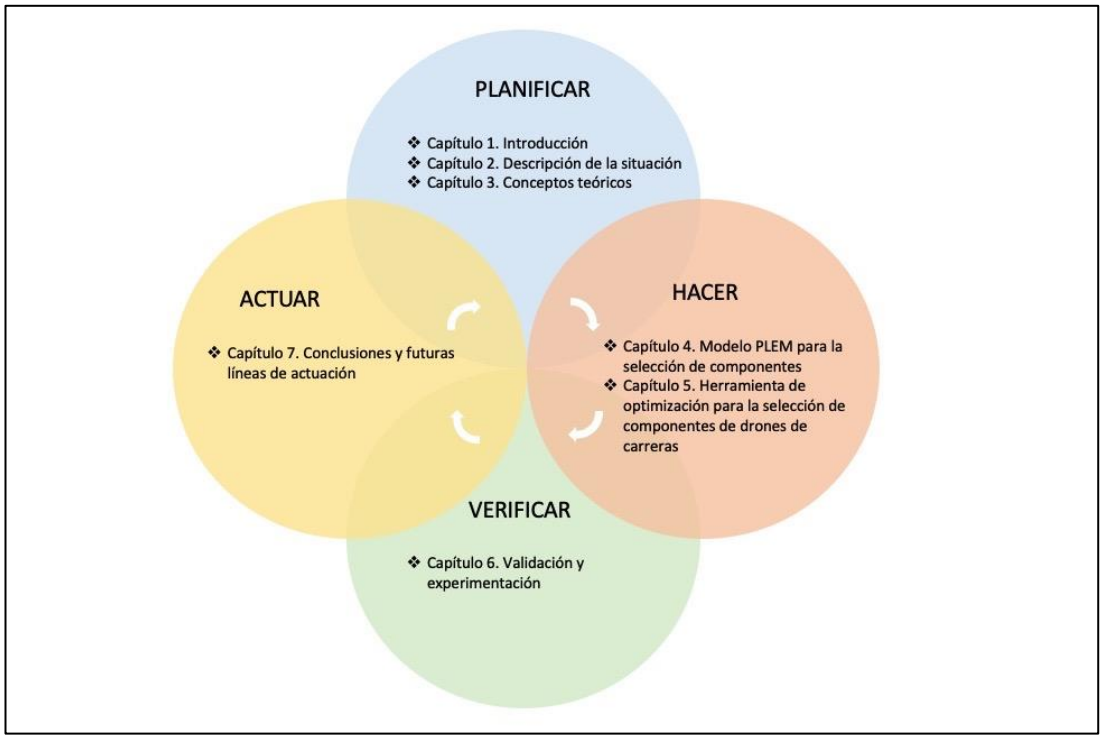

Figura 1. Ciclo de Deming del TFG. Fuente: Elaboración propia basada en García, Quispe, & Ráez (2003)

1.4. Adquisición de Competencias Transversales.

En este apartado se explica cómo se han trabajado cada una de las competencias definidas por la UPV durante el desarrollo del TFG:

CT-01. Comprensión e integración. Esta competencia transversal se ha trabajado a lo largo de todo el TFG con la integración de conceptos ya aprendidos y otros nuevos que se han entendido e identificado para su posterior explicación.

CT-02. Aplicación y pensamiento práctico. A lo largo del presente TFG se ha hecho necesaria esta competencia a la hora de implementar metodologías propias y aprendidas para la consecución de los entregables, solucionar problemas de programación, y sobre todo, en la realización el punto de validación y experimentación desarrollando un metodología de validación y valoración de resultados.

CT-03. Análisis y resolución de problemas. Esta competencia transversal se ha puesto en práctica a la hora de identificar la solución para optimizar la selección de componentes de los drones de carreras, de análisis para la investigación sobre la Investigación Operativa y sus herramientas, resolver los problemas que han ido surgiendo relacionados con la elaboración del modelo, así como con los pequeños obstáculos propios del trabajo autónomo para solucionar problemas de programación en Python.

CT-04. Innovación, creatividad y emprendimiento. Esta competencia transversal se ha trabajado a la hora de modelar el problema creando las asunciones y las variables para dar con un modelo que represente el problema real de selección de componentes y que sea capaz de dar solución al problema. También han sido utilizadas la Innovación y emprendimiento visualizando la necesidad de una interfaz de usuario que perfectamente puede ser comercializable, mediante una hoja Excel desarrollada en Macro para el uso de botones que ejecutan el código de Visual Basic programado.

CT-05. Diseño y proyecto. Al ser un proyecto en sí mismo, se ha tenido que planificar, definir tareas, hacer seguimientos y desarrollar un presupuesto con éxito dentro del plazo estipulado.

CT-06. Trabajo en equipo y liderazgo. Se ha colaborado con integrantes del grupo Drone UPV para lo que ha sido necesario coordinarse y solicitar al grupo la información necesaria para realizar una correcta identificación y descripción del problema a resolver, y para validar la herramienta propuesta para tal fin.

CT-07. Responsabilidad ética, medioambiental y profesional. Una de las ventajas, posiblemente de la más importante, del uso de la herramienta que se ha diseñado es la reducción de residuos electrónicos por incumplir los requisitos de rendimiento o satisfacción del piloto. Esta herramienta pretende optimizar los recursos y seleccionar los mejores componentes dentro de las preferencias del piloto y los objetivos que este le marque.

CT-08. Comunicación efectiva. Se ha trabajado esta competencia durante la escritura de toda la memoria, al tratar de redactar la información de forma clara y coherente.

CT-09. Pensamiento crítico. El pensamiento crítico ha sido una habilidad fundamental durante la definición, modelado e implementación del problema, para conseguir que la herramienta funcione a la máxima perfección sin errores debidas a alguna de las fases que intervienen en el diseño de la herramienta.

CT-10. Conocimiento de problemas contemporáneos. Se ha desarrollado en la investigación de modelos que han tenido como finalidad la solución de un problema real en el contexto de las competiciones de drones de carreras.

CT-11. Aprendizaje permanente. El trabajo autónomo ha contribuido en la mejora de mis habilidades de programación en Python y Visual Basic, modelado de problemas PLEM y conocimientos sobre el funcionamiento de los drones, aprendizajes útiles que seguro serán de utilidad en el futuro para enfrentar próximos retos.

CT-12. Planificación y gestión del tiempo. Durante toda la producción de este TFG se ha hecho muy necesaria esta competencia transversal. Una correcta gestión del tiempo y una planificación meticulosa son elementales esenciales para la consecución de los objetivos tanto personales, académicos y laborales.

CT-13. Instrumental específica. El conocimiento y puesta en práctica de habilidades, conocimientos técnicos y herramientas actualizadas, ha posibilitado la identificación de las herramientas necesarias para la resolución del problema.

Se desarrolla una tabla donde se especifica para cada competencia en que capítulos se ha desarrollado (Figura 2):

	Capítulo 2 **Descripción de la situación actual**	**Capítulo 3** **Conceptos teóricos**	**Capítulo 4** **Modelo de programación lineal entera mixta para la selección de componentes.**	**Capítulo 5** **Herramienta de optimización para la selección de componentes de drones de carreras**	**Capítulo 6** **Validación y experimentación**	**Capítulo 7** **Conclusiones y futuras líneas de actuación**
CT-01. Comprensión e integración.	X	X	X	X	X	X
CT-02. Aplicación y pensamiento práctico.	X		X	X	X	
CT-03. Análisis y resolución de problemas.	X	X		X	X	X
CT-04. Innovación, creatividad y emprendimiento.			X	X	X	X
CT-05. Diseño y proyecto.	X	X	X	X	X	X
CT-06. Trabajo en equipo y liderazgo.	X		X	X		
CT-07. Responsabilidad ética, medioambiental y profesional.	X		X	X	X	X
CT-08. Comunicación efectiva.	X	X	X	X	X	X
CT-09. Pensamiento crítico.			X	X	X	X
CT-10. Conocimiento de problemas contemporáneos	X	X				
CT-11. Aprendizaje permanente	X	X	X	X	X	X
CT-12. Planificación y gestión del tiempo	X	X	X	X	X	X
CT-13. Instrumental específica			X	X	X	

Figura 2. Relación de las competencias desarrolladas en cada capítulo. Fuente: Elaboración propia

2. DESCRIPCIÓN SITUACIÓN ACTUAL

2.1. Introducción

En este capítulo se contextualiza el problema. Para ello se describe el origen del equipo Drone UPV y en qué consisten sus actividades, puesto que es quien tiene esta problemática. Asimismo, el capítulo comprende la historia de la empresa, la misión, visión, estructura organizativa y por último se da paso a la identificación y definición del problema objeto del presente TFG.

2.2. Drone UPV

El proyecto Drone UPV nace de un grupo de investigación llamado Uax, compuesto por profesores de la UPV con un gran currículo y una asombrosa experiencia en proyectos de investigación. Este grupo tenía por objetivo el diseño y fabricación de aeronaves para diferentes aplicaciones, y además colaboraba con varios grupos de investigación del ámbito de la tecnología de Drones. En 2013, en torno a este proyecto se sumaron al equipo 15 estudiantes que al poco tiempo sugirieron realizar su primera competición *indoor* de drones de carreras (DRONE UPV, 2022). Este tipo de carreras consisten en un circuito cerrado con diversos obstáculos los cuales se han de superar lo más rápido posible.

Una vez se establece como objetivo la participación en competiciones de carreras, en 2014 el grupo se presenta a su primera carrera nacional de drones, donde diversos pilotos del ámbito nacional se reúnen para competir por el primer puesto, resultando en todo un éxito. Es entonces cuando el coordinador de Generación Espontánea de la UPV les propone formar parte del programa, lo que les sirve de puente para consolidar el equipo, además de contar con una sinergia que les permitirá evolucionar gracias a los conocimientos de diversos integrantes cualificados y para obtener una red de contactos para futuras colaboraciones (DRONE UPV, 2022). Cabe destacar que Generación Espontánea (GE) es una lanzadera para iniciativas provenientes de los estudiantes de la universidad, al mismo tiempo potencian al máximo las tan aclamadas competencias transversales y la figura del mentor-estudiante (Generación Espontánea UPV, 2022). Al mismo tiempo Drone UPV lleva a cabo proyectos como el diseño y desarrollo con impresión 3D de un novedoso modelo de dron para la alta competición, tomando el nombre de ÍO FPV, tratándose del primer modelo de sus características desarrollado, en ese entonces, íntegramente con tecnología de impresión 3D en el país (Noticias UPV, 2015).

El grupo de investigación Uax pasa a llamarse Spain Drone World University en 2016, tomando el relevo por estudiantes universitarios. El grupo se conservó llevando a cabo actividades de diseño y fabricaciones para mejorar las prestaciones del dron y definir los mejores procesos para la fabricación de un dron ágil y de alto rendimiento, es cuando en 2018 participan en la Carrefour Dron Race, donde reunieron a pilotos de gran prestigio nacional e internacional. A raíz de este evento Drone University decidió enfocarse en los eventos, formalizando Drone UPV, nombre que se le atribuye al equipo de estudiantes encargado no solo de la fabricación de los avanzados drones y creación de eventos sino que también se encargan de fomentar el emprendimiento de los estudiantes y las iniciativas revolucionarias que ayudan al avance global. Es así pues como en 2019 tanto Drone University como Drone UPV definen sus actividades para configurar un futuro donde los drones sean una herramienta esencial para la sociedad con diferentes funcionalidades que satisfagan las necesidades de distintos sectores (DRONE UPV, 2022).

Centrándose en las competiciones de drones de carreras, los organizadores de estos eventos junto con los competidores y sus equipos deben cumplir con el reglamento vigente de competiciones de carreras de drones oficiales. Esta normativa específica las bases de una competición de drones como el tipo de motor que están permitidos utilizar (solo eléctricos), prohibiciones en cuanto a drones preprogramados o sistemas de posicionamiento para la corrección de la dirección, diámetro máximo de las hélices, etc. Es por ello por lo que la herramienta es una gran ventaja a la hora de seleccionar los componentes y es que se limita a la selección de componentes permitidos para este tipo de campeonatos.

2.3. Estructura de la Organización

A continuación, se muestra el organigrama de Drone UPV (Figura 3).

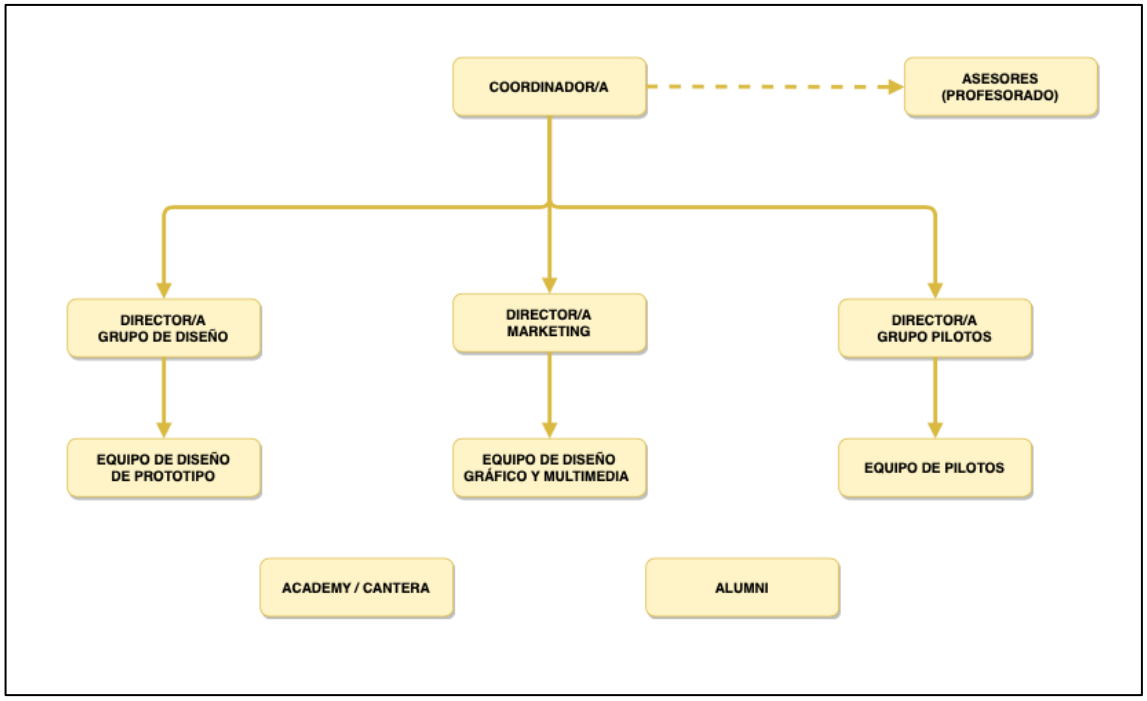

Figura 3: Organigrama de Drone UPV. Fuente: Elaboración propia basada en (DRONE UPV, 2022)

Se puede observar que se trata de un organigrama funcional, ya que se diferencian las funciones principales por departamentos. En la parte superior del organigrama se indican los puestos de mayor responsabilidad de la compañía y en cada línea que emerge hacia abajo y se diversifica aparecen los departamentos funcionales que forman las áreas principales de la compañía, representado de este modo la jerarquía de la compañía. Drone UPV tiene tres áreas de actividad principales:

- Desarrollo de prototipos. Se encargan de la compra de componentes, ensamblaje y mantenimiento de los drones de competición.
- Gestión de eventos. Se encargan de la gestión de los campeonatos a nivel nacional e internacional, además de ocuparse de las redes sociales, las campañas de marketing, grabación de videos y multimedia.

- Pilotaje. Son los pilotos y se encargan de realizar entrenamientos, participar en competiciones nacionales e internacionales, tanto gestionadas por Drone UPV como por otras entidades, y ser la cara del equipo.

Por otro lado, la Academy o Cantera, se compone por estudiantes de la UPV que compaginan sus estudios con tareas dentro de Drone UPV, como desarrollar nuevos prototipos, formaciones sobre drones, soporte de gestión de eventos etc.

2.4. Descripción del Problema

El presente TFG se centra en resolver el problema de la selección de componentes a emplear para ensamblar un dron de carreras que respete las preferencias del piloto y que optimice aquellos parámetros de rendimiento que sean relevantes para el piloto según el circuito al que se enfrenta.

Este proceso de selección de componentes es complejo debido a la gran cantidad y variedad de componentes que pueden ser encontrados en el mercado con unas características y coste similares. Asimismo, la mayoría de los pilotos carecen de los conocimientos técnicos necesarios para poder determinar qué componentes son más apropiados a su estilo de vuelo, o incluso qué componentes son compatibles para ser ensamblados de forma conjunta de forma segura. Cabe recalcar que, si dos componentes no son compatibles y son ensamblados de forma conjunta, puede conllevar que se quemen y, por tanto, no se puedan volver a utilizar.

Es por ello por lo que pilotos, tanto primerizos como avanzados, recurren a Drone UPV para que estos los asesoren sobre qué componentes escoger a la hora de ensamblar un nuevo dron (Figura 4). Actualmente Drone UPV no tiene definida una metodología para elegir los componentes que configuran los drones, sino que seleccionan de forma intuitiva y manual y basándose en sus conocimientos y experiencia previa aquellos componentes que creen serán más apropiados para el piloto. Una vez el dron está ensamblado, se comprueba si la configuración definida cumple con los requisitos de rendimiento definidos por el piloto mediante pruebas de vuelo. En caso de que el rendimiento no cumpla con estos requisitos mínimos, se decide entre cambiar algún componente con el coste que esto supone, o mantener la configuración definida. Por tanto, este es un proceso iterativo y costoso tanto económica como temporalmente, en el cual solo se puede conocer el rendimiento del dron una vez se ha ensamblado.

Figura 4. Symmetrical airframe models (a) HS200. (b) HS210–220. (c) HS200–210. (d) HS240. (e) HS220–230. Fuente: Castiblanco, Garcia-Nieto, Simarro, & Salcedo (2021)

Para empezar a trabajar en la mejora de este proceso de selección de componentes, Drone UPV ha desarrollado un estudio en el que se recogen **datos experimentales** del rendimiento de determinadas configuraciones de drones de carreras. Sin embargo, esta base de datos cuenta con una gran cantidad de información que no se puede analizar manualmente para la selección de los componentes para un determinado circuito y requisitos de piloto.

En el presente TFG, se parte de dicha base de datos que recoge parámetros de rendimiento de los airframes, motores y hélices de los drones de carreras, siendo estos los componentes para los que se desarrolla la herramienta de selección de componentes. A continuación, se define brevemente las principales características de estos tres componentes que se tendrán en cuenta en el TFG:

1. **Airframe:** es la estructura del dron, donde van empotrados los componentes electrónicos. Usualmente es fabricado con fibra de carbono. Los airframes se diferencian básicamente con tres elementos (Figura 5):
    - Longitud: longitud (λ) del Airframe desde el eje de un motor al eje del motor opuesto (comúnmente conocido como wheelbase).
    - Ángulo entre brazos y patas: Son los grados de separación entre los dos brazos (α) y las dos patas (β) del Airframe. Los brazos consisten en los ejes superiores del airframe, mientras que las patas son los dos ejes inferiores.
    - Geometría: el Airframe puede ser simétrico, no simétrico o hibrido. Un airframe es simétrico cuando tanto el ángulo entre sus brazos como el ángulo entre sus patas es de 90º. Un airframe es no simétrico cuando los ángulos entre sus brazos y entre sus patas son iguales e inferiores a 90º. Un airframe es híbrido cuando el ángulo entre sus brazos es igual a 65º, y el ángulo entre sus patas es diferente.

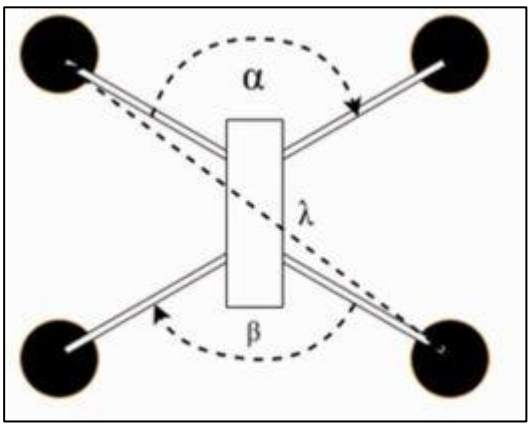

Figura 5: Boceto de las medidas de un dron. Fuente: Castiblanco, Garcia-Nieto, Simarro, & Salcedo (2021)

2. **Motores:** cada dron de carreras lleva montado en sus patas y brazos un motor, para un total de cuatro motores. Todos ellos deben ser exactamente el mismo motor.
3. **Hélices:** cada motor debe ir con su respectiva hélice, que deben ser del mismo tamaño o longitud.

Se escogen estos componentes por su previo estudio, realizado por el grupo de estudiantes Drone UPV, de este modo se utilizan datos comprobados en bancos de pruebas estáticos, de cada una de las combinaciones de estos. Los componentes electrónicos relacionados con las tarjetas de control y variadores de frecuencia (ESC), así como los receptores de video y radiocontrol, el transmisor de video y las respectivas antenas, **se asumen los mismos para todas las configuraciones**.

Asimismo, existen diversos aspectos a tener en cuenta en el proceso de selección de componentes de drones de carreras, relacionados con la compatibilidad de los componentes entre sí y las preferencias de los pilotos:

- En cuanto a la **compatibilidad de los componentes entre sí**:
    - Ciertas geometrías solo permiten unos determinados ángulos y longitudes concretos. Es decir, según el tipo de geometría pueden se pueden considerarse ciertas combinaciones entre ángulos y longitudes, ya que algunas son inexistentes en el mercado.
    - No se pueden colocar cualquier hélice a un motor del dron, ya que según la longitud de esta puede ensamblarse al motor o no.
    - Los componentes por seleccionar dependen también del tipo de batería empleado por el piloto, puesto que algunos de ellos funcionan con baterías de cuatro celdas y otros con baterías de más de cuatro celdas.
- En cuanto a las **preferencias de los pilotos**:
    - Los pilotos pueden definir un valor mínimo y máximo para la longitud del airframe y sus ángulos entre los brazos y patas. En caso de no tener preferencias en este aspecto, se considera que todos los componentes disponibles pueden ser seleccionados.
    - Los pilotos pueden definir la mínima y máxima maniobrabilidad y agilidad requerida en el dron. Un piloto puede requerir diferentes maniobrabilidades y agilidades según el circuito en el que va a participar. En caso de no tener preferencias, se considera el mínimo y máximo posible para estas características, no limitando la selección de componentes.
    - Los pilotos definen el rango de posiciones de la palanca de gas que suelen emplear, para que el rendimiento del dron configurado sea el óptimo para la forma de pilotar del piloto.
    - Los pilotos pueden tener conocimientos de control, pudiendo realizar ajustes en el firmware del controlador o no. En caso de no tener dichos conocimientos, la oferta de componentes a ser seleccionados se ve limitada por la exclusión de aquellos elementos que requieren de este tipo de ajustes.

Una vez conocidos los componentes que se van a considerar en el desarrollo de la herramienta y las limitaciones a tener en cuenta, se da paso a la definición de los parámetros de rendimiento analizados y que se incluyen en la herramienta desarrollada:

- Agilidad: se refiere a la rapidez del movimiento del vehículo. Se considera que existen cinco niveles de agilidad: Muy bajo, Bajo, Medio, Alto y Muy alto.

- Maniobrabilidad: se refiere al control del vehículo en los cambios de dirección, sentido y movimiento. Se considera que existen cinco niveles de maniobrabilidad: Muy bajo, Bajo, Medio, Alto y Muy alto.
- Potencia: este parámetro indica la relación entre el consumo y el voltaje necesario para hacer funcionar con un buen rendimiento las hélices. Su unidad de medida son los vatios (W).
- Amperaje: la cantidad de energía que consume el dron y por tanto necesita para su funcionamiento. Su unidad de medida son los Amperios (A).
- Empuje: cuanto mayor empuje, el dron responde mejor en cuanto a agilidad y maniobrabilidad. Para las pruebas se ha considerado los gramos como unidad de medida.
- Voltaje: cuanto mayor voltaje mayor autonomía y más durabilidad en la conducción. Su unidad de medida son los Voltios (V).

Según el circuito de carreras, así como las preferencias de los pilotos, puede ser conveniente tratar de maximizar o minimizar un parámetro de rendimiento u otro. Asimismo, los pilotos podrían estar interesados en minimizar los costes derivados de la compra de los componentes seleccionados, maximizar el empuje y la maniobrabilidad de forma simultánea.

3. CONCEPTOS TEÓRICOS

3.1. Introducción

En este capítulo se va a explicar los antecedentes de la Investigación Operativa, realizando una clasificación de los problemas que trata, las técnicas o metodologías para abordarlos y las etapas necesarias para el desarrollo del modelo. Asimismo, se profundiza en la programación matemática y se explican los diferentes softwares capaces de soportar el modelo y la técnica que se utiliza (PLEM). Por último se realiza una búsqueda de herramientas de optimización que resuelvan la misma problemática.

3.2. Investigación Operativa

La Investigación Operativa o de Operaciones (IO) surgió a raíz de la Segunda Guerra Mundial, cuando aparece la necesidad de tomar difíciles decisiones respecto a la asignación de recursos a las diferentes actividades referidas a operaciones militares (Corrêa Bernardo, Corrêa Chaves, Gonçalves Sant'Ana, & Pagán Martínez, 2018). Con el objetivo de conseguir la victoria, se formaron varios grupos científicos multidisciplinares de diferentes ramas de la ciencia para investigar sobre métodos matemáticos capaces de mejorar notablemente estrategias o artefactos para la guerra, como el radar (Montufar Benítez, López Pérez, & Flores Cantú, 2018). Pronto la IO comenzó a implementarse en otros sectores y áreas, aplicando las técnicas que más se ajustasen al problema, consiguiendo grandes logros en la agricultura y en lo civil (Taha, 2012).

En 1947 es cuando se oficializa el campo científico denominado IO, de la mano del avance de las computadoras y el desarrollo del *algoritmo simplex* (Dantzig G. B., 1990). Existen distintas definiciones sobre la IO, una de ellas y la que considera más moderna, abarcando las aplicaciones actuales, extraída del libro de Tormos Juan & Lova Ruiz (2003) es:

"*Aplicación de métodos científicos a la mejora de la efectividad de las operaciones, decisiones y gestión. Para ello se utilizarán medios tales como análisis de fatos creación de modelos matemáticos y se propondrán aproximaciones innovadoras; los profesionales de la Investigación Operativa obtienen información sobre una base científica que orienta el proceso de toma de decisiones. Además, ellos son los encargados de desarrollar el software necesario, sistemas, servicios y productos"*.

A lo largo de la época se fue desarrollando más y más esta rama, descubriendo más tipos de problemáticas que podían ser modelados mediante la IO. En la actualidad, existen diversas clasificaciones de los tipos de problemas que se pueden resolver mediante la IO. Según Wilson (1985) los tipos de problema que se pueden abordar mediante la IO son:

- Problemas de colas: cálculo del tiempo de espera según diversas hipótesis.
- Problemas de inventario: cálculo del inventario ideal.
- Problemas de asignación: asignar los recursos disponibles de forma óptima.
- Programación y enrutamiento: elegir la ruta optima o toma de decisiones según escenarios dinámicos.
- Mantenimiento y reemplazo: para el mantenimiento de máquinas.
- Problemas de búsqueda: problemas de secuenciación.

- Teoría de juegos: problemas de estrategia de juego de uno o más jugadores.

Para la resolución de estos problemas existen diversas técnicas o métodos de IO. Cada modelo se compone de distintas tipos de variables y características como la incertidumbre o el número de objetivos. Se ha clasifican según (Mula, Peidro, Díaz-Madroñero, & Vicens, 2010): Programación Lineal, Programación No lineal, Programación Multiobjetivo, Programación difusa, Programación estocástica, Optimización robusta, Simulación, Heurísticas y Metaheurísticas, Modelos híbridos.

Según Taha (2012), para poder resolver un problema a través de cualquier técnica de IO es necesario seguir cinco pasos:

- Identificación y descripción del problema: definición del alcance del problema, es un trabajo que se recomienda hacer en equipo para identificar correctamente las posibles decisiones, determinar los objetivos y las limitaciones del problema.
- Formulación del modelo matemático: transcripción del problema a expresiones matemáticas. Si estas expresiones llegan a ser muy complicadas para su cálculo, se puede realizar un estudio para la simplificación del problema y utilización de métodos heurísticos
- Implementación del modelo en la herramienta: el uso de una calculadora computacional o en este caso un software de optimización capaz de soportar el modelo creado.
- Validación del modelo: comprobar que el modelo se comporta según lo esperado, las soluciones proporcionadas son lógicas, se pueden aceptar intuitivamente los resultados…
- Ejecución del modelo: implica la traducción de las soluciones, a través de la relación de la información de la nomenclatura del modelo, asignada a cada elemento del problema.

Estos pasos presentados son los que se emplean en el desarrollo de la propuesta realizada en el presente TFG para la selección de componentes de drones de carreras (

Figura 6).

Figura 6. Fases de la Investigación operativa. Fuente: Elaboración propia

Dado que la programación matemática ofrece la mejor solución para un determinado conjunto de objetivos, se selecciona esa técnica para la definición de la herramienta. Ya que el objetivo es crear una herramienta que sea capaz de proporcionar la mejor combinación de componentes en base diversos objetivos y requisitos definidos por los decisores.

3.3. Programación Matemática

La Programación Matemática nace de la mano de George Dantzig, en la década de los 1940-1950, durante estos años se manifestaron grandes avances en los métodos matemáticos y la computación, creando una sinergia de conocimientos que posibilitó la creación de grandes herramientas para la resolución de problemas reales (Diaz Muñoz, 2005) . Como se ha comentado anteriormente, el uso de la Programación Matemática es adecuado para problemas con recursos limitados, los cuales se tienen que asignar lo mejor posible en función del objetivo marcado. Dentro de la programación matemática se distinguen dos tipos de modelos según su linealidad (Alemany, 2020):

- **Modelos lineales**: No hay más de una variable multiplicando a los parámetros o coeficientes, es decir no hay más de una variable dependiente asociada al mismo coeficiente. Están compuestos por una Función Objetivo, variables, índices, parámetros y restricciones lineales. En función de la naturaleza de las variables de decisión se pueden clasificar en:
    - Programación lineal continua (PLC): Las variables de decisión son reales y no negativas.
    - Programación lineal entera (PLE) o binaria (PLB): Las variables de decisión tienen que ser enteras o binarias, respectivamente.
    - Programación lineal entera-mixta (PLEM): Las variables de decisión pueden ser enteras, binarias o reales.
- **Modelos no lineales**: cuando existen relaciones no lineales entre variables de decisión, es decir, para la transcripción a expresiones matemáticas se encuentran variables de decisión multiplicadas, divididas o con cualquier otra expresión diferente a la suma o resta entre sí. Impidiendo la linealidad de la función objetivo. Estos modelos suelen proporcionar óptimos locales, ya que debida a su compleja función, aplicando las técnicas actuales, solo se estaría buscando en una zonas concretas, consiguiendo el óptimo de la zona explorada (de la Fuente García & Priore Moreno, 1996).

Particularizando en el problema de selección de componentes de drones de carreras, solo se necesitan variables binarias que indiquen si se escoge o no un componente, por tanto, se debería emplear la PLEM. Además, en el problema bajo estudio se define la posibilidad de optimizar múltiples objetivos de forma simultánea, por lo que la herramienta propuesta deberá ser también multiobjetivo.

3.4. Software de Optimización

Para poder resolver los modelos de programación matemática, es necesario emplear un software de optimización. Se realiza una búsqueda de los posibles softwares de optimización para escoger

el que más se ajuste a las necesidades del problema tiendo en cuenta las limitaciones económicas y la accesibilidad de licencias.

El diseño de este tipo de softwares está orientado a la construcción de modelos matemáticos en base a problemas reales, que no se podrían resolver de forma manual. Tras la búsqueda se clasifican los softwares en base a comerciales y de código abierto. En la Figura 7 se muestran los logos de los softwares capaces de abordar problemas de PLEM y el año de su creación.

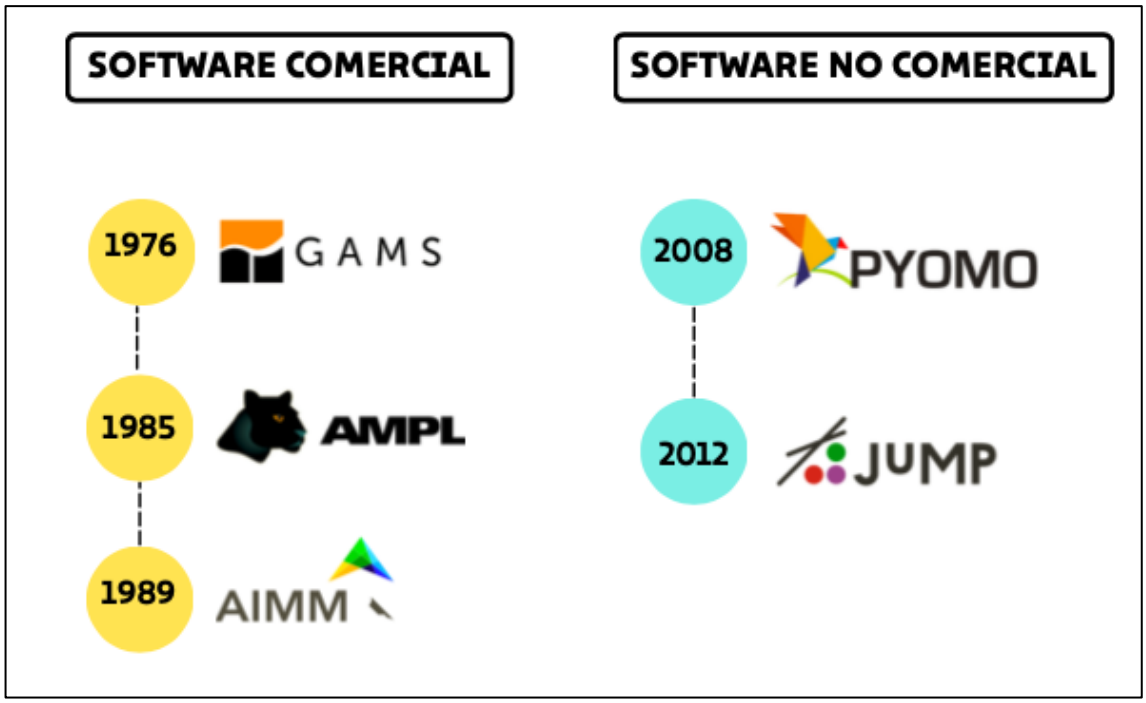

Figura 7. Softwares basados en lenguaje de modelo algebraico. Fuente: Elaboración propia

Por un lado, los softwares comerciales o de código cerrado son aquellos con licencias privadas o de autor, por lo tanto, presentan ciertas limitaciones a la hora de modificarse, distribuirse y usarse. No obstante, tiene algunas ventajas como mejores acabados y un manejo más intuitivo. Concretamente, dentro de los softwares comerciales de optimización se encuentran:

- GAMS, derivado de "General Algebraic Modeling System", dispone de su propio lenguaje algebraico que permite a los usuarios transcribir rápidamente problemas de optimización del mundo real a código informático. Además, su arquitectura proporciona una gran flexibilidad, al posibilitar el cambio a de los Solvers utilizados de forma independiente, sin que el modelo se vea afectado (GAMS, 2022).
- AMPL, proveniente de "A Mathematical Programming Language" es un lenguaje para problemas de programación matemática y optimización a gran escala. Combina notación algebraica familiar y contiene de un poderoso entorno de comando interactivo, además AMPL facilita la creación de modelos, el uso de una amplia variedad de solucionadores y el examen de soluciones (Fourer, 2003).
- AIMMS, acrónimo de "Advanced Interactive Multidimensional Modeling System" incorpora un gran número de funcionalidades enfocadas en la mejora de la gestión del modelo, como una interfaz para los Solvers, que permite el control del rendimiento de su funcionamiento mediante configuraciones de este, herramientas para el análisis de la

solución y modificaciones propias del método aplicado por el Solver para la construcción de la solución mediante métodos y algoritmos elegidos personalmente (Roelofs & Bisschop, 2006).

Por otro lado, los softwares de optimización de código abierto suelen ser paquetes o extensiones de lenguajes de programación de código abierto, esto quiere decir que son códigos que pueden ser ejecutados, modificados y distribuidos sin límites. Otra ventaja de estos es que el coste asociado a su uso es mínimo o incluso gratuito. Como desventajas, es necesario de un cierto conocimiento de programación más avanzado para construir aplicaciones extra que no están incorporadas en el código fuente, como por ejemplo traducir las soluciones para que sean más fáciles de interpretar por el usuario.

- Pyomo, es un paquete de software de optimización derivado de Python centrado en la formulación, resolución y análisis de los modelos matemáticos estructurados, es capaz de crear instancias de problemas de PLEM y resolver estas instancias mediante Solvers comerciales o de código abierto (Hart, 2017). Los elementos del modelado están integrados en lenguaje utilizado, con todas las funciones proporcionando un amplio conjunto de bibliotecas de apoyo.
- JuMP, es un paquete de extensión del lenguaje de programación Julia recientemente nuevo, con una facilidad en su lenguaje de programación de modelos de optimización. Pero con limitada capacidad en problemas de gran complejidad debida a la falta de aplicaciones o elementos para la definición de un robusto modelo (Chen & Brunaud, 2017).

Tras analizar los distintos softwares, se elige el paquete Pyomo, porque además de ser un paquete derivado de un lenguaje de programación de código abierto y ser gratuito, el lenguaje de programación que utiliza, Python, es en la actualidad, uno de los lenguajes de programación más usados en las grandes empresas como Facebook, Google, IBM etc. Siendo este un motivo profesional, se selecciona el uso de Python, ya que se adquiere unos conocimientos de programación en un lenguaje de programación que está muy demandado por las empresas, ganando de esta forma cierta habilidades que pueden ser extrapolados en una futura empresa.

Otro de los motivos de escoger Pyomo, es que ya se había trabajado anteriormente con el lenguaje Python por cuenta propia, por lo tanto, ya lo conocía y me era familiar, haciendo que la curva de aprendizaje para programas con este paquete sea mucho más corta frente a Julia.

Por último, el Solver que se ha seleccionado, siendo este el que se encarga de solucionar el modelo, es Gurobi. A pesar de ser un Solver comercial dispone de una licencia gratuita para estudiantes y académicos.

3.5. Herramientas para la Selección de Componentes de Drones de Carreras

Resumiendo, el problema brevemente, este consiste en la toma de decisión de la configuración de cinco elementos o componentes con unos parámetros medidos que varían según la combinación formada entre ellos. Se desea conseguir la mejor combinación según los objetivos definidos, que respete las restricciones existentes por la compatibilidad entre distintos componentes y las preferencias del piloto.

Para justificar este proyecto como innovación, se ha realizado varias búsquedas en internet y páginas webs como Google Scholar y Scopues, mediante las siguientes palabras clave:

- Optimization model
- Component selection
- Mathematical programming
- Racing drone
- Airframe
- Propeller
- Motor

Tras varias búsquedas se ha comprobado que no existe una herramienta optimización que proponga un modelo para la selección de componentes de drones de carreras tanto a nivel académico como a nivel comercial.

Por otro lado, en lo que respecta a la innovación del TFG, se desarrolla una propuesta de una nueva herramienta, basada en un modelo matemático, que es capaz de optimizar la selección de los componentes de drones de carreras, teniendo en cuenta la preferencia, conocimientos sobre drones de carreras y objetivos de rendimiento del piloto.

Además, dentro de las competiciones de drones de carreras, se debe cumplir con el reglamento vigente de competiciones de carreras de drones oficiales. Esta normativa específica los componentes permitidos, siendo motivo de expulsión no cumplir con el reglamento. Por lo tanto, la herramienta es una gran ventaja a la hora de seleccionar los componentes, ya que no solo tiene en cuenta los objetivos y preferencias del piloto, si no que pretende aportar una solución que cumpla con el reglamento ya que se ha modelado para este especifico sector. Convirtiendo a la herramienta en un sistema único e innovador que puede ser utilizado tanto para Drone UPV como para cualquier empresa que se dedique al sector de los drones de carreras, ya sea vendiendo componentes o desarrollando prototipos.

4. MODELO DE PROGRAMACIÓN LINEAL ENTERA MIXTA PARA LA SELECCIÓN DE COMPONENTES

4.1. Introducción

En este capítulo se definen las hipótesis bajo las que se formula el modelo de PLEM para seleccionar los componentes de un dron de carreras. A continuación, se formula el modelo matemático mediante la definición de su nomenclatura, función objetivo y restricciones.

4.2. Asunciones

Las asunciones proporcionan el orden y lógica al estudio, ayudan a la descripción y explicación, limitando el escenario de posibilidades a una situación descrita y compuesta por cada una de ellas. A continuación, se enuncian las asunciones que permiten la creación del modelo, basándose en las necesidades actuales del equipo Drone UPV:

Unicidad:

1. Solo se puede seleccionar un tipo de airframe caracterizado por su geometría, longitud entre los ejes opuestos, y los ángulos entre las patas y brazos del airframe.
2. Solo se puede seleccionar un tipo de motor por dron.
3. Solo se puede seleccionar una longitud de hélice por dron.

Limitantes para conseguir la personalización del dron:

4. La longitud entre los ejes opuestos del airframe debe ser superior a una longitud mínima definida e inferior a una longitud máxima definida por el piloto según sus preferencias.
5. Los ángulos entre las patas y brazos del airframe deben ser superiores a unos grados mínimos e inferiores a unos grados máximos definidos por el piloto según sus preferencias.
6. La maniobrabilidad del dron debe ser superior a la maniobrabilidad mínima e inferior a la maniobrabilidad máxima definida por el piloto según sus preferencias.
7. La agilidad del dron debe ser superior a la agilidad mínima definida e inferior a la agilidad máxima definida por el piloto según sus preferencias.
8. Los resultados contemplados deben estar dentro de un rango de movimiento del stick izquierdo del mando definido por el piloto según sus preferencias.

Combinación de componentes:

9. Algunas geometrías de airframe solo pueden ser seleccionadas en el caso en que el piloto pueda realizar ajustes de control en el firmware del controlador (necesaria controlabilidad). Por tanto, para los pilotos inexpertos que no tienen el conocimiento suficiente para realizar estos ajustes de control, habrá ciertas opciones de airframe que no estarán disponibles.
10. Según el tipo de batería empleada por el piloto (hasta cuatro celdas o más de cuatro celdas) se podrán emplear un tipo de motores u otros.

11. No todas las hélices se pueden combinar con cualquier tipo de motor ni viceversa, ya que según el motor se puede colocar un tipo hélice con un especifico tamaño.

4.3. Formulación del Modelo

4.3.1. Nomenclatura

En primer lugar, se definen los índices, parámetros y variables de decisión que se van a utilizar para la formulación del modelo matemático. En la primera columna se muestra la nomenclatura definida y en la segunda se muestra su descripción (Tabla 1).

Tabla 1. Nomenclatura. Fuente: Elaboración propia.

Índices	
g	Índice que hace referencia a la geometría del airframe del dron.
l	Índice que hace referencia a la longitud del airframe desde el eje de un motor al eje opuesto.
a	Índice que hace referencia al ángulo de separación entre los brazos del airframe.
b	Índice que hace referencia al ángulo de separación entre las patas del airframe.
m	Índice que hace referencia al motor.
h	Índice que hace referencia a la hélice.
s	Índice que hace referencia a la posición del stick izquierdo del mando
Parámetros	
n	Número de motores y hélices necesarios para configurar el dron.
ca_{glab}	Coste de un airframe con geometría g, longitud l, ángulo de separación a entre los brazos del airframe y ángulo de separación b entre las patas del airframe (€/unidad).
cm_m	Coste de un motor m (€/unidad).
ch_h	Coste de una hélice h (€/unidad).
li_g	Longitud mínima del airframe si este posee geometría g.
la_g	Longitud máxima del airframe si este posee geometría g.
ai_g	Ángulo mínimo entre los brazos (a) en airframes si este posee geometría g.
aa_g	Ángulo máximo entre los brazos (a) en airframes si este posee geometría g.
bi_g	Ángulo mínimo entre las patas (b) en airframes si este posee geometría g.
ba_g	Ángulo máximo entra las patas (b) en airframes si este posee geometría g.
mh_{mh}	Parámetro binario con valor uno cuando es posible montar de forma conjunta el motor m con la hélice h, y cero en caso contrario.
nc_g	Parámetro binario con valor uno cuando es necesaria la controlabilidad (ajustes de control en el firmware del controlador) en el airframe con geometría g, y cero en caso contrario.
pc	Parámetro binario con valor uno cuando el piloto puede realizar ajustes de control en el firmware del controlador debido a sus conocimientos y habilidades, y cero en caso contrario.
nb_m	Parámetro binario con valor uno cuando el motor m funciona con baterías de hasta cuatro celdas, y cero cuando funcionan tanto con baterías de hasta cuatro celdas como con batería de más de cuatro celdas.
pb	Parámetro binario con valor uno cuando el piloto utiliza baterías de hasta cuatro celdas y cero cuando utiliza baterías con más de cuatro celdas.
amp_{smh}	Amperaje de un dron con motores m y hélices h cuando el stick se sitúa en posición s.
emp_{smh}	Empuje de un dron con motores m y hélices h cuando el stick se sitúa en posición s.
vol_{smh}	Voltaje de un dron con motores m y hélices h cuando el stick se sitúa en posición s.
pot_{smh}	Potencia de un dron con motores m y hélices h cuando el stick se sitúa en posición s.

man_{glab}	Maniobrabilidad del dron con airframe de geometría g, longitud l, ángulo de separación a entre los brazos del airframe y ángulo de separación b entre las patas del airframe.
ag_{glab}	Agilidad del dron con airframe de geometría g, longitud l, ángulo de separación a entre los brazos del airframe y ángulo de separación b entre las patas del airframe.
$lmax$	Mínima longitud (l) entre dos ejes opuestos del airframe requerida por el piloto.
$lmin$	Máxima longitud (l) entre dos ejes opuestos del airframe requerida por el piloto.
$amin$	Mínimo ángulo (a) de separación entre los brazos del airframe requerida por el piloto.
$amax$	Máximo ángulo (a) de separación entre los brazos del airframe requerida por el piloto.
$bmin$	Mínimo ángulo (b) de separación entre las patas del airframe requerida por el piloto.
$bmax$	Máximo ángulo (b) de separación entre las patas del airframe requerida por el piloto.
$smin$	Posición mínima del stick s empleada por el piloto.
$smax$	Posición máxima del stick s empleada por el piloto.
$mamin$	Maniobrabilidad mínima requerida por el piloto.
$mamax$	Maniobrabilidad máxima requerida por el piloto.
$agmin$	Agilidad mínima requerida por el piloto.
$agmax$	Agilidad máxima requerida por el piloto.
Variables de decisión	
Y_{glabmh}	Variable binaria que toma valor uno si se configura el dron con un airframe con geometría g, longitud l, ángulo de separación a entre los brazos del airframe y ángulo de separación b entre las patas del airframe, con motores m y hélices h, y cero en caso contrario.
YA_{glab}	Variable binaria que toma valor uno si se configura el dron con un airframe con geometría g, longitud l, ángulo de separación a entre los brazos del airframe y ángulo de separación b entre las patas del airframe, y cero en caso contrario.
YM_m	Variable binaria que toma valor uno si se configura con motores m, y cero en caso contrario.
YH_h	Variable binaria que toma valor uno si se configura el dron con hélices h, y cero en caso contrario.
YS_{smh}	Variable binaria que toma valor uno si se configura el dron con motores m y hélices h para ser operado con el stick en posición s, y cero en caso contrario.

4.3.2. Función objetivo

El modelo propuesto considera un total de siete objetivos que se combinan posteriormente mediante el método de la suma ponderada. Esto permite que el modelo pueda ser ejecutado para optimizar un solo objetivo o más en función de las preferencias del piloto que desee configurar su dron. Además, seis de estos siete objetivos pueden ser minimizados o maximizados en función de las preferencias del piloto que desea configurar su nuevo dron. Así, un objetivo se encuentra relacionado con el aspecto económico mientras que el resto se encuentran relacionados con diferentes características de rendimiento del dron.

- Minimización de costes (Z_1). Se consideran los costes derivados de la elección del tipo de airframe, motores y hélices a emplear en el dron (1). Cada variable de decisión lleva multiplicando el parámetro que hace referencia al coste unitario del componente en particular.

$$Min\ Z_1 = \sum_{g=1}^{G}\sum_{l=1}^{L}\sum_{a=1}^{A}\sum_{b=1}^{B} ca_{glab} * YA_{glab} + \sum_{m=1}^{M} cm_m * n * YM_m + \sum_{h=1}^{H} ch_h * n * YH_h \quad (1)$$

- Minimización o maximización del amperaje (Z_2). Las ecuaciones (2) y (3) expresan el sumatorio de todas las posibilidades que permite la variable de decisión (YS_{smh}), que indica la combinación del tipo de motor y hélice, además de las posiciones del stick posibles según la forma de pilotaje del piloto, multiplicado por el valor del amperaje que proporciona esa configuración de motor y hélice, en sus distintas posiciones del stick. Se utilizan los objetivos de minimización (2) o maximización (3) para que el parámetro sea los más bajo o alto posible según la necesidades del piloto, el circuito u ambos.

$$\text{Min } Z_2 = \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{h=1}^{H} amp_{smh} * YS_{smh} \quad (2)$$

$$\text{Max } Z_2 = \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{h=1}^{H} amp_{smh} * YS_{smh} \quad (3)$$

- Minimización o maximización del empuje (Z_3). Las ecuaciones (4) y (5) expresan el sumatorio de todas las posibilidades que permite la variable de decisión (YS_{smh}), que indica la combinación del tipo de motor y hélice, además de las posiciones del stick posibles según la forma de pilotaje del piloto, multiplicado por el valor del empuje que proporciona esa configuración de motor y hélice, en sus distintas posiciones del stick. En función de las dificultas de circuito y las preferencias del piloto se puede escoger entre minimizar (4) o maximizar (5) el empuje

$$Min\ Z_3 = \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{h=1}^{H} emp_{smh} * YS_{smh} \quad (4)$$

$$Max\ Z_3 = \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{h=1}^{H} emp_{smh} * YS_{smh} \quad (5)$$

- Minimización o maximización del voltaje (Z_4). Las ecuaciones (6) y (7) expresan el sumatorio de todas las posibilidades que permite la variable de decisión (YS_{smh}), que indica la combinación del tipo de motor y hélice, además de las posiciones del stick posibles según la forma de pilotaje del piloto, multiplicado por el valor del voltaje que proporciona esa configuración de motor y hélice, en sus distintas posiciones del stick. En función de lo que se desee se puede minimizar (6) o maximizar (7) este parámetro.

$$Min\ Z_4 = \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{h=1}^{H} vol_{smh} * YS_{smh} \quad (6)$$

$$Max\ Z_4 = \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{h=1}^{H} vol_{smh} * YS_{smh} \quad (7)$$

- Minimización o maximización de la potencia (Z_5). Las ecuaciones (8) y (9) expresan el sumatorio de todas las posibilidades que permite la variable de decisión (YS_{smh}), que indica la combinación del tipo de motor y hélice, además de las posiciones del stick posibles según la forma de pilotaje del piloto, multiplicado por el valor de la potencia que proporciona esa configuración de motor y hélice, en sus distintas posiciones del stick. Según la conducción del piloto y las características del circuito se querrá minimizar (8) o maximizar (9) la potencia.

$$Min\ Z_5 = \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{h=1}^{H} pot_{smh} * YS_{smh} \tag{8}$$

$$Max\ Z_5 = \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{h=1}^{H} pot_{smh} * YS_{smh} \tag{9}$$

- Minimización o maximización de la maniobrabilidad (Z_6). Las ecuaciones (10) y (11) expresan el sumatorio de todas las configuraciones posibles que aglutina la variable de decisión que indica las características del airframe escogido (YA_{glab}), es decir la combinación del tipo de geometría, longitud y ángulos, multiplicada por el valor de la maniobrabilidad que da la configuración en concreto. Dependiendo del circuito el piloto puede preferir la configuración de un dron con la mínima (10) o máxima (11) maniobrabilidad posible.

$$Min\ Z_6 = \sum_{g=1}^{G}\sum_{l=1}^{L}\sum_{a=1}^{A}\sum_{b=1}^{B} man_{glab} * YA_{glab} \tag{10}$$

$$Max\ Z_6 = \sum_{g=1}^{G}\sum_{l=1}^{L}\sum_{a=1}^{A}\sum_{b=1}^{B} man_{glab} * YA_{glab} \tag{11}$$

- Minimización o maximización de la agilidad. Las ecuaciones (12) y (13) expresan el sumatorio de todas las configuraciones posibles que aglutina la variable de decisión que indica las características del airframe escogido (YA_{glab}) multiplicada por el valor de la agilidad que da la configuración en concreto. Dependiendo del circuito el piloto puede preferir la configuración de un dron con la mínima (12) o máxima (13) agilidad posible.

$$Min\ Z_7 = \sum_{g=1}^{G}\sum_{l=1}^{L}\sum_{a=1}^{A}\sum_{b=1}^{B} ag_{glab} * YA_{glab} \tag{12}$$

$$Max\ Z_7 = \sum_{g=1}^{G}\sum_{l=1}^{L}\sum_{a=1}^{A}\sum_{b=1}^{B} ag_{glab} * YA_{glab} \tag{13}$$

Con objeto de construir una única función objetivo (14), se emplea el método de la suma ponderada. Este método consiste en asignar pesos a los objetivos según las preferencias de los decisores, asignando un mayor peso a aquellos objetivos que sean más relevantes para el decisor. En este caso el decisor es el piloto que desea configurar un dron de carreras. Los pesos asignados a los objetivos (w_o, donde o es el objetivo al que se hace referencia) deben adquirir valores entre cero y uno, asegurando que la suma de los pesos asignados a todos los objetivos sea igual a uno

($\sum_o w_o = 1$). Como hay algunos objetivos que están maximizando y otros minimizando, se le asigna el signo positivo (+) o negativo (–) en función de si se maximiza o minimiza el objetivo respectivamente.

Como además cada uno de los objetivos tiene un orden de magnitud diferente y tienen diferentes unidades, se escalan los valores para que el modelo funcione bien. Para ello, dividimos el valor del objetivo por el máximo valor que se puede obtener para ese objetivo ($maxZo$, donde o es el objetivo al que se hace referencia). Así todos adquirirán valores entre cero y uno.

$$Max\ Z = -w_1 * \frac{Z_1}{maxZ1} \pm w_2 * \frac{Z_2}{maxZ2} \pm w_3 * \frac{Z_3}{maxZ3} \pm w_4 * \frac{Z_4}{maxZ4} \pm w_5 * \frac{Z_5}{maxZ5} \pm w_6 * \frac{Z_6}{maxZ6} \pm w_7 * \frac{Z_7}{maxZ7} \quad (14)$$

4.3.3. Restricciones

El modelo está sujeto a las siguientes restricciones:

El dron se configura con un solo airframe caracterizado por su geometría, longitud entre ejes opuestos, y ángulos entre brazos y patas, así como con un solo tipo de motor y hélice (15).

$$\sum_g \sum_l \sum_a \sum_b \sum_m \sum_h Y_{glabmh} = 1 \quad (15)$$

La restricción (16) define la relación entre la variable de decisión que indica la configuración del dron completo (Y_{glabmh}) y la variable de decisión que indica las características del airframe escogido (YA_{glab}).

$$\sum_m \sum_h Y_{glabmh} = YA_{glab} \qquad \forall g, l, a, b \quad (16)$$

La restricción (17) define la relación entre la variable de decisión que indica la configuración del dron completo (Y_{glabmh}) y la variable de decisión que indica el tipo de motor escogido (YM_m).

$$\sum_g \sum_l \sum_a \sum_b \sum_h Y_{glabmh} = YM_m \qquad \forall m \quad (17)$$

La restricción (18) define la relación entre la variable de decisión que indica la configuración del dron completo (Y_{glabmh}) y la variable de decisión que indica el tipo de hélice escogida escogido (YH_h).

$$\sum_g \sum_l \sum_a \sum_b \sum_m Y_{glabmh} = YH_h \qquad \forall h \quad (18)$$

La restricción (19) define la relación entre la variable de decisión que indica la configuración del dron completo (Y_{glabmh}), y la variable de decisión que indica el tipo de motores y hélices escogidas, y las posiciones del stick empleadas por el piloto (YS_{smh}). Además, se debe tener en

cuenta que no se deben considerar las posiciones las posiciones de stick no permitidas, por ser superiores a la posición máxima (20) o inferiores a la posición mínima (21) requerida por el piloto.

$$\sum_{g}\sum_{l}\sum_{a}\sum_{b} Y_{glabmh} = YS_{smh} \qquad \forall smin \geq s \geq smax, m, h \tag{19}$$

$$\sum_{m}^{M}\sum_{h}^{H} YS_{smh} = 0 \qquad \forall s > smax \tag{20}$$

$$\sum_{m}^{M}\sum_{h}^{H} YS_{smh} = 0 \qquad \forall s < smin \tag{21}$$

Solo se pueden utilizar ciertas combinaciones de motores y hélices debido a la compatibilidad entre ellas por el modelo o marca del motor y la de las hélices. Por ello se utiliza la expresión matemática (22) que restringe las posibilidades de la variable de decisión que indica el tipo de motores y hélices escogidas, y las posiciones del stick empleadas (YS_{smh}) con el parámetro que indica las combinaciones posibles (mh_{mh}).

$$YS_{smh} \leq mh_{mh} \qquad \forall s, m, h \tag{22}$$

La longitud del airframe debe ser superior a una longitud mínima ($lmin$) definida por el piloto, e inferior a una longitud máxima ($lmax$) definida por el piloto según sus preferencias. Por tanto, no se puede escoger un airframe con longitud menor a la longitud mínima, o con longitud mayor a la longitud máxima (23).

$$\sum_{g=1}^{G}\sum_{l\geq lmin}^{lmax}\sum_{a=1}^{A}\sum_{b=1}^{B} YA_{glab} = 1 \tag{23}$$

Los ángulos a y b del airframe debe ser superior a los ángulos mínimos ($amin$) y ($bmin$) definidos por el piloto, e inferior a unos ángulos máximos ($amax$) y ($bmax$) definidos por el piloto según sus preferencias. Por tanto, no se puede escoger un airframe con ángulos inferiores a los ángulos mínimos o superiores a los ángulos máximos (24) y (25).

$$\sum_{g=1}^{G}\sum_{l=1}^{L}\sum_{a\geq amin}^{amax}\sum_{b=1}^{B} YA_{glab} = 1 \tag{24}$$

$$\sum_{g=1}^{G}\sum_{l=1}^{L}\sum_{a=1}^{A}\sum_{b\geq bmin}^{bmax} YA_{glab} = 1 \tag{25}$$

Algunas geometrías necesitan de ajustes en el firmware del controlador para el correcto funcionamiento del dron, esta condición la determina el parámetro (nc_g). La ecuación (26) restringe las posibilidades de ciertas geometrías imposibilitando la opción de escoger una geometría con necesidad de ajustes en caso de que el piloto especifique que no tiene los conocimientos suficientes para realizar los ajustes pertinentes.

$$\sum_{l=1}^{L}\sum_{a=1}^{A}\sum_{b=1}^{B} nc_g * YA_{glab} \leq pc \qquad \forall g \tag{26}$$

Todos los motores pueden ser seleccionados si el piloto emplea baterías de hasta cuatro celdas, pero solo algunos motores pueden ser empleados con baterías de más de cuatro celdas. Esta expresión (27) es similar a la anterior, solo que esta afecta únicamente al tipo de motor que puede ser utilizado, limitando el valor de la variable de decisión que indica el tipo de motor (YM_m) con el parámetro (nb_m) que indica los motores que funcionan con cuatro celdas o más y el valor introducido por el piloto (pb).

$$nb_m * YM_m \leq pb \qquad \forall m \tag{27}$$

La maniobrabilidad del dron debe ser superior a la maniobrabilidad mínima definida *(*$mamin$) por el piloto, e inferior a la maniobrabilidad máxima definida por el piloto ($mamax$). Mediante la expresión (28) se limita el valor de la maniobrabilidad que toma con la combinación del airframe escogida.

$$mamin \leq \sum_{g=1}^{G} \sum_{l=1}^{L} \sum_{a=1}^{A} \sum_{b=1}^{B} man_{glab} * YA_{glab} \leq mamax \tag{28}$$

La agilidad del dron debe ser superior a la agilidad mínima definida por el piloto ($agmin$), e inferior a la agilidad máxima definida por el piloto ($agmax$). Mediante la expresión (29) se limita el valor de la agilidad que toma con la combinación del airframe escogida.

$$agmin \leq \sum_{g=1}^{G} \sum_{l=1}^{L} \sum_{a=1}^{A} \sum_{b=1}^{B} ag_{glab} * YA_{glab} \leq agmax \tag{29}$$

No es posible seleccionar airframes con longitudes no posibles para la geometría escogida. Los parámetros ($lmin_g$) y ($lmax_g$) limitan las posibles longitudes que se pueden escoger según la geometría (30) y (31).

$$\sum_{l=1}^{lmin_g-1} \sum_{a=1}^{A} \sum_{b=1}^{B} YA_{glab} = 0 \qquad \forall g \tag{30}$$

$$\sum_{l=lmax_g+1}^{L} \sum_{a=1}^{A} \sum_{b=1}^{B} YA_{glab} = 0 \qquad \forall g \tag{31}$$

No es posible seleccionar airframes con ángulos entre los brazos no posibles para la geometría escogida. Lo mismo ocurre con los ángulos de las patas. El conjunto de las restricciones (32), (33), (34) y (35) imposibilitan la selección de las configuraciones que no pueden ser posibles debidas los valores que pueden tomar los ángulos entre las patas (a) y los brazos (b) del airframe en función de la geometría seleccionada.

$$\sum_{l=1}^{L} \sum_{a=1}^{amin_g-1} \sum_{b=1}^{B} YA_{glab} = 0 \qquad \forall g \tag{32}$$

$$\sum_{l=1}^{L} \sum_{a=amax_g+1}^{A} \sum_{b=1}^{B} YA_{glab} = 0 \qquad \forall g \tag{33}$$

$$\sum_{l=1}^{L}\sum_{a=1}^{A}\sum_{b=1}^{bmin_g-1} YA_{glab} = 0 \qquad \forall g \tag{34}$$

$$\sum_{l=1}^{L}\sum_{a=1}^{A}\sum_{b=bmax_g+1}^{B} YA_{glab} = 0 \qquad \forall g \tag{35}$$

En los airframes de geometría simétrica (g=1) y los airframes de geometría no simétrica (g=2) se debe asegurar que el ángulo entre las patas y el ángulo entre los brazos sean iguales, por lo que hay que imposibilitar que el resto de las opciones sean posibles (36).

$$\sum_{l=1}^{L}\sum_{a=1}^{A}\sum_{b\neq a} YA_{glab} = 0 \qquad \forall g \leq 2 \tag{36}$$

4.4. Conclusiones

Como conclusión se puede observar en los apartados anteriores que la construcción de un modelo conlleva unos pasos previos necesarios para la coherencia de la función objetivo, restricciones, parámetros e índices.

Asimismo, se propone un modelo de PLEM para la selección de los componentes de un dron de carreras, que respeta las limitaciones existentes por las posibles combinaciones entre los tipos de airframes, la capacidad de programar el firmware, las preferencias determinadas por el piloto en cuanto a maniobrabilidad, agilidad, empuje, voltaje y en cuanto a sus preferencias con respecto a las características de los componentes como los motores según el tipo de batería, y todas las demás restricciones que se han definido y descrito con detalle.

La definición de cada uno de los elementos que forman el modelo es imprescindible para su correcto funcionamiento, si faltase alguna de las ecuaciones o índices, el modelo no estaría valorando todas las condiciones del problema y por lo tanto no estaría dando una solución óptima fiable para el problema real. Es por ello por lo que es muy importante asegurarse de que el modelo se ajusta al problema y contempla todas las asunciones que se han determinado.

En definitiva, el modelo en si es la representación matemática que expresa todas las asunciones que se definen y dimensionan la herramienta, mediante las fórmulas matemáticas lógicas, los parámetros e índices que se han definido con anterioridad.

5. HERRAMIENTA DE OPTIMIZACIÓN PARA LA SELECCIÓN DE COMPONENTES DE DRONES DE CARRERAS

5.1. Introducción

A continuación, se procede a realizar la descripción de la arquitectura de la herramienta desarrollada, aportando un esquema que ejemplifica claramente la relación entre los elementos. Además, se describe la estructura de la base de datos utilizada y, finalmente, se presenta el código de programación en el software utilizado para resolver el modelo matemático propuesto.

5.2. Arquitectura de la Herramienta

La herramienta de optimización se compone por nueve elementos que se interconectan entre ellos para permitir la resolución del modelo PLEM propuesto (Figura 7).

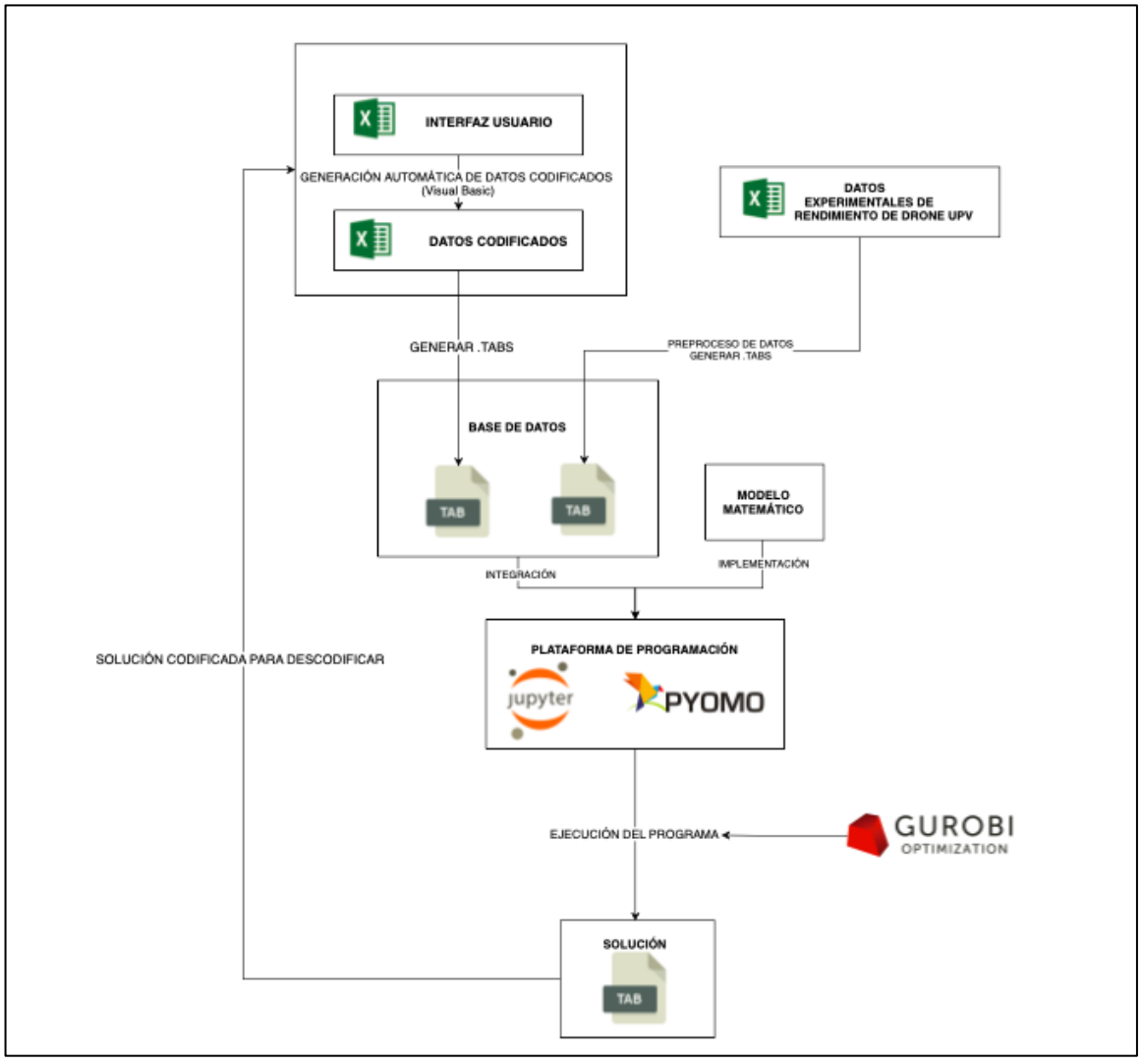

Figura 8. Esquema arquitectura de la herramienta. Fuente: Elaboración propia

A continuación, se definen las principales características y funcionalidades de los componentes de la herramienta, así como una breve justificación de su elección:

- Pyomo 6.4.0. Pyomo es un paquete de software de código abierto basado en Python que admite un conjunto diverso de capacidades de optimización para formular, resolver y analizar modelos de optimización (Hart, 2017). Como se ha explicado anteriormente, se elige Pyomo por las ventajas que presenta como el lenguaje de programación en el que está basado, ya que es uno de los más utilizados por las grandes multinacionales.
- Jupyterlab. Para poder utilizar el paquete de optimización Pyomo es necesario un entorno de desarrollo integrado o IDE. Jupyterlab es compatible con este software y además es gratuito. Es un entorno de programación realmente intuitivo y cómodo para desarrollar una herramienta tan robusta como la que se pretende conseguir. También presenta una gran facilidad en su accesibilidad, ya que se encuentra en la nube y no es necesaria una instalación previa (Jupyter, 2022).
- Gurobi 9.0. Pyomo permite la implementación de los modelos de programación matemática pero necesita llamar a un optimizador para que los resuelva. Gurobi es un Solver de optimización matemática compatible con Phyton, capaz de resolver problemas de PLEM con una gran eficiencia. Además, dispone de una licencia gratuita y no limitada para estudiantes y académicos (uso no comercial).
- Archivos .Tab. Se ha definido un modelo abstracto, es decir, los datos no se encuentran integrados en el propio modelo. La ventaja del modelo abstracto es que se puede utilizar el mismo modelo para diferentes instancias de datos. Esto hace necesario la lectura de datos desde bases de datos externas. Se ha escogido utilizar ficheros .Tab como bases de datos de todos los índices y parámetros del modelo y también se emplean para recopilar los resultados tanto a nivel de objetivo como a nivel de variables de decisión.
- Microsoft Excel. El equipo de Drone UPV nos ha facilitado los datos experimentales en varias hojas de Excel. A través de la licencia ofrecida por la universidad se puede hacer uso completo de todas sus funcionalidades, convirtiéndola en una herramienta potente con una fácil manipulación de la información para la construcción de los archivos .Tab. También, se han realizado ciertas comprobaciones manuales en el apartado de validación para determinadas configuraciones y escenarios mediante el uso de filtros de columnas. Por otro lado, se ha desarrollo una interfaz en Excel para la recopilación de las preferencias del piloto, creando una herramienta interactiva que permite la recopilación de información y su transformación automática a los índices que se emplean para la denominación de cada componente que configura el dron. De esta forma se diseña una herramienta robusta e intuitiva para el usuario que la maneje.

5.3. Base de Datos

La base de datos es una parte elemental para el correcto funcionamiento de la herramienta de programación, ya que es lo que alimenta al programa a través de los datos que se le proporciona. Esta consiste en una colección de información que se relaciona entre sí, creando un conjunto de datos para su posterior gestión mediante un sistema.

Existen dos tipos de datos de entrada: datos relacionados con los componentes y su rendimiento, y datos proporcionados por los pilotos sobre sus preferencias. Los datos de entrada relativos al rendimiento de los componentes han sido facilitados por el equipo de Drone UPV. Estos datos parten de una serie de experimentos para medir los parámetros referidos al rendimiento del dron, probando con todas las configuraciones posibles entre los componentes estudiados. Esta

información se recopiló en un Libro de Microsoft Excel con todos los datos de medición realizados. En cada Hoja de dicho Libro se almacena la información relativa a un motor concreto, y los valores de cada parámetro en función de la hélice empleada. Esta información debe ser estructurada y codificada para que pueda ser incluida en la herramienta desarrollada en el TFG.

Por otro lado, los datos que contienen la información de la maniobrabilidad y agilidad, al ser párametros que no se pueden medir de forma cuantitava mediante algún instrumento de medida, ya que son caracteristicas cualitativas que se basan en la experiencia del piloto y la estabilidad del airframe, se establecen unos criterios para la asignación de estos valores. De este modo se coloca una escala del uno al cinco siendo el valor uno el relativo a la maniobrabilidad o agilidad más baja y, cinco el valor asignado a las maniobrabilidades o agilidades más altas (Tabla 2).

Tabla 2. Relación de los niveles con sus valores cuantitativos. Fuente: Elaboración propia

NIVEL (Calificativo)	VALOR (Cuantitativo)
Muy bajo	1
Bajo	2
Medio	3
Alto	4
Muy alto	5

Por otra parte, para recopilar la información relativa a las preferencias de los pilotos, se ha desarrollado una interfaz interactiva que permite a los pilotos seleccionar de entre las posibles características de los drones, aquellas que sean más adecuadas para su forma de vuelo. Asimismo, les permite la selección de los objetivos a optimizar y el peso que desean asignar a cada uno de ellos. Esta interfaz ha sido creada en Microsoft Excel y mediante el uso del lenguaje de programación Visual Basic.

Así, el piloto accede a la interfaz donde selecciona en las casillas blancas (debajo de las casillas negras que especifican la unidad de medida) entre las opciones posibles del desplegable tanto para definir las características que desean del dron como el rendimiento o coste que desee potenciar o disminuir. Cabe destacar que esta hoja lleva incorporado un botón que elimina instantáneamente toda la información existente en las casillas que puede modificar el piloto, de esta forma se puede realizar una nueva búsqueda rápidamente sin necesidad de borrar manualmente toda esta información. Esta hoja pretende ser lo más intuitiva posible, convirtiendo a la herramienta en un sistema interactivo y flexible en la cual se pueden realizar instancias según las necesidades actuales y venideras del usuario (Figura 9).

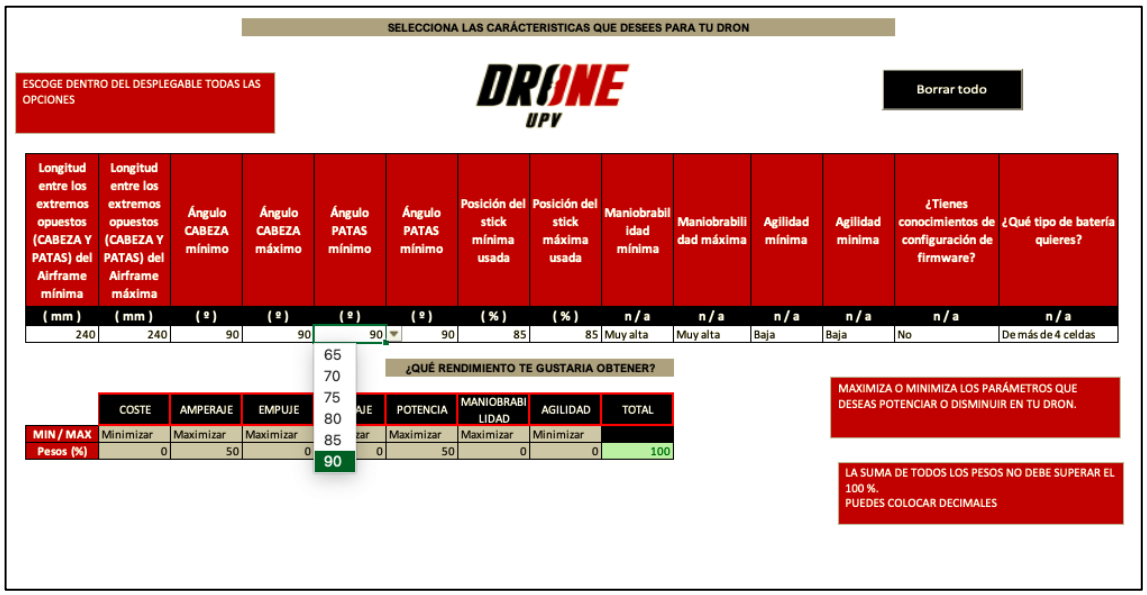

Figura 9. Interfaz Excel mediante (Hoja 1). Fuente: Elaboración propia

La segunda hoja (Figura 10) se encarga de transcribir toda la información que ha introducido el piloto, a los índices asignados para cada componente. Además de codificar los parámetros, también se codifican los pesos que ha colocado el piloto para cada objetivo, identificando cada peso en su respectiva celda. Se han desarrollado dos botones para cada tipo de transcripción, una para la transcripción de los parámetros y otra para la transcripción de los pesos ANEXO I. Esta segunda hoja Excel pretende facilitar el uso de la herramienta a los usuarios que no entienden de programación, facilitando la integración de los datos en los .Tabs correspondientes de cada parámetro definido en el modelo en Pyomo.

Figura 10. Hoja de transcripción automática de los parámetros elegidos por el piloto (Hoja 2). Fuente: Elaboración propia

Todos estos datos se han estructurado para la construcción de los archivos .Tab. Estos archivos son los que contienen los datos de entrada al programa para ser leídos al compilar el código, por tanto deben tener una composición estandarizada y estructurada para su correcta lectura. Estos archivos se pueden generar desde el mismo entorno de programación Jupyterlab. Cada archivo .Tab lleva asignado un nombre diferente relacionado con los datos que contiene. Por ejemplo, para el parámetro de agilidad se crea un archivo .Tab con el nombre “Param_ag.tab” dentro de este se colocan los índices que afectan al valor de este parámetro y su valor. A continuación, se muestra una imagen del archivo .Tab creado para la agilidad (Figura 11).

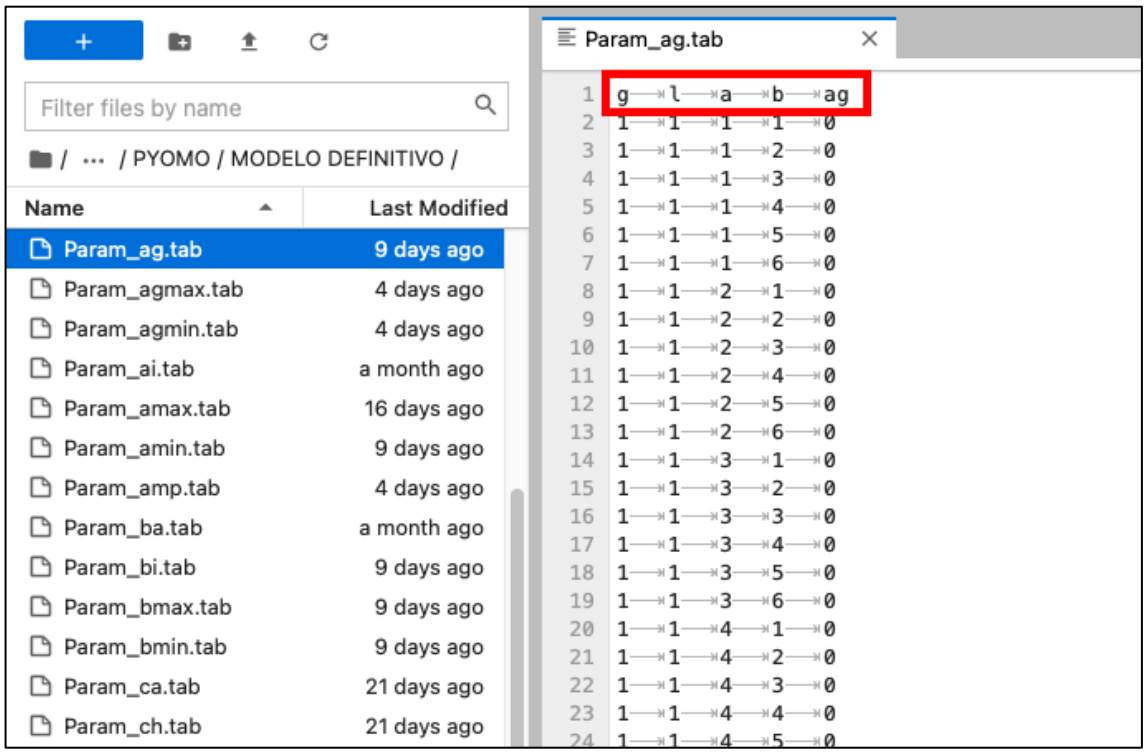

Figura 11. Base de datos. Fuente: Elaboración propia.

5.4. Fichero Pyomo

Una vez se ha formulado el modelo se implementa en el fichero en un fichero .py, para mantener la organización del modelo y el correcto funcionamiento se sigue una construcción del programa similar al modelo matemático. A continuación se explica la estructura:

1) Importación del paquete de optimización: en primer lugar se debe realizar la llamada a Pyomo con la siguiente instrucción:

from pyomo.environ **import ***

2) Definición del modelo: se le atribuye un nombre al modelo y se especifica el tipo de modelo: concreto o abstracto. Los modelos concretos incluyen los datos en la propia formulación del modelo y se define con la siguiente instrucción:

Nombre_del modelo = ConcreteModel*(`Nombre_del_fichero´)*

o puede ser abstracto como es el caso de este, donde los datos se declaran aparte. Estos tipos de modelo tienen una ventaja que es que puedes ejecutar el mismo modelo con muchas instancias de datos diferentes simplemente cambiando los ficheros de datos.

Nombre_del modelo = AbstractModel*(`Nombre_del_fichero´)*

3) Transcripción de nomenclatura: se da paso a la declaración de los índices, parámetros y variables. Si definimos el Nombre_del_modelo con una letra, por ejemplo, la z, se muestra la instrucción para cada elemento:

   a. Para la definición de los índices (Figura 12), se utiliza la expresión “Set()”.

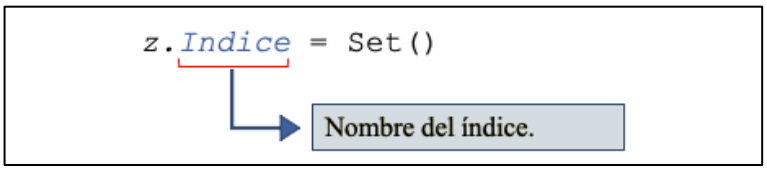

Figura 12. Definición de los índices en Pyomo. Fuente: elaboración propia basado en (Esteso Álvares, 2018)

b. En los parámetros se utiliza la instrucción "Param()". Además, en caso de que el parámetro dependa de uno o más índices, se debe indicar entre paréntesis, colocando una coma entre cada uno para separarlos (Figura 13).

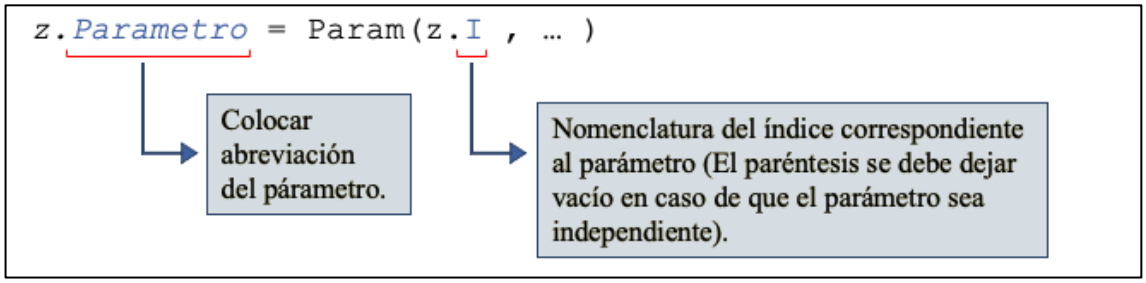

Figura 13. Definición de los parámetros en Pyomo. Fuente: Elaboración propia basado en (Esteso Álvares, 2018)

c. Para las variables se utiliza la instrucción " Var()" , en caso de que la variable dependa de uno o más índices, estos se deben indicar dentro del paréntesis, colocando una coma entre cada uno para separarlos. Además es necesario indicar el tipo de variable, pudiendo ser binaria, entera o continua. **Por defecto se definen como continuas,** por lo tanto para las continuas no hace falta especificar el tipo de variable. Para indicarlo se utiliza la instrucción "domain = Bolean or Intenger". En caso de ser continuas o entras se debe especificar los bounds, es decir, entre que rango de valores se mueve la variable.

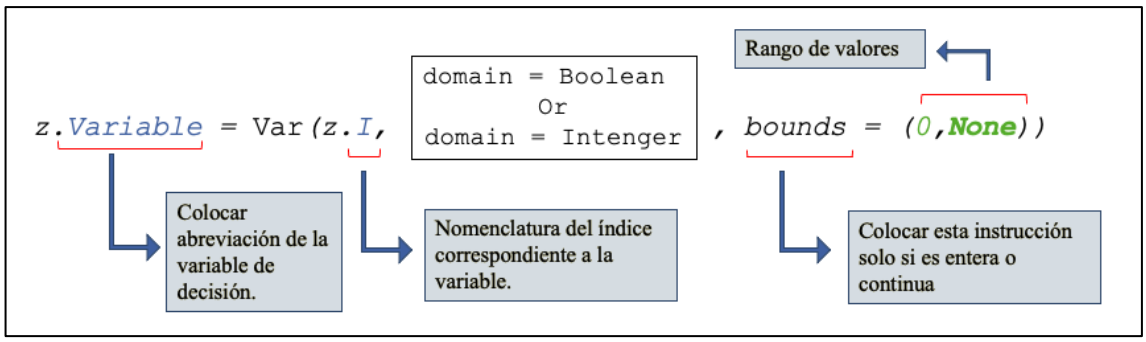

Figura 14. Definición de variables en Pyomo. Fuente: Elaboración propia basado en (Esteso Álvares, 2018)

4) Transcripción de expresiones y función objetivo: seguidamente se definen las expresiones matemáticas que se han definido en el modelo y que posteriormente se integran en una única función objetivo. En la función objetivo se debe indicar si se maximiza o minimiza la función.

a) La instrucción para definir una expresión se muestra en la Figura 15. Se explica detalladamente todos los elementos que se deben definir, las palabras que no tienen una flecha son la estructura de la instrucción y por lo tanto no se deben modificar, exceptuando en la última línea el "z.Exp" la letra detrás del punto varía según se haya nombrado el modelo:

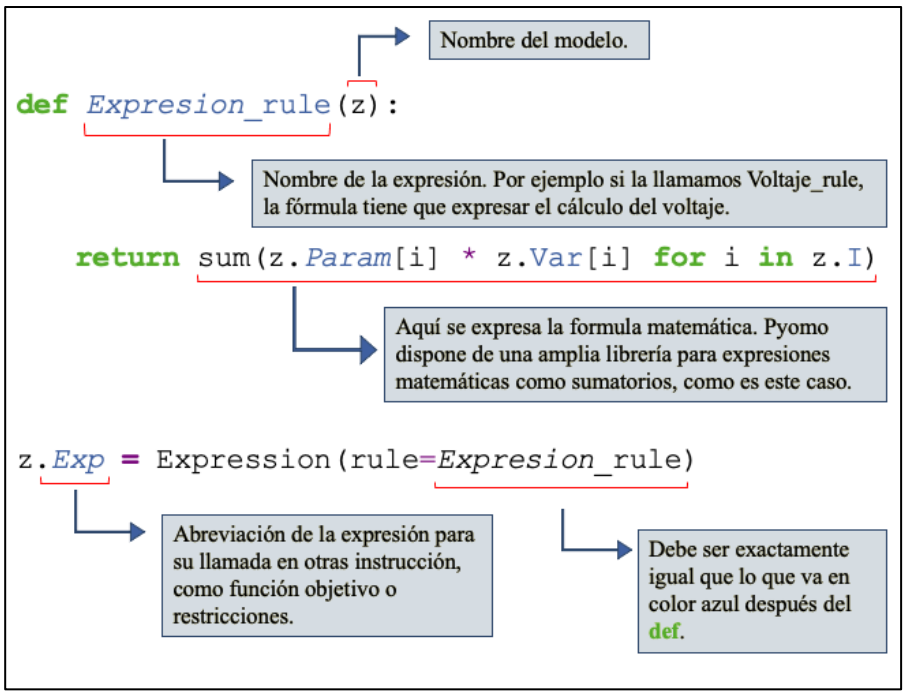

Figura 15. Definición de una expresión en Pyomo. Fuente: Elaboración propia basado en (Esteso Álvares, 2018)

b) La instrucción para definir la función objetivo se muestra en la Figura 16. En esta instrucción deben aparecer las llamadas a las expresiones que se han definido previamente, además de indicar si se desea maximizar o minimizar.

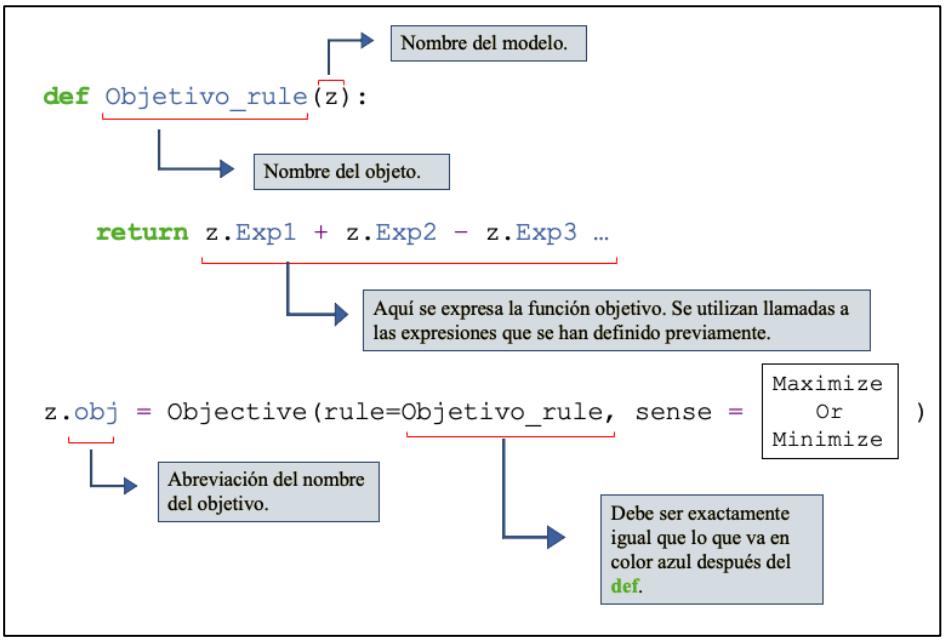

Figura 16. Definición de la función objetivo en Pyomo. Fuente: Elaboración propia basado en (Esteso Álvares, 2018)

5) Transcripción de las restricciones: se prosigue con la definición de las expresiones matemáticas que constituyen las restricciones que se han construido en base a las asunciones. Cabe destacar que Pyomo dispone de una gran librería que permite la definición de expresiones relativamente robustas que se desean transcribir desde el modelo al lenguaje del software. La definición de la formula, en el apartado "return", se suele dividir en dos partes, la parte izquierda (LHS) y la parte derecha (RHS) estas dos partes se separan mediante un símbolo matemático (igual, menor o igual que , mayor o igual que) (Figura 17).

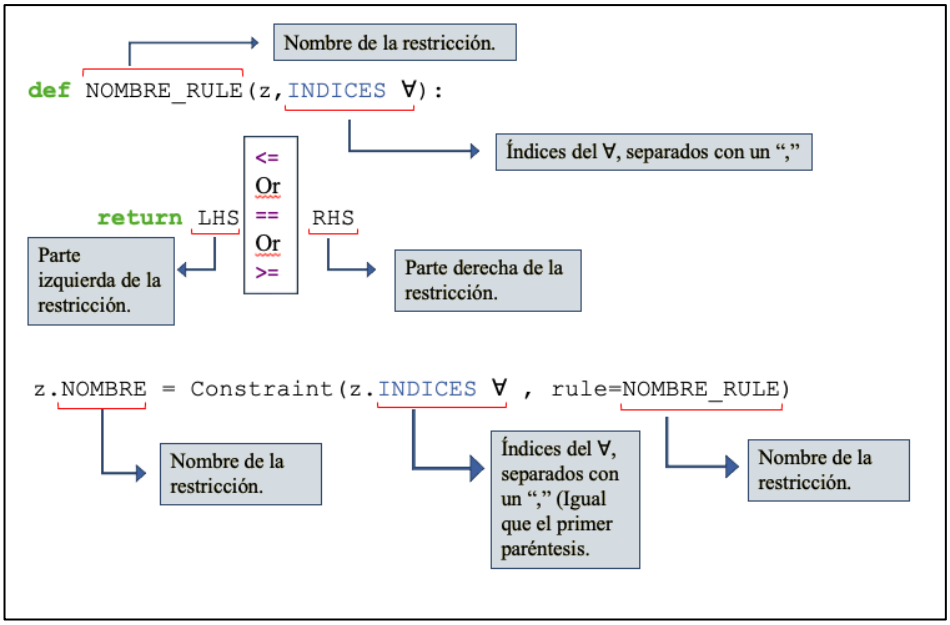

Figura 17. Definición de restricciones en Pyomo. Fuente: Elaboración propia basado en (Esteso Álvares, 2018)

6) Lectura de datos de entrada: al tratarse de un modelo abstracto de debe haber una lectura de ficheros los cuales contienen los datos de entrada al programa, en este caso en .tab. En primer lugar, se debe declarar el portal de datos con la instrucción " DataPortal()". La parte izquierda del igual es el nombre ejemplo "dp":

dp = DataPortal()

Posteriormente se introduce la instrucción de lectura de dato en particular, para cada índice y parámetro mediante la instrucción "Nombre_del_portal_.load()"

a) Para índices (Figura 18): Se coloca la instrucción ".load()" especificando el nombre del archivo y el índice al que se le tiene que atribuir esta información.

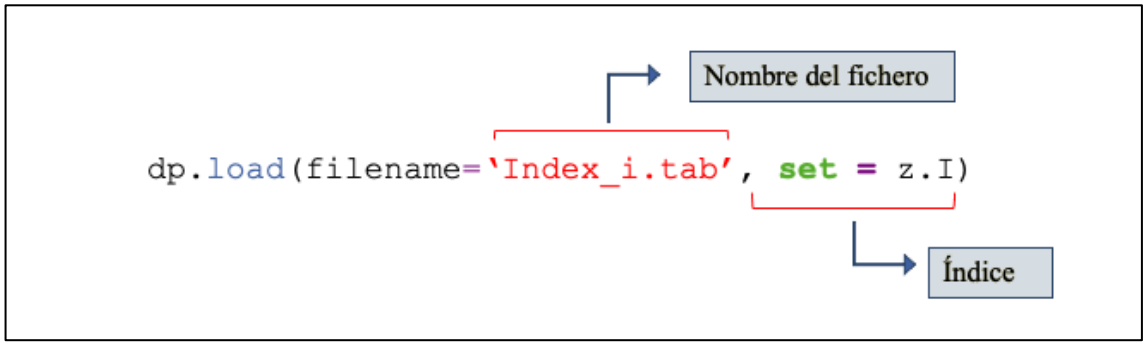

Figura 18. Instrucción de lectura de índices en Pyomo. Fuente: Elaboración propia basado en (Esteso Álvares, 2018)

b) Para parámetros (Figura 19). Se coloca la instrucción ".load()" especificando el nombre del archivo, el parámetro al que se le tiene que atribuir esta información y los índices de los que depende el parámetro.

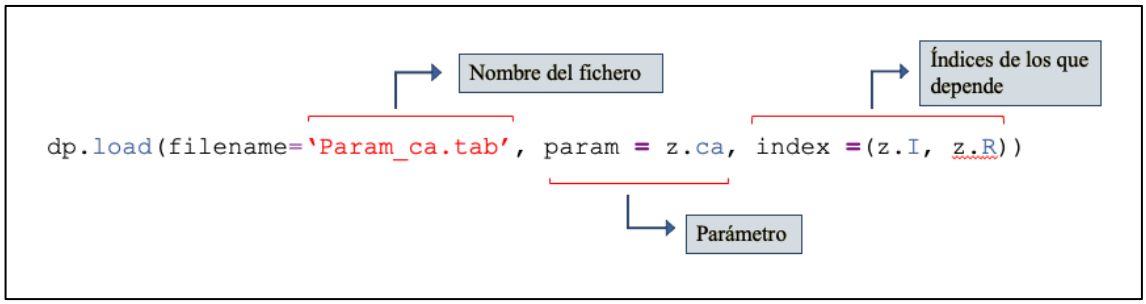

Figura 19.Instrucción lectura de parámetros en Pyomo. Fuente: Elaboración propia basado en (Esteso Álvares, 2018)

7) Resolución del modelo: se declara la instrucción para ejecutar el modelo especificando el programa Solver que va a solucionar el modelo, en este caso Gurobi.

a) En primer lugar, para crear la instancia con los datos que se han leído previamente:

```
inst = z.create_instance(dp)
```

b) Esta instrucción es opcional y se utiliza para que se imprima la ejecución del modelo en la consola. Poder comprobar la explosión del modelo en la consola permite comprobar donde pueden encontrarse los posibles errores en la formulación del modelo.:

```
inst.pprint()
```

c) Para llamar al Solver que a su vez llama a Gurobi, para resolver el modelo:

```
opt = SolverFactory('gurobi')
```

d) Ejecutar los resultados de la instancia: Aquí se resuelve con el Solver para la instancia de datos que has definido previamente.

```
results = opt.solve(inst)
```

8) Exportar/Imprimir resultados: se llama a una función externa que actúa como mini programa que traslada los datos de las variables de decisión a un fichero .tab exclusivamente para la lectura de los resultados de estas y el valor de la función objetivo.

a) Para imprimir los resultados en la consola:

```
inst.solutions.store_to(results)

print(results)
```

b) Para exportar los resultados en otro archivo .tab se hace una llamada a otro programa pequeño denominado "pyoutil", este programa ha sido creado por el Centro de Investigación en Gestión e Ingeniería de Producción. Su función es exportar los resultados de las variables de decisión y el valor de la función objetivo, creando unos archivos .tab con el nombre de la variable de forma automática. La utilización del programa permite una lectura más rápida y clara de la solución.

```
import pyoutil as pu
```

c) Ejecutar el modelo "pyoutil" al finalizar el código:

```
pu.save_results_to_file(inst, results)
```

Una vez explicado el desarrollo del del fichero en Pyomo, se hacen algunas aclaración en relación con la notación del lenguaje, como:

- Se utiliza "." para la separación de decimales en los números.
- Para la definición de cada elemento se coloca siempre el nombre del modelo y un punto delante del elemento que se va a definir. Ejemplo: "z.Voltaje".
- Es importante definir cada parámetro o variable con la instrucción exacta, con las mayúsculas y minúsculas exactamente iguales a la explicación, ya que una minúscula o mayúscula mal puesta genera un error en la ejecución del programa.

El código que se ha desarrollado en base al modelo de PLEM propuesto se encuentra en el ANEXO II.

5.5. Conclusiones

Se ha propuesto una herramienta con una arquitectura basada en un entorno de programación en Jupyterlab, mediante un lenguaje de programación basado en Python, que pretende automatizar la toma de decisión en la selección de los componentes de un dron aportando una solución óptima basada en un modelo de PLEM.

Una vez se ha programado la herramienta, se han insertado los datos correctamente y se han definido todos los elementos que componen el sistema, se da paso a la validación de este, para comprobar que el funcionamiento de la herramienta es correcto y aporta soluciones verdaderamente óptimas.

6. VALIDACIÓN Y APLICACIÓN A CASO REAL

6.1. Introducción

En este capítulo se exponen los datos de entrada que se emplean tanto para la validación como para la aplicación al caso real. Luego se define la metodología de validación y se lleva a cabo es validación mediante escenarios propuestos y por último se aplica la herramienta a casos reales.

6.2. Datos de Entrada

El modelo PLEM diseñado permite la selección de tres componentes del dron de carreras: el airframe, los motores y las hélices. A su vez se consideran componentes de diferentes marcas y que cuentan con diferentes características. Cabe resaltar que únicamente se han considerado componentes sobre los que el grupo Drone UPV ha realizado estudios, sabiendo su comportamiento en vuelo.

A continuación, se especifican todos los tipos de cada componente con su respectivo índice y características, así como los precios de cada uno:

I. **Airframe.** El tipo de airframe puede ser **simétrico, no simétrico o hibrido**. Dentro de cada tipo se diferencian por su longitud entre los dos motores opuestos y por los ángulos entre sus bazos y patas. La Tabla 3 muestra las combinaciones de características consideradas en este TFG.

Tabla 3. Características del Airframe. Fuente: Elaboración propia

GEOMETRÍA	*g*	LONGITUD (mm)	ÁNGULOS ENTRE BRAZOS (º)	ÁNGULOS ENTRE PATAS (º)	OBSERVACIONES
Simétrico	1	210, 220, 230, 240, 250	90	90	El ángulo de los brazos tiene que ser igual al ángulo de las patas
No simétrico	2	210, 220, 230	65, 70, 75, 80	65, 70, 75, 80	El ángulo de los brazos tiene que ser igual al ángulo de las patas
Hibrido	3	220, 230, 240, 250	65	70, 75, 80, 85, 90	El ángulo de los brazos solo puede ser igual a 65

Para todos los tipos de airframes se ha considerado el mismo precio ya que este solo varía según su material y no según las características consideradas en el modelo propuesto. El precio que se ha tomado como referencia según varias búsquedas de airframes es de 127,88 €. Dado que el precio es independiente de las características consideradas en el modelo, este valor será constante en la función objetivo y por tanto podría no ser considerado en el modelo. No obstante, se ha modelado este aspecto por si en el futuro hubiera una variación de los precios en función de las características de los airframes.

La Tabla 4 establece la relación entre la longitud entre los ejes opuestos del airframe y el índice l empleado, y la relación entre los ángulos entre los brazos (a) y las patas (b) del airframe y el índice a y b empleado para identificarlos.

Tabla 4. Relación de índices relativos a la longitud y ángulos del airframe. Fuente: Elaboración propia

l	Longitud entre ejes opuestos
1	210 mm
2	220 mm
3	230 mm
4	240 mm
5	250 mm

a / b	Ángulos entre brazos (a) y patas del airframe
1	65º
2	70º
3	75º
4	80º
5	85º
6	90º

II. **Motor.** En el TFG se va a considerar siete tipos de motores, cuyos precios se muestran en la Tabla 5. El nombre del motor se refiere a las siglas del motor y el número de revoluciones por minuto que es capaz de girar con un voltio.

Tabla 5. Tipos de motores y precios. Fuente: Elaboración propia

MOTOR	m	PRECIO	cm
BB1950 KV	1	32,44 €/ud	1
BB2800 KV	2	32,44 €/ud	2
MCKV3 1910 KV	3	38,51 €/ud	3
MCKV3 2100 KV	4	38,51 €/ud	4
MCKV2 1950 KV	5	32,25 €/ud	5
MCKV2 2250 KV	6	32,25 €/ud	6
MCKV2 2550 KV	7	32,25 €/ud	7

III. **Hélices.** En el TFG se va a considerar cinco hélices diferentes que se diferencian en cuanto a la longitud de sus aspas (Tabla 6).

Tabla 6. Tipos de hélices y precios. Fuente: Elaboración propia

TIPO DE HÉLICE	h	PRECIO	ch
T4943	1	0.87 €/ud	1
T5143s	2	1.03 €/ud	2
T5146	3	0.46 €/ud	3
T5147	4	0.60 €/ud	4
T5150	5	1.04 €/ud	5

Por otro lado, otro dato de entrada que se tiene en cuenta es la posición del stick izquierdo del mando que controla el dron. Dependiendo de la posición en que se encuentre, ciertos valores, de los parámetros analizados, varían. En general estos valores aumentan cuanto más se inclina hacia arriba este stick. Para dimensionar las posiciones, se toma un intervalo en porcentaje siendo el 0% la posición en la que el stick se encuentra lo más abajo posible y el 100% lo más arriba posible en el eje permitido por la palanca (Figura 20).

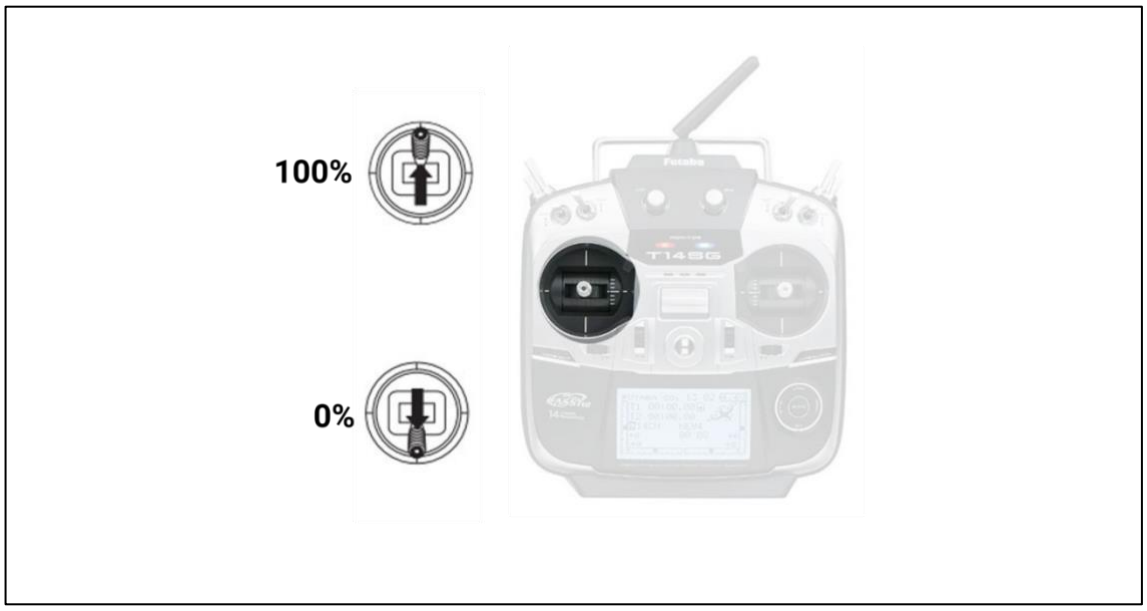

Figura 20. Posiciones mínima y máxima de la palanca izquierda. Fuente: Elaboración propia

Solo se consideran las posiciones a partir de la 50%, ya que no se suelen volar a menos de eso. Los datos proporcionados por Drone UPV solamente consideran las posiciones del 50% al 100% ya que a menos de esta posición el dron no puede despegar, los datos se han tomado de cinco en cinco. Se define la posición del stick con el índice s, siendo el valor uno la posición 50%, el valor dos la posición 55% y así hasta el valor 11, siendo este el valor para la posición 100%.

Como datos de entrada se consideran también el rendimiento del dron en lo que respecta a empuje, voltaje, amperaje, potencia, maniobrabilidad y potencia. Los valores de cada parámetro no se pueden mostrar por tema de confidencialidad de datos.

Por último, con respecto a las preferencias de los pilotos, estas varían en función del decisor. Por tanto, con cada piloto se deberá ejecutar el modelo con sus preferencias. Consiguiendo una traducción automática de lo que el piloto ha solicitado a través del Excel interactivo que se ha preparado y presentado en la sección 5.3. Estos datos de preferencias se irán detallando en las próximas secciones ya que varían en cada ejecución del modelo.

El piloto también tiene control en dos de los parámetros de rendimiento que son maniobrabilidad y agilidad. Pueden escoger un mínimo y una máximo de ambos, según las habilidades, las capacidades de conducción del dron y el circuito.

6.3. Metodología de Validación

Para poder afirmar que la herramienta cumple con todas las características explicadas a lo largo del TFG, se deben realizar una serie de pruebas que verifiquen que las soluciones que proporciona cumplen con los requisitos establecidos por el modelo.

Cabe destacar que la metodología de validación pertenece a la fase o etapa cuatro, tras haber implementado el modelo en la herramienta y haber depurado el código correctamente para que no haya errores de programación. En la Figura 21 se muestra un diagrama de flujo que explica los pasos de la metodología utilizada.

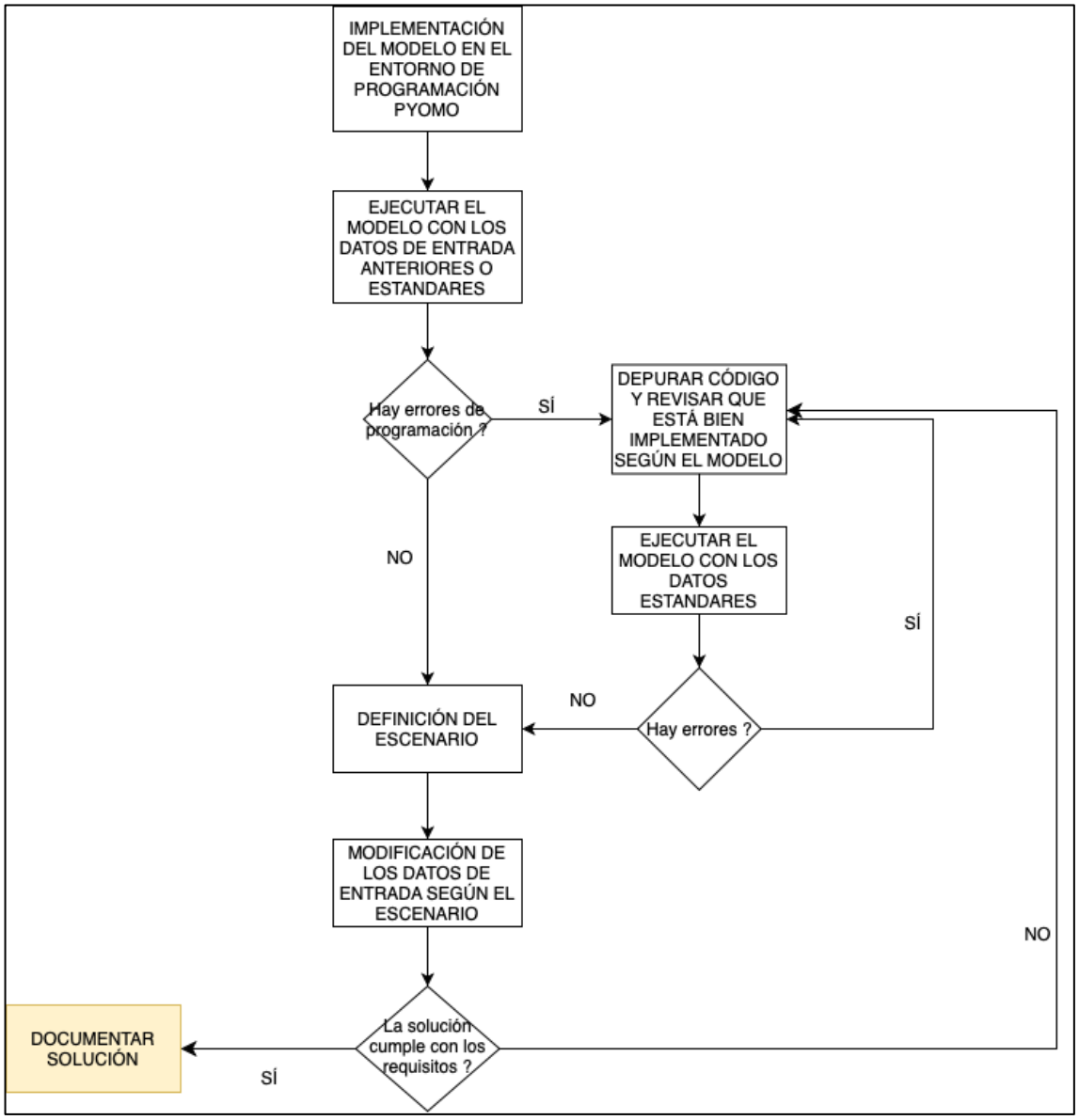

Figura 21: Metodología de validación. Fuente: Elaboración propia

Para comprobar que la herramienta funciona correctamente, se deben proponer escenarios que conlleven una solución esperada, es por ello por lo que se ejecuta el modelo para dos tipos de escenarios:

- Escenarios en los que se optimiza **un solo objetivo** para comprobar que la herramienta proporciona soluciones que optimizan cada uno de los objetivos.
- Escenarios que representen **situaciones conocidas,** estos escenarios dan lugar a soluciones forzadas en las que se conoce la solución o soluciones óptimas para comprobar que la herramienta es fiable.

6.3.1. Validación de objetivos

En la primera tipología de escenarios se ejecuta el modelo nueve veces, asignando en cada escenario todo el peso a un objetivo. En este primer escenario se va a considerar que las preferencias del piloto no son restrictivas, es decir, en los parámetros que limitan las posibles soluciones, según las preferencias del usuario, se van a colocar valores mínimos para sus parámetros que limitan la cota inferior y valores máximos para los valores que limitan la cota superior (Figura 22). De esta manera el modelo considerará todas las posibilidades siendo restringido únicamente por las restricciones de la incompatibilidad de las combinaciones entre componentes.

lmin	lmax	amin	amax	bmin	bmax	smin
1	5	1	6	1	6	1
smax	**mamin**	**mamax**	**agmin**	**agmax**	**pc**	**pb**
11	1	5	1	5	1	1

Figura 22. Parámetros de preferencias del piloto. Fuente: Elaboración propia

En la Tabla 7 se define el número del escenario, el objetivo optimizado, las configuraciones esperadas que cumplen con el objetivo habiendo analizado previamente de forma manual los datos y el valor esperado para la función objetivo optimizada.

Tabla 7. Escenarios para la validación de la herramienta. Fuente: Elaboración propia

ESCENARIO	OBJETIVO	CONFIGURACIÓN ESPERADA	VALOR ESPERADO F.O
1	Z1: Minimización de costes	Elección del motor y hélice más barato, ya que para los datos de entrada que se han considerado el coste del airframe es fijo. [motor, hélice]: [5,3], [6,3], [7,3]	258,71 €
2	Z2: Maximización del amperaje	Elección del motor y hélice con mayor amperaje. [motor, hélice]: [6,5]	324,03 A
3	Z2: Minimización del amperaje	Elección del motor y hélice con menor empuje. [motor, hélice]: [6,3]	198,52 A
4	Z3: Maximización del empuje	Elección del motor y hélice con mayor empuje. [motor, hélice]: [5,3]	14561,1 g
5	Z3: Minimización del empuje	Elección del motor y hélice con menor empuje. [motor, hélice]: [2,1]	8130,76 g
6	Z4: Maximización del voltaje	Elección del motor y hélice con mayor voltaje. [motor, hélice]: [5,3]	369,87 V
7	Z4: Minimización del voltaje	Elección del motor y hélice con menor voltaje. [motor, hélice]: [2,5]	164,22 V
8	Z5: Maximización de la potencia	Elección del motor y hélice con mayor potencia. [motor, hélice]: [5,4]	6448,75 W

Tabla 8. Escenarios para la validación de la herramienta (continuación). Fuente: Elaboración propia

ESCENARIO	OBJETIVO	CONFIGURACIÓN ESPERADA	VALOR ESPERADO F.O
9	Z5: Minimización de la potencia	Elección del motor y hélice con menor potencia. [motor, hélice]: [6,3]	3122,5 W
10	Z6: Minimización de la maniobrabilidad	Elección del airframe con menor maniobrabilidad. [g,l,a,b]: [2,1,1,1], [2,1,2,2], [2,1,3,3], [2,1,4,4]	1
11	Z7: Maximización de la maniobrabilidad	Elección del airframe con mayor maniobrabilidad. [g,l,a,b]: [1,5,6,6]	5
12	Z8: Minimización de la agilidad	Elección del airframe con menor agilidad. [g,l,a,b]: [3,2,1,2], [3,3,1,2], [3,4,1,2], [3,5,1,2]	1
13	Z9: Maximización de la agilidad	Elección del airframe con mayor agilidad. [g,l,a,b]:[2,1,4,4], [2,2,4,4], [2,3,4,4]	5

La Tabla 13 muestra los resultados obtenidos tras ejecutar los escenarios con un solo objetivo optimizado. En esta tabla se puede comparar la configuración y el resultado, de la función objetivo, esperado que se obtiene de cada escenario.

Tabla 9. Ejecución de los resultados de la herramienta según los escenarios. Fuente: Elaboración propia

ESCENARIO	CONFIGURACIÓN [g,l,a,b,m,h]		FUNCIÓN OBJETIVO	
	ESPERADA	OBTENIDA	ESPERADA	OBTENIDA
1	[g,l,a,b,5,3] [g,l,a,b,6,3] **[g,l,a,b,7,3]**	[1,4,6,6,7,3]	258,71 €	258,71 €
2	**[g,l,a,b,6,5]**	[3,5,1,6,6,5]	324.03 A	324.03 A
3	**[g,l,a,b,6,3]**	[3,5,1,6,6,3]	198,52 A	198,52 A
4	**[g,l,a,b,5,3]**	[1,3,6,6,5,3]	14561,1 g	14561.1 g
5	**[g,l,a,b,6,3]**	[3,5,1,5,2,1]	8130.76 g	8130.76 g
6	**[g,l,a,b,5,3]**	[3,5,1,2,5,3]	369.87 V	369,87 V
7	**[g,l,a,b,2,5]**	[3,5,1,6,2,5]	164.22 V	164.22 V
8	**[g,l,a,b,5,4]**	[1,4,6,6,5,4]	6448.75 W	6448.75 W
9	**[g,l,a,b,6,3]**	[3,5,1,6,6,3]	3122.5 W	3122.5 W
10	[2,1,1,1,m,h] **[2,1,2,2,m,h]** [2,1,3,3,m,h] [2,1,4,4,m,h]	[2,1,2,2,7,4]	1	1
11	**[1,5,6,6,m,h]**	[1,5,6,6,1,1]	5	5
12	[3,2,1,2,m,h] [3,3,1,2,m,h] [3,4,1,2,m,h] **[3,5,1,2,m,h]**	[3,5,1,2,1,1]	1	1
13	[1,5,6,6,m,h] [2,1,4,4,m,h] [2,2,4,4,m,h] **[2,3,4,4,m,h]**	[2,3,4,4,1,1]	5	5

Como se ha podido comprobar, la herramienta efectivamente escoge las configuraciones que se esperaban y optimiza los objetivos en cuestión de segundos.

6.3.2. Validación por comparación con casos conocidos

En segundo lugar, se ejecutan los escenarios con situaciones conocidas. En este tipo de escenarios se plantean casos conocidos para comprobar que, aplicando distintos parámetros y según las preferencias del piloto, la herramienta funciona correctamente y aporta soluciones factibles. Para todos los escenarios se considera el objetivo de **minimizar costes.** En la Tabla 10 se describe cada escenario, la solución esperada y la solución obtenida.

Se proponen cuatro escenarios conocidos, que inducen a una respuesta determinada.

Tabla 10. Validación de escenarios conocidos. Fuente: Elaboración propia

ESCENARIOS	DESCRIPCIÓN	ESPERADA	OBTENIDA
1	El piloto desea una **maniobrabilidad mínima de 2,5 y máxima de 3** , **ángulo mínimo de 70º** tanto para la **cabeza** como para las **patas** y **agilidad máxima de 3**. Además quiere un motor con **batería de más de cuatro celdas.**	[1,1,6,6,1,h] [1,1,6,6,3,h] [1,1,6,6,4,h] **[1,1,6,6,5,h]** [1,1,6,6,6,h]	[1,1,6,6,5,3]
2	El piloto **no** tiene **conocimientos** suficientes para modificar el controlador del firmware pero desea una **longitud mínima de 230 mm y una agilidad máxima de 4.**	**[1,3,6,6,m,h]**	[1,3,6,6,5,3]
3	El piloto desea una **agilidad igual a 5** y un **ángulo mínimo de 85º**. Además quiere un **motor con batería de más de cuatro celdas.**	[1,5,6,6,1,h] [1,5,6,6,3,h] [1,5,6,6,4,h] **[1,5,6,6,5,h]** [1,5,6,6,6,h]	[1,5,6,6,5,3]
4	Le pedimos a la herramienta un ángulo para la cabeza inferior a 80º y se coloca que el piloto no tiene la capacidad de hacer cambios en el controlador, por lo que tendrá que escoger un airframe con una geometría simétrica. Con este tipo de geometría estrictamente se tiene que cumplir que los ángulos de la cabeza y las patas sean iguales y de 90º.	**Infactibilidad**	Termination message: Model was proven to be infeasible.

Se puede apreciar que la herramienta se comporta tal y como lo esperado en cada uno de los escenarios.

6.4. Aplicación Herramienta a Casos Reales

En este apartado se van a introducir los datos basados en la elección de preferencias de cuatro pilotos, mediante la hoja Excel interactiva se coge esta información y se traduce a los datos de entrada para la ejecución del modelo.

Los pilotos pueden seleccionar la longitud, ángulos, maniobrabilidad y agilidad mínima y máxima según sus preferencias. Además, el modelo considera únicamente las posiciones del stick que generalmente utiliza el piloto, ya que cada piloto tiene una manera de pilotar, a veces incluso haciendo uso únicamente de ciertas posiciones. Por último, el piloto debe especificar si tiene conocimientos suficientes para realizar cambios de firmware y si quiere utilizar baterías de más de 4 celdas o de 4 celdas (Tabla 11).

Tabla 11. Datos de las preferencias de los pilotos de Drone UPV. Fuente: Elaboración propia

	PILOTO 1	P1	PILOTO 2	P2	PILOTO 3	P3	PILOTO 4	P4
lmin	220	2	240	4	210	1	210	1
lmax	230	3	250	5	230	3	250	5
amin	80	4	65	1	80	4	65	1
amax	90	6	75	3	90	6	90	6
bmin	80	4	65	1	65	1	65	1
bmax	90	6	90	6	75	3	90	6
smin	50	1	50	1	50	1	50	1
smax	100	11	100	11	100	11	100	11
mamin	Baja	2	Media	3	Media	3	Muy baja	1
mamax	Media	3	Muy alta	5	Alta	4	Muy alta	5
agmin	Alta	4	Muy baja	1	Muy baja	1	Muy baja	1
agmax	Muy alta	5	Baja	2	Muy alta	5	Muy alta	5
pc	Sí	1	No	0	No	0	No	0
pb	Más de 4 celdas	0	Más de 4 celdas	0	Más de 4 celdas	0	Más de 4 celdas	0

Además de las preferencia de las características de los componentes y el rendimiento de la maniobrabilidad y agilidad, los pilotos deben especificar sus prioridades en cuanto a rendimiento y costes se refiere. Deben poner en la hoja de preferencias el peso que le darían a los diferentes objetivos existentes (maximizar empuje, minimizar amperaje…).

La interfaz desarrollada en Excel transcribe la información de la hoja de introducción de datos por parte de los pilotos en otra hoja y la codifica en el parámetro determinado para poder migrar los datos correctamente a la herramienta de optimización. Se recoge tanto los parámetros como los pesos especificados por cada piloto (Tabla 12).

Tabla 12. Objetivos y pesos de cada objetivo. Fuente: Elaboración propia basada en la información proporcionada por Drone UPV

Piloto 1			**Piloto 2**		
Objetivo	**W**	**Peso**	**Objetivo**	**W**	**Peso**
• Minimizar amperaje • Maximizar empuje • Maximizar maniobrabilidad • Maximizar agilidad	W2 W5 W11 W13	80% 6,67% 6,67% 6,67%	• Minimizar coste • Maximizar maniobrabilidad • Minimizar agilidad	W1 W11 W12	10% 10% 80%
Piloto 3			**Piloto 4**		
Objetivo	**W**	**Peso**	**Objetivo**	**W**	**Peso**
• Minimizar amperaje • Maximizar voltaje • Minimizar maniobrabilidad • Maximizar agilidad	W2 W7 W10 W13	13,33% 13,33% 13,33% 60%	• Minimizar coste • Maximizar maniobrabilidad	W1 W11	50% 50%

Cabe destacar que el piloto 4 sería el piloto novato que no sabe aún sobre las características que le van mejor (todas las opciones contempladas), no sabe programar en la controladora, y le interesa un coste bajo y maniobrabilidad del dron elevada.

Ejecutando los distintos escenarios de cada piloto se obtienen las siguientes soluciones con su valor de la función objetivo que pretende maximizar su valor, siendo uno su valor máximo (Tabla 13)

Tabla 13. Soluciones de las pruebas de casos reales. Fuente: Elaboración propia

Piloto	Configuración	Componentes reales	Resultado Función Objetivo
1	[2,3,4,4,6,3]	Dron con geometría no simétrica, longitud 230, ángulos de 80º para los brazos y patas, motor MCKV2 2250 y hélice T5146	-0.35
2	[3,5,1,2,6,3]	Dron con geometría hibrida, longitud 250, ángulo de 70º entre sus brazos y 90º entre las patas, motor MCKV2 2250 y hélice T5146	-0.19
3	[1,2,6,6,5,3]	Dron con geometría simétrica, longitud entre extremos opuestos del eje de los motores de 220, ángulos de 90º tanto en los brazos como en las patas, motor MCKV2 1950 y hélice T5146	0.39
4	[1,5,6,6,5,3]	Dron con geometría simétrica, longitud 250, ángulos de 90º entre los brazos y 90º entre las patas motor MCKV2 1950 y hélice T5146	0.05

En el próximo apartado se evalúa la herramienta comparando los resultados aportados con la herramienta y la decisión que escogería Drone UPV para cada piloto sin usar la herramienta desarrollada, utilizando únicamente su intuición y experiencia.

6.5. Evaluación de la Herramienta

Tras haber realizado todas las pruebas con los distintos escenarios propuestos, en este apartado se pretende evaluar la herramienta, para ver el beneficio de su uso frente a seguir seleccionando los componentes por intuición como lo hacía Drone UPV.

En primer lugar, para la evaluación de la eficiencia computacional se debe decir que el ordenador utilizado para la ejecución del modelo mediante Pyomo 6.4.0 y Gurobi 9.0, dentro del entorno de programación de Jupyter lab, tiene un procesador 1.8 GHz Dual-Core Intel Core i5 y con una RAM de 8 GB.

Se va a analizar la eficiencia computacional de la herramienta, analizando el porcentaje de bondad que hace falta para obtener la solución óptima (GAP) y el tiempo de resolución de la herramienta en aportar una solución y en caso de tener 0% de GAP, ser esta la óptima. Se analizan únicamente los escenarios de caso real, ya que son estas las ejecuciones las que nos interesan, al ser las que realmente se van a ejecutar en el uso rutinario para la tarea de selección de componente del dron de carreras que se desea desarrollar (Tabla 14).

Tabla 14. Evaluación del GAP y tiempo de ejecución de los casos reales. Fuente: Elaboración propia

PILOTOS	GAP%	TIEMPO DE RESOLUCIÓN (Wall time + Time)
Datos de entrada: Preferencias del piloto 1	0.0	0,649 seg
Datos de entrada: Preferencias del piloto 2	0.0	0,653 seg
Datos de entrada: Preferencias del piloto 3	0.0	0,612 seg
Datos de entrada: Preferencias del piloto 4	0.0	0,78 seg

Asimismo, se comparan los resultados que se han obtenido del dron seleccionado para los objetivos considerados con la configuración obtenida con el modelo y con la configuración obtenida por Drone UPV. Para cada piloto se propone una tabla con el método utilizado, la configuración seleccionada y los objetivos que se desean de la configuración.

Para cada piloto existía un peso de cada objetivo, es por ello por lo que el resultado más importante es el de la función objetivo ya que los objetivos por individual pueden ser mejores o peores que la selección de la herramienta, pero para comprobar que la herramienta funciona correctamente el valor del conjunto, es decir, el de la función objetivo debe ser igual o mejor que la selección por el método Drone UPV. En las siguientes tablas se representa la comparación entre la opción proporcionada por Drone UPV y la herramienta de optimización según las preferencias de cada uno de los cuatro pilotos, se compara en porcentaje las mejoras que supone la opción de la herramienta.

Para el primer piloto las dos configuraciones obedecen las preferencias del piloto y obedecen lar restricciones por compatibilidad de componentes. Sin embargo, la configuración es diferente, por lo tanto, el rendimiento en cada parámetro es distinto. (

Tabla 15).

Tabla 15. Comparación de la configuración de Drone UPV frente a la herramienta de optimización (Piloto 1). Fuente: Elaboración propia

	Drone UPV	Herramienta TFG	Mejora (%)
Configuración	Geometría no simétrica, longitud 220, ángulo entre brazos y patas de 80º , motor BB1950KV y hélice T5146	Geometría no simétrica, longitud 230, ángulos de 80º entre los brazos y 80º entre las patas, motor MCKV2 2250 y hélice T5146	
Min amperaje	253.98	198.52	17,11%
Max empuje	13158.08	9420.35	-25,66%
Max maniobrabilidad	1.5	2	10%
Max agilidad	5	5	0%
FO	-0.48	-0.35	13%

¿Factible?	SÍ	SI	

Seguidamente, para el piloto dos, este deseaba un dron que priorizara la controlabilidad frente a la rapidez y como se muestra la solución de la herramienta es mejor en todos los parámetros (Tabla 16).

Tabla 16. Comparación de la configuración de Drone UPV frente a la herramienta de optimización (Piloto 2). Fuente: Elaboración propia

	Drone UPV	**Herramienta TFG**	**Mejora (%)**
Configuración	Geometría hibrida, con longitud 250, con ángulo entre brazos de 65 y patas de 90, con motor BB2800KV y hélice T5146	Geometría hibrida, longitud 250, ángulo de 70º para entre los brazos y 90º entre las patas motor MCKV2 2250 y hélice T5146	
Min coste	259,47 €	258.71	0.29%
Max maniobrabilidad	3	3	0%
Min agilidad	3	1	40%
FO	-0.51	-0.19	32%
¿Factible?	NO	SÍ	

El piloto tres, este desea un dron rápido, priorizando la agilidad frente a la maniobrabilidad. Como se puede observar la configuración que recomienda Drone UPV es infactible, a pesar de que la función objetivo es mayos que la herramienta de optimización (Tabla 17).

Tabla 17.Comparación de la configuración de Drone UPV frente a la herramienta de optimización (Piloto 3). Fuente: Elaboración propia

	Drone UPV	**Herramienta TFG**	**Mejora (%)**
Configuración	Geometría simétrica, longitud 250, ángulo entre brazos y patas de 90, con motor BB2800KV y hélice T4943	Geometría simétrica, longitud 220, ángulos de 90º entre los brazos y 90º entre las patas, motor MCKV2 1950 y hélice T5146	
Min amperaje	247.71	305.57	-17,85%
Max voltaje	167.34	369.87	54,75%
Min maniobrabilidad	3	3.5	-10%
Max agilidad	5	3	-40%
FO	0.48	0.39	- 9%
¿Factible?	NO	SÍ	

Por último, el piloto cuatro es el más inexperto de los cuatro, él solamente desea un dron que sea barato y fácil de manejar, sin ninguna preferencia en concreto, pero con la limitación de que no tiene conocimiento de control de firmware. En los resultados de la Tabla 18 se observa que los rendimientos en cada parámetro son idénticos, esto se debe a que la configuración del airframe es la misma, solo cambia el motor y las hélices.

Tabla 18. Comparación de la configuración de Drone UPV frente a la herramienta de optimización (Piloto 4). Fuente: Elaboración propia

	Drone UPV	Herramienta TFG	Mejora (%)
Configuración	Geometría simétrica, con longitud 250, con ángulo entre brazos y patas de 90, con motor BB2800KV y hélice T5150	Geometría simétrica, longitud 250, ángulos de 90º entre los brazos y patas, motor MCKV2 1950 y hélice T5146	
Min coste	258,71	258,71	0%
Max maniobrabilidad	5	5	0%
FO	0.05	0.05	0%
¿Factible?	NO	SÍ	

Como se puede observar en algunos resultados, en alguno de los parámetros (analizándolos individualmente) la recomendación de Drone UPV es mejor, no obstante, según las preferencias que han seleccionado los pilotos dos, tres y cuatro, la opción escogida por Drone UPV no es factible. Esto se debe a que estos pilotos han elegido que el motor del dron tiene que ser recargable con una batería de más de cuatro celdas. Drone UPV recomienda el motor BB2800KV para estos pilotos y este tipo de motor no dispone de una batería de más de cuatro celdas. Sin embargo, la herramienta aporta soluciones óptimas, que cumplen tanto con las preferencias del dron como las limitaciones que presenta la compatibilidad de los componentes.

6.6. Conclusiones

Tras realizar una serie de pruebas mediante una metodología definida, se ha podido comprobar el correcto funcionamiento de la herramienta de optimización para la selección de componentes para drones de carreras.

El trabajo de depuración y corrección de errores ha permitido que el código sea cada vez más robusto, consiguiendo un modelo que optimiza la toma de decisión sobre la selección de los componentes de drones, obteniendo siempre la configuración óptima en segundos a catando las preferencias del piloto y las restricciones presentes.

Por tanto, se da por finalizada la fase de validación con éxito, consiguiendo unas mejoras notables en comparación al proceso que realizaba Drone UPV anteriormente, que se basaba en la experiencia y conocimiento de los desarrolladores de prototipos. La cantidad de información hace que el equipo no pueda tener en cuenta todo sin ningún soporte o herramienta como la que ha sido diseñada.

7. CONCLUSIONES Y FUTURAS LINEAS DE ACTUACIÓN

En el presente TFG se ha modelado, programado y validado una herramienta de optimización para la selección de componentes de drones de carreras. Una vez validada la herramienta se ha podido comprobar los grandes beneficios de su uso respecto a; la reducción del tiempo en la toma de decisión de la combinación óptima teniendo en cuenta las preferencias y objetivos del piloto, en segundos; altos rendimientos en los parámetros que se han tomado como objetivo; una interfaz intuitiva, que puede ser utilizada para cualquier tipo de piloto, incluso sin conocimientos en drones de carreras; y un modelo implementado en una librería libre (Python) que puede ser modificada para futuras necesidades de las empresas. Gracias a las aportaciones de Drone UPV se ha podido definir y validar la herramienta de optimización propuesta

Para la implantación de la herramienta tanto en Drone UPV como en cualquier empresa que se dedique a la venta de componentes o ensamblaje de drones de carreras, se propone la metodología Kaizen. Esta metodología de mejora continua consiste en la implantación de herramientas en pequeños pasos, sin grandes inversiones y con la participación de todo el equipo de trabajo (Conesa, 2007):

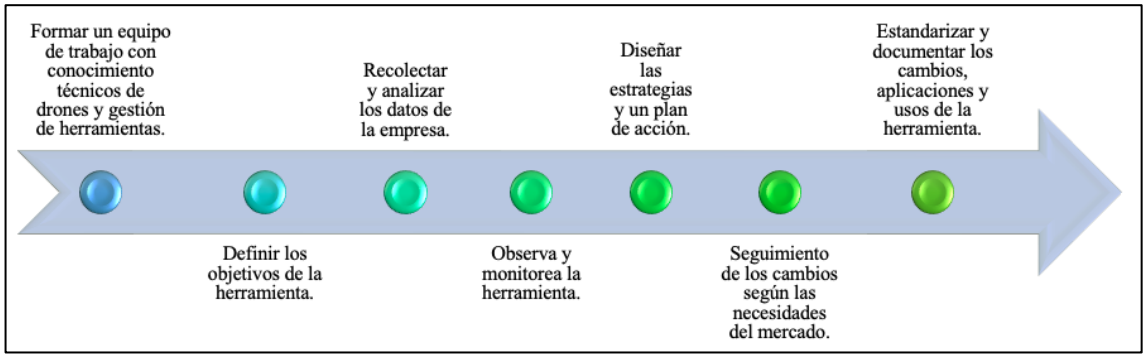

Figura 23. Kaizen para la implementación de la herramienta en las empresas. Fuente: Elaboración propia basada en (Rodriguéz, 2022)

Como futuras líneas de actuación se pueden listar posibles mejoras y para desarrollar el potencial de la herramienta como:

- Desarrollar un mini programa de conexión entre el entorno de programación en Python y la interfaz de Excel, para migrar los datos de forma automática, sin necesidad que trasladar manualmente los datos de la hoja que dispone de la información de los pilotos codificada (Figura 10).
- Integrar más datos experimentales para abarcar más tipos de combinaciones y considerar otro tipo de componentes como cámaras, controladoras, etc.
- Desarrollar una herramienta parecida a la hoja de transcripción de la información de los pilotos a datos codificados para interpretar la solución de forma rápida, sin necesidad de buscar manualmente el código de cada componente.
- Utilizar la misma herramienta para drones de diferentes tipos de competiciones o sectores. Para seleccionar los compontes los drones en las que se realizan pruebas para diferentes funcionalidades como el transporte de materiales, de esta forma se dispone de una herramienta que tenga en cuenta las necesidades del desarrollador, como mayor empuje, potencia o control.

8. BIBLIOGRÁFIA

Alemany, M. (2020). *Tema Programación Lineal de Metodo Cuantitativos en Organización Industrial.* Universitat Politécnica de Valencia . Diapositivas para GIOI-UPV.

Aliexpress. (2022). *Aliexpress*. Obtenido de www.aliexpress.com

Banggood. (2022). *Banggood*. Obtenido de www.banggood.com

Castiblanco, J. M., Garcia-Nieto, S., Simarro, R., & Salcedo, J. (2021). *Experimental study on the dynamic behaviour of drones designed for racing competitions.* Obtenido de 175682932110057. https://doi.org/10.1177/1756.

Castillo, E. C. (2002). *Formulación y Resolución de Modelos de Programación Matemática en Ingeniería y Ciencia.* Escuela Técnica Superior de Ingenieros Industriales, Escuela Técnica Superior de Ingenieros de Caminos, Canales y Puertos. Universidad de Castilla La Mancha. Obtenido de https://www.researchgate.net/profile/Enrique-Castillo-2/publication/319533757_Some_Examples_Using_GAMS/links/5c12180ca6fdcc494ff05673/Some-Examples-Using-GAMS.pdf

Conesa, J. P. (2007). *Kaizen: Cuando la mejora se hace realidad* (Vol. 273). Técnica Industrial.

Corrêa Bernardo, C., Corrêa Chaves, V., Gonçalves Sant'Ana, R., & Pagán Martínez, M. (2018). Perspectivas históricas de la Investigación Operacional. *Bolema: Boletim de Educação Matemática, 61*(32), 354–374. Obtenido de https://doi.org/10.1590/1980-4415v32n61a03

Dantzig, G. B. (1990). Origins of the simplex method. En *A history of scientific computing* (págs. 141-151).

Dantzig, G. B. (2002). *Linear Programming. Operation Research* (Vol. 50). Recuperado el Junio de 2022, de https://doi.org/10.1287/opre.50.1.42.17798

de la Fuente García, D., & Priore Moreno, P. (1996). *Programación lineal entera y programación no lineal.* Oviedo: Universidad de Oviedo. Obtenido de Books Google.

Diaz Muñoz, G. M. (2005). Linear Programming as a Tool for Decision Making (Programación Lineal Como Herramienta Para Toma De Decisiones. *Sotavento* (10). Obtenido de https://ssrn.com/abstract=1508437

DRONE UPV. (2022). *DRONE University*. Recuperado el Mayo de 2022, de https://drone-uni.com/es/droneupv

Esteso Álvares, A. (2018). *Implementación de modelos matemáticos en Pyomo.*

Fourer, R. G. (2003). *AMPL*. Thomson/Brooks/Cole.

GAMS. (2022). *User's Guide*. Recuperado el 2022 de Junio , de GAMS: https://www.gams.com/latest/docs/UG_MAIN.html#UG_Language_Environment

García, M., Quispe, C., & Ráez, L. (08 de Marzo de 2003). Mejora continua de la calidad en los procesos. *Industrial Data*, 89-94. Recuperado el Junio de 2022, de Mentor de CEOs: https://mentordeceos.com/el-ciclo-pdca/

Generación Espontánea UPV. (2022). Recuperado el Mayo de 2022, de Generación Espontánea: https://generacionespontanea.upv.es

Hart, W. E.-P. (2017). *Pyomo – Optimization Modeling in Python. Second Edition* (Vol. 67). Springer. Obtenido de http://www.pyomo.org/about

Jupyter, P. (2022). *Documentación Jupyter lab*. Obtenido de https://jupyterlab.readthedocs.io

Martín, S. F. (1958). Investigación operativa. *Documentación Administrativa*(6). Obtenido de https://revistasonline.inap.es/index.php/DA/article/download/956/1011

Montufar Benítez, M. A., López Pérez, J. F., & Flores Cantú, H. R. (2018). *Investigación de operaciones.* México, México: Grupo Editorial Patria.

Mula, J., Peidro, D., Díaz-Madroñero, M., & Vicens, E. (2010). Mathematical programming models for supply chain production and transport planning. *European Journal of Operational Research, 3*(204), 377–390. Obtenido de https://doi.org/10.1016/j.ejor.2009.09.008

Noticias UPV. (20 de Mayo de 2015). Recuperado el Junio de 2022, de UPV: http://www.upv.es/noticias-upv/noticia-7463-aeronaves-no-t-es.html

Rodriguéz, J. (22 de junio de 2022). *Método Kaizen: definición, pasos y ejemplos*. Recuperado el Julio de 2022, de Hubspot: https://blog.hubspot.es/sales/metodo-kaizen

Roelofs, M., & Bisschop, J. (2006). *Aimms Language Reference.* Lulu.com. Obtenido de https://documentation.aimms.com/_downloads/AIMMS_ref.pdf

Taha, H. A. (2012). *Investigación de operaciones.* México: PEARSON EDUCACIÓN, Novena edición.

Tormos Juan, P., & Lova Ruiz, A. (2003). *Investigación operativa para ingenieros.* Valencia: Universitat Politècnica de València.

Wilson, J. M. (1985). Classification of Models in Operational Research. *3*(36), 253. Obtenido de The Journal of the Operational Research Society: https://doi.org/10.2307/2582257

PRESUPUESTO

1. PRESUPUESTO

En este documento se realiza el cálculo aproximado del coste asociado a cada actividad desarrollado en el TFG. El precio horario se ha basado en los precios del mercado de ingenieros del sector en los tiempos actuales. El presupuesto se va a estructurar en cuatro apartados principales, cada apartado tendrá sus respectivas tareas que estarán relacionadas con el apartado en cuestión. Mediante este análisis de coste se mide la cantidad de trabajo y el valor global del TFG en el apartado (Resumen)

En la tasa horaria se ha estimado la experiencia, titulación y habilidades del perfil encargado de cada una de las tareas, con un sueldo estimado al mercado laboral actual.

1.1. Identificación y descripción del problema

Tarea	Descripción	Cantidad (h)	Tasa horaria (€/h)	Importe (€)
1. Descripción del problema	Estructuración de la problemática para el posterior modelado.	35	20	700,00 €
2. Reuniones	Coordinación con el equipo Drone UPV para la definición del problema	5	15	75,00 €
3. Modelización del problema PLEM	Dentro esta tarea se realiza la búsqueda del modelo que se ajuste al problema	40	30	960,00 €
4. Tutorías	Tutorías con los tutores para la planificación y verificación de la definición de la problemática	5	25	125,00 €
TOTAL				**1.860,00 €**

1.2. Diseño e implementación de la herramienta

Tarea	Descripción	Cantidad (h)	Tasa horaria (€/h)	Importe (€)
1. Definición de la arquitectura	Selección de los elementos y estructura de la herramienta	10	50	500,00 €
2. Implementación de la herramienta en Pyomo	Desarrollo de código implementando el modelo matemático previo	80	70	5.600,00 €
3. Estructuración de los datos	Estructurar los datos para su codificación e implementación en los .Tabs	15	20	300,00 €
4. Desarrollo de la interfaz en Excel mediante Visual Basic	Diseño y desarrollo de las hojas de la hoja de la interfaz y la hoja automática de transcripción	24	50	1.200,00 €
5. Tutorías	Seguimiento del TFG	3	25	75,00 €
TOTAL				**7.675,00 €**

1.3. Experimentación y validación de la herramienta

Tarea	Descripción	Cantidad (h)	Tasa horaria (€/h)	Importe (€)
1. Definición de la metodología de validación	Definición del flujo de trabajo para la validación de la herramienta.	6	20	120,00 €
2. Creación de escenarios	Definición de los escenarios para validar la herramienta	2	20	40,00 €
3. Recopilación de casos reales de pilotos reales	Coordinación con los pilotos de Drone UPV para la recopilación de sus preferencias	1	15	15,00 €
4. Validación de los resultados	Comparación de los resultados de la herramienta frente a las recomendaciones de Drone UPV	20	20	400,00 €
5. Tutorías	Planificación de actuación y seguimiento de TFG	3	25	75,00 €
TOTAL				**650,00 €**

1.4. Redacción y elaboración de la memoria

Tarea	Descripción	Cantidad (h)	Tasa horaria (€/h)	Importe (€)
1. Búsqueda de información	Investigación de artículos, paginas, libros etc. para la creación del marco conceptual	20	15	300,00 €
2. Formato del TFG	Revisión y corrección del formato del TFG, espacios, tipo de letra, tablas, figuras etc.	30	15	450,00 €
3. Diseño de figuras	Creación de figuras representativas para la compresión del TFG.	48	20	960,00 €
4. Tutorías para hito de finalización	Seguimiento final para la entrega del TFG	15	25	375,00 €
TOTAL				**2.085,00 €**

1.5. Resumen

En el último apartado del presupuesto se resumen los conceptos con el total de cada uno de los cuatro apartados, agregándole una desviación de los costes de un 4%.

CONCEPTO	IMPORTE (€)
1.1. Identificación y modelización del problema	1.860,00 €
1.2. Diseño e implementación de la herramienta	7.675,00 €
1.3. Experimentación y validación de la herramienta	650,00 €
1.4. Redacción y elaboración de la memoria	2.085,00 €
Presupuesto TFG	**12.270,00 €**
Desviación 4%	490,80 €
TOTAL	**12.760,80 €**

ANEXOS

ANEXO I. Código en Visual Basic de la interfaz desarrollada

```
Private Sub CommandButton1_Click()
Dim l() As Integer
Dim a() As Integer
Dim b() As Integer
Dim s() As Integer
Dim x As Integer
Dim mag() As String
Dim pc() As String
Dim pb() As String

ReDim l(5)
ReDim a(6)
ReDim b(6)
ReDim s(11)
ReDim mag(5)
ReDim pc(2)
ReDim pb(2)

For x = 1 To 5
l(x) = Worksheets("Transcripcion").Cells(2 + x, 13).Value
Next x

For x = 1 To 6
a(x) = Worksheets("Transcripcion").Cells(2 + x, 16).Value
b(x) = Worksheets("Transcripcion").Cells(2 + x, 16).Value
Next x
```

```
For x = 1 To 11
s(x) = Worksheets("Transcripcion").Cells(2 + x, 7).Value
Next x

For x = 1 To 5
mag(x) = Worksheets("Transcripcion").Cells(2 + x, 9).Value
Next x
For x = 1 To 2
pc(x) = Worksheets("Transcripcion").Cells(15 + x, 12).Value
pb(x) = Worksheets("Transcripcion").Cells(15 + x, 15).Value
Next x

For x = 1 To 5
    If Worksheets("Preferencias").Cells(12, 3).Value = l(x) Then
        Worksheets("Transcripcion").Cells(3, 3).Value = x
    End If
    If Worksheets("Preferencias").Cells(12, 4).Value = l(x) Then
        Worksheets("Transcripcion").Cells(4, 3).Value = x
    End If
Next x

For x = 1 To 6
    If Worksheets("Preferencias").Cells(12, 5).Value = a(x) Then
        Worksheets("Transcripcion").Cells(5, 3).Value = x
    End If
    If Worksheets("Preferencias").Cells(12, 6).Value = a(x) Then
        Worksheets("Transcripcion").Cells(6, 3).Value = x
    End If
Next x
```

```
For x = 1 To 6
    If Worksheets("Preferencias").Cells(12, 7).Value = b(x) Then
        Worksheets("Transcripcion").Cells(7, 3).Value = x
    End If
    If Worksheets("Preferencias").Cells(12, 8).Value = b(x) Then
        Worksheets("Transcripcion").Cells(8, 3).Value = x
    End If
Next x

For x = 1 To 11
    If Worksheets("Preferencias").Cells(12, 9).Value = s(x) Then
        Worksheets("Transcripcion").Cells(9, 3).Value = x
    End If
    If Worksheets("Preferencias").Cells(12, 10).Value = s(x) Then
        Worksheets("Transcripcion").Cells(10, 3).Value = x
    End If
Next x

For x = 1 To 5
    If Worksheets("Preferencias").Cells(12, 11).Value = mag(x) Then
        Worksheets("Transcripcion").Cells(11, 3).Value = x
    End If
    If Worksheets("Preferencias").Cells(12, 12).Value = mag(x) Then
        Worksheets("Transcripcion").Cells(12, 3).Value = x
    End If

    If Worksheets("Preferencias").Cells(12, 13).Value = mag(x) Then
        Worksheets("Transcripcion").Cells(13, 3).Value = x
    End If
    If Worksheets("Preferencias").Cells(12, 14).Value = mag(x) Then
        Worksheets("Transcripcion").Cells(14, 3).Value = x
    End If
Next x
```

```
For x = 1 To 2
    If Worksheets("Preferencias").Cells(12, 15).Value = pc(x) And x =
1 Then
        Worksheets("Transcripcion").Cells(15, 3).Value = x
    End If
    If Worksheets("Preferencias").Cells(12, 15).Value = pc(x) And x =
2 Then
        Worksheets("Transcripcion").Cells(15, 3).Value = 0
    End If
    If Worksheets("Preferencias").Cells(12, 16).Value = pb(x) And x =
1 Then
        Worksheets("Transcripcion").Cells(16, 3).Value = x
    End If
    If Worksheets("Preferencias").Cells(12, 16).Value = pb(x) And x =
2 Then
        Worksheets("Transcripcion").Cells(16, 3).Value = 0
    End If
Next x

End Sub

Private Sub CommandButton2_Click()
Dim pesos() As Single
Dim maxmin() As String

ReDim maxmin(2)
ReDim pesos(7)

Dim j As Integer

For j = 1 To 7
pesos(j) = Worksheets("Preferencias").Cells(18, 3 + j).Value
Next j
```

```
For j = 1 To 2
maxmin(j) = Worksheets("Transcripcion").Cells(32, 1 + j).Value
Next j

If Worksheets("Preferencias").Cells(17, 4).Value = maxmin(1) Then
Worksheets("Transcripcion").Cells(18, 3).Value = pesos(1)
ElseIf Worksheets("Preferencias").Cells(17, 4).Value <> maxmin(1) Then
Worksheets("Transcripcion").Cells(18, 3).Value = ""
End If

If Worksheets("Preferencias").Cells(17, 5).Value = maxmin(1) Then
Worksheets("Transcripcion").Cells(19, 3).Value = pesos(2)
ElseIf Worksheets("Preferencias").Cells(17, 5).Value <> maxmin(1) Then
Worksheets("Transcripcion").Cells(19, 3).Value = ""
End If

If Worksheets("Preferencias").Cells(17, 5).Value = maxmin(2) Then
Worksheets("Transcripcion").Cells(20, 3).Value = pesos(2)
ElseIf Worksheets("Preferencias").Cells(17, 5).Value <> maxmin(2) Then
Worksheets("Transcripcion").Cells(20, 3).Value = ""
End If

If Worksheets("Preferencias").Cells(17, 6).Value = maxmin(1) Then
Worksheets("Transcripcion").Cells(21, 3).Value = pesos(3)
ElseIf Worksheets("Preferencias").Cells(17, 6).Value <> maxmin(1) Then
Worksheets("Transcripcion").Cells(21, 3).Value = ""
End If



If Worksheets("Preferencias").Cells(17, 6).Value = maxmin(2) Then
```

```
Worksheets("Transcripcion").Cells(22, 3).Value = pesos(3)
ElseIf Worksheets("Preferencias").Cells(17, 6).Value <> maxmin(2) Then
Worksheets("Transcripcion").Cells(22, 3).Value = ""
End If

If Worksheets("Preferencias").Cells(17, 7).Value = maxmin(1) Then
Worksheets("Transcripcion").Cells(23, 3).Value = pesos(4)
ElseIf Worksheets("Preferencias").Cells(17, 7).Value <> maxmin(1) Then
Worksheets("Transcripcion").Cells(23, 3).Value = ""
End If

If Worksheets("Preferencias").Cells(17, 7).Value = maxmin(2) Then
Worksheets("Transcripcion").Cells(24, 3).Value = pesos(4)
ElseIf Worksheets("Preferencias").Cells(17, 7).Value <> maxmin(2) Then
Worksheets("Transcripcion").Cells(24, 3).Value = ""
End If


If Worksheets("Preferencias").Cells(17, 8).Value = maxmin(1) Then
Worksheets("Transcripcion").Cells(25, 3).Value = pesos(5)
ElseIf Worksheets("Preferencias").Cells(17, 8).Value <> maxmin(1) Then
Worksheets("Transcripcion").Cells(25, 3).Value = ""
End If

If Worksheets("Preferencias").Cells(17, 8).Value = maxmin(2) Then
Worksheets("Transcripcion").Cells(26, 3).Value = pesos(5)
ElseIf Worksheets("Preferencias").Cells(17, 8).Value <> maxmin(2) Then
Worksheets("Transcripcion").Cells(26, 3).Value = ""
End If


If Worksheets("Preferencias").Cells(17, 9).Value = maxmin(1) Then
```

```
Worksheets("Transcripcion").Cells(27, 3).Value = pesos(6)
ElseIf Worksheets("Preferencias").Cells(17, 9).Value <> maxmin(1) Then
Worksheets("Transcripcion").Cells(27, 3).Value = ""
End If

If Worksheets("Preferencias").Cells(17, 9).Value = maxmin(2) Then
Worksheets("Transcripcion").Cells(28, 3).Value = pesos(6)
ElseIf Worksheets("Preferencias").Cells(17, 9).Value <> maxmin(2) Then
Worksheets("Transcripcion").Cells(28, 3).Value = ""
End If

If Worksheets("Preferencias").Cells(17, 10).Value = maxmin(1) Then
Worksheets("Transcripcion").Cells(29, 3).Value = pesos(7)
ElseIf Worksheets("Preferencias").Cells(17, 10).Value <> maxmin(1)
Then
Worksheets("Transcripcion").Cells(29, 3).Value = ""
End If

If Worksheets("Preferencias").Cells(17, 10).Value = maxmin(2) Then
Worksheets("Transcripcion").Cells(30, 3).Value = pesos(7)
ElseIf Worksheets("Preferencias").Cells(17, 10).Value <> maxmin(2)
Then
Worksheets("Transcripcion").Cells(30, 3).Value = ""
End If



End Sub
```

ANEXO II. Modelo implementado en Pyomo

```
from pyomo.environ import *

z = AbstractModel('MODELO DEFINITIVO')

#INDEX
z.G = Set()
z.L = Set()
z.A = Set()
z.B = Set()
z.M = Set()
z.H = Set()
z.S = Set()
z.W = Set()

#DATA
z.n = Param()
z.ca = Param(z.G,z.L,z.A,z.B)
z.cm = Param(z.M)
z.ch=Param(z.H)
z.li = Param(z.G)
z.la = Param(z.G)
z.ai = Param(z.G)
z.aa = Param(z.G)
z.bi = Param(z.G)
z.ba = Param(z.G)
z.lmax = Param()
z.lmin = Param()
z.amax = Param()
z.amin = Param()
z.bmax = Param()
z.bmin = Param()
z.smax = Param()
z.smin = Param()
z.mamax = Param()
z.mamin = Param()
z.agmax = Param()
z.agmin = Param()
z.mh = Param(z.M,z.H)
z.nc = Param(z.G)
z.pc = Param()
z.nb = Param(z.M)
z.pb = Param()
z.amp = Param(z.S,z.M,z.H)
z.emp = Param(z.S,z.M,z.H)
z.vol = Param(z.S,z.M,z.H)
z.pot = Param(z.S,z.M,z.H)
z.man = Param(z.G,z.L,z.A,z.B)
z.ag = Param(z.G,z.L,z.A,z.B)
z.weight = Param(z.W)
z.max = Param(z.W)
```

```
#VARIABLES

z.Y = Var(z.G, z.L,z.A,z.B, z.M,z.H,domain=Boolean)
z.YA = Var(z.G, z.L,z.A,z.B, domain=Boolean)
z.YM = Var( z.M, domain=Boolean)
z.YH = Var(z.H,domain=Boolean)
z.YS = Var(z.S,z.M,z.H ,domain=Boolean)

#EXPRESSIONS
def CostAir_rule(z):
    return sum(z.ca[g,l,a,b] * z.YA[g,l,a,b]  for g in z.G for l in
z.L for a in z.A for b in z.B)
z.CostAir = Expression(rule=CostAir_rule)

def CostMot_rule(z):
    return sum(z.cm[m] * z.n * z.YM[m]  for m in z.M )
z.CostMot = Expression(rule=CostMot_rule)

def CostHel_rule(z):
    return sum(z.ch[h] * z.n * z.YH[h] for h in z.H)
z.CostHel = Expression(rule=CostHel_rule)

def Amper_rule(z):
    return sum(z.amp[s,m,h] * z.YS[s,m,h] for s in z.S for m in z.M
for h in z.H)
z.Amper = Expression(rule=Amper_rule)

def Empuj_rule(z):
    return sum(z.emp[s,m,h] * z.YS[s,m,h]  for s in z.S for m in z.M
for h in z.H)
z.Empuj = Expression(rule=Empuj_rule)

def Volt_rule(z):
    return sum(z.vol[s,m,h] * z.YS[s,m,h]  for s in z.S for m in z.M
for h in z.H)
z.Volt = Expression(rule=Volt_rule)

def Poten_rule(z):
    return sum(z.pot[s,m,h] * z.YS[s,m,h]  for s in z.S for m in z.M
for h in z.H)
z.Poten = Expression(rule=Poten_rule)

def Manb_rule(z):
    return sum(z.man[g,l,a,b] * z.YA[g,l,a,b]  for g in z.G for l in
z.L for a in z.A for b in z.B )
z.Manb = Expression(rule=Manb_rule)

def Agil_rule(z):
    return sum(z.ag[g,l,a,b] * z.YA[g,l,a,b]  for g in z.G for l in
z.L for a in z.A for b in z.B )
z.Agil = Expression(rule=Agil_rule)

def W1_rule(z):
    return z.weight[1]*(z.CostAir + z.CostMot + z.CostHel)/z.max[1]
z.W1 = Expression(rule=W1_rule)
```

```
def W2_rule(z):
    return z.weight[2]*(z.Amper)/z.max[2]
z.W2 = Expression(rule=W2_rule)

def W3_rule(z):
    return z.weight[3]*(z.Amper)/z.max[3]
z.W3 = Expression(rule=W3_rule)

def W4_rule(z):
    return z.weight[4]*(z.Empuj)/z.max[4]
z.W4 = Expression(rule=W4_rule)

def W5_rule(z):
    return z.weight[5]*(z.Empuj)/z.max[5]
z.W5 = Expression(rule=W5_rule)

def W6_rule(z):
    return z.weight[6]*(z.Volt)/z.max[6]
z.W6 = Expression(rule=W4_rule)

def W7_rule(z):
    return z.weight[7]*(z.Volt)/z.max[7]
z.W7 = Expression(rule=W7_rule)

def W8_rule(z):
    return z.weight[8]*(z.Poten)/z.max[8]
z.W8 = Expression(rule=W8_rule)

def W9_rule(z):
    return z.weight[9]*(z.Poten)/z.max[9]
z.W9 = Expression(rule=W9_rule)

def W10_rule(z):
    return z.weight[10]*(z.Manb)/z.max[10]
z.W10 = Expression(rule=W10_rule)

def W11_rule(z):
    return z.weight[11]*(z.Manb)/z.max[11]
z.W11 = Expression(rule=W11_rule)

def W12_rule(z):
    return z.weight[12]*(z.Agil)/z.max[12]
z.W12 = Expression(rule=W12_rule)

def W13_rule(z):
    return z.weight[13]*(z.Agil)/z.max[13]
z.W13 = Expression(rule=W13_rule)

#OBJECTIVE
#def Objetivo_rule(z):
 #   return z.CostAir + z.CostMot + z.CostHel
#z.obj = Objective(rule=Objetivo_rule, sense=minimize)

#def Objetivo_rule(z):
```

```
    #return z.Amper
#z.obj = Objective(rule=Objetivo_rule, sense=maximize)

#def Objetivo_rule(z):
 #   return z.Amper
#z.obj = Objective(rule=Objetivo_rule, sense=minimize)

#def Objetivo_rule(z):
    #return z.Empuj
#z.obj = Objective(rule=Objetivo_rule, sense=maximize)

#def Objetivo_rule(z):
 #   return z.Empuj
#z.obj = Objective(rule=Objetivo_rule, sense=minimize)

#def Objetivo_rule(z):
 #   return z.Volt
#z.obj = Objective(rule=Objetivo_rule, sense=maximize)

#def Objetivo_rule(z):
    #return z.Manb
#z.obj = Objective(rule=Objetivo_rule, sense=maximize)

#def Objetivo_rule(z):
    #return z.Volt
#z.obj = Objective(rule=Objetivo_rule, sense=minimize)

#def Objetivo_rule(z):
 #   return z.Poten
#z.obj = Objective(rule=Objetivo_rule, sense=maximize)

#def Objetivo_rule(z):
 #   return z.Poten
#z.obj = Objective(rule=Objetivo_rule, sense=minimize)

#def Objetivo_rule(z):
    #return z.Manb
#z.obj = Objective(rule=Objetivo_rule, sense=minimize)

#def Objetivo_rule(z):
    #return z.Agil
#z.obj = Objective(rule=Objetivo_rule, sense=maximize)

#def Objetivo_rule(z):
    #return z.Agil
#z.obj = Objective(rule=Objetivo_rule, sense=minimize)

def Objetivo_rule(z):
    return - z.W1 - z.W2 + z.W3 - z.W4 + z.W5 - z.W6 + z.W7 - z.W8 +
z.W9 - z.W10 + z.W11 - z.W12 + z.W13
z.obj = Objective(rule=Objetivo_rule, sense=maximize)

#RESTRICCIONES
def Selec_rule(z):
```

```
    return sum(z.Y[g,l,a,b,m,h] for g in z.G for l in z.L for a in z.A
for b in z.B for m in z.M for h in z.H) == 1
z.Selec = Constraint(rule=Selec_rule)

def Air_rule(z,g,l,a,b):
    return sum(z.Y[g,l,a,b,m,h] for m in z.M for h in z.H) ==
z.YA[g,l,a,b]
z.Air = Constraint(z.G,z.L,z.A,z.B,rule=Air_rule)

def Mot_rule(z,m):
    return sum(z.Y[g,l,a,b,m,h] for g in z.G for l in z.L for a in z.A
for b in z.B for h in z.H) == z.YM[m]
z.Mot = Constraint(z.M, rule=Mot_rule)

def Hel_rule(z,h):
    return sum(z.Y[g,l,a,b,m,h] for g in z.G for l in z.L for a in z.A
for b in z.B for m in z.M) == z.YH[h]
z.Hel = Constraint(z.H,rule=Hel_rule)

def Stick_rule(z,s,m,h):
    if s<=z.smax or s>=z.smin:
        return sum(z.Y[g,l,a,b,m,h] for g in z.G for l in z.L for a in
z.A for b in z.B)== z.YS[s,m,h]
z.Stick = Constraint(z.S, z.M, z.H, rule=Stick_rule)

def Stick2_rule(z,s):
    if s>z.smax or s<z.smin:
        return sum(z.YS[s,m,h] for m in z.M for h in z.H)==0
    else:
        return Constraint.Skip
z.Stick2 = Constraint(z.S, rule=Stick2_rule)

def AcotarMotHel_rule(z,s,m,h):
    return z.YS[s,m,h]<=z.mh[m,h]
z.AcotarMotHel = Constraint(z.S,z.M,z.H, rule=AcotarMotHel_rule)

def Longitud1_rule(z):
    return sum(z.YA[g,l,a,b] for g in z.G for l in z.L if l >= z.lmin
and l <= z.lmax for a in z.A for b in z.B)==1
z.Longitud1 = Constraint(rule=Longitud1_rule)

def AnguloA1_rule(z):
    return sum(z.YA[g,l,a,b] for g in z.G for l in z.L for a in z.A if
a >= z.amin and a <= z.amax for b in z.B)==1
z.AnguloA1 = Constraint(rule=AnguloA1_rule)


def AnguloB1_rule(z):
    return sum(z.YA[g,l,a,b] for g in z.G for l in z.L for a in z.A
for b in z.B if b >= z.bmin and b <= z.bmax)==1
z.AnguloB1 = Constraint(rule=AnguloB1_rule)


def AjustesPi_rule(z):
```

```
    return sum(z.nc[g]*z.YA[g,l,a,b] for g in z.G for l in z.L for a
in z.A for b in z.B) <= z.pc
z.AjustesPi = Constraint(rule=AjustesPi_rule)

def Baterias_rule(z,m):
    if z.pb == 0 and z.nb[m] == 1:
        return 0 ==z.YM[m]
    else:
        return Constraint.Skip
z.Baterias = Constraint(z.M, rule=Baterias_rule)

def Maniobr_rule(z):
    return z.mamin <= sum( z.man[g,l,a,b]*z.YA[g,l,a,b] for g in z.G
for l in z.L for a in z.A for b in z.B)
z.Maniobr = Constraint( rule = Maniobr_rule)

def Maniobr1_rule(z):
    return z.mamax >= sum( z.man[g,l,a,b]*z.YA[g,l,a,b] for g in z.G
for l in z.L for a in z.A for b in z.B)
z.Maniobr1 = Constraint( rule = Maniobr1_rule)

def Agilidad_rule(z):
    return z.agmin <= sum( z.ag[g,l,a,b]*z.YA[g,l,a,b] for g in z.G
for l in z.L for a in z.A for b in z.B)
z.Agilidad = Constraint( rule = Agilidad_rule)

def Agilidad1_rule(z):
    return z.agmax >= sum( z.ag[g,l,a,b]*z.YA[g,l,a,b] for g in z.G
for l in z.L for a in z.A for b in z.B)
z.Agilidad1 = Constraint( rule = Agilidad1_rule)

def Geo_rule(z,g):
    if z.li[g] > 1:
        return sum(z.YA[g,l,a,b] for l in z.L if l <= z.li[g]-1  for a
in z.A for b in z.B)==0
    else:
        return Constraint.Skip
z.Geo = Constraint(z.G,rule = Geo_rule)

def Geo1_rule(z,g):
    if z.la[g] < 5:
        return sum(z.YA[g,l,a,b] for l in z.L if l >= z.la[g]+1 for a
in z.A for b in z.B)==0
    else:
        return Constraint.Skip
z.Geo1 = Constraint(z.G,rule = Geo1_rule)

def AirAngulo_rule(z,g):
    if z.ai[g] > 1:
        return sum(z.YA[g,l,a,b] for l in z.L for a in z.A if a <=
(z.ai[g]-1) for b in z.B)==0
    else:
        return Constraint.Skip
z.AirAngulo = Constraint(z.G,rule = AirAngulo_rule)
```

```
def AirAngulo1_rule(z,g):
    if z.aa[g] < 6:
        return sum(z.YA[g,l,a,b] for l in z.L for a in z.A if a >=
(z.aa[g]+1) for b in z.B)== 0
    else:
        return Constraint.Skip
z.AirAngulo1 = Constraint(z.G,rule = AirAngulo1_rule)

def AirAngulob_rule(z,g):
    if z.bi[g] > 1:
        return sum(z.YA[g,l,a,b] for l in z.L for a in z.A  for b in
z.B if b <= (z.bi[g]-1))==0
    else:
        return Constraint.Skip
z.AirAngulob = Constraint(z.G,rule = AirAngulob_rule)

def AirAngulob1_rule(z,g):
    if z.ba[g] < 6:
        return sum(z.YA[g,l,a,b] for l in z.L for a in z.A  for b in
z.B if b >= (z.ba[g]+1))==0
    else:
        return Constraint.Skip
z.AirAngulob1 = Constraint(z.G,rule = AirAngulob1_rule)

def Geometrias_rule(z,g):
    if g<=2:
        return sum(z.YA[g,l,a,b] for l in z.L for a in z.A  for b in
z.B if b != a)==0
    else:
          return Constraint.Skip
z.Geometrias = Constraint(z.G,rule = Geometrias_rule)

#LOAD DATA
dp = DataPortal()
dp.load(filename='Index_g.tab', set=z.G)
dp.load(filename='Index_l.tab', set=z.L)
dp.load(filename='Index_a.tab', set=z.A)
dp.load(filename='Index_b.tab', set=z.B)
dp.load(filename='Index_m.tab', set=z.M)
dp.load(filename='Index_h.tab', set=z.H)
dp.load(filename='Index_s.tab', set=z.S)
dp.load(filename='Index_w.tab', set=z.W)
dp.load(filename='Param_n.tab', param=z.n)
dp.load(filename='Param_ca.tab', param=z.ca, index=(z.G,z.L,z.A,z.B))
dp.load(filename='Param_cm.tab', param=z.cm, index=(z.M))
dp.load(filename='Param_ch.tab', param=z.ch, index=(z.H))
dp.load(filename='Param_li.tab', param=z.li, index=(z.G))
dp.load(filename='Param_la.tab', param=z.la, index=(z.G))
dp.load(filename='Param_ai.tab', param=z.ai, index=(z.G))
dp.load(filename='Param_aa.tab', param=z.aa, index=(z.G))
dp.load(filename='Param_bi.tab', param=z.bi, index=(z.G))
dp.load(filename='Param_ba.tab', param=z.ba, index=(z.G))
dp.load(filename='Param_lmax.tab', param=z.lmax)
dp.load(filename='Param_lmin.tab', param=z.lmin)
dp.load(filename='Param_amax.tab', param=z.amax)
```

```
dp.load(filename='Param_amin.tab', param=z.amin)
dp.load(filename='Param_bmax.tab', param=z.bmax)
dp.load(filename='Param_bmin.tab', param=z.bmin)
dp.load(filename='Param_smin.tab', param=z.smin)
dp.load(filename='Param_smax.tab', param=z.smax)
dp.load(filename='Param_mamin.tab', param=z.mamin)
dp.load(filename='Param_mamax.tab', param=z.mamax)
dp.load(filename='Param_agmin.tab', param=z.agmin)
dp.load(filename='Param_agmax.tab', param=z.agmax)
dp.load(filename='Param_mh.tab', param=z.mh, index=(z.M,z.H))
dp.load(filename='Param_nc.tab', param=z.nc, index=(z.G))
dp.load(filename='Param_nb.tab', param=z.nb, index=(z.M))
dp.load(filename='Param_amp.tab', param=z.amp, index=(z.S,z.M,z.H))
dp.load(filename='Param_emp.tab', param=z.emp, index=(z.S,z.M,z.H))
dp.load(filename='Param_vol.tab', param=z.vol, index=(z.S,z.M,z.H))
dp.load(filename='Param_pot.tab', param=z.pot, index=(z.S,z.M,z.H))
dp.load(filename='Param_pb.tab', param=z.pb)
dp.load(filename='Param_pc.tab', param=z.pc)
dp.load(filename='Param_man.tab', param=z.man,
index=(z.G,z.L,z.A,z.B))
dp.load(filename='Param_ag.tab', param=z.ag, index=(z.G,z.L,z.A,z.B))
dp.load(filename='Param_weight.tab', param=z.weight, index=(z.W))
dp.load(filename='Param_max.tab', param=z.max, index=(z.W))

#OPTIMIZE
inst = z.create_instance(dp)
#inst.pprint()
opt = SolverFactory('gurobi')
results = opt.solve(inst)
#PRINT RESULTS
inst.solutions.store_to(results)
print(results)
import pyoutil as pu
pu.save_results_to_file(inst, results)
```