Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Sistema de Inteligencia Artificial para la optimización comercial de vuelos

Artificial Intelligence System for commercial flight optimisation

Nerea Rodríguez Hernández

La Laguna, 5 de julio de 2022

D. **Juan José Salazar González**, con N.I.F. 43.356.435-D, profesor Catedrático de Universidad adscrito al Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

”*Sistema de Inteligencia Artificial para la optimización comercial de vuelos*”

ha sido realizada bajo su dirección por Dña. **Nerea Rodríguez Hernández**, con N.I.F. 78.645.381-D.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 5 de julio de 2022

Agradecimientos

El amor, la paciencia y por todas las veces que me ayudaron a seguir luchando, los mismos que han permitido formarme y dedicarme a lo que quiero: mi padre, mi madre, mi hermana, mi hermano y mi tía. No son nada más y nada menos que los principales testigos del esfuerzo y sacrificio que he dedicado en este nuevo logro.

El tiempo y la confianza que muchas veces me ha faltado a mí misma para conseguir llegar a donde he llegado y que sin ese apoyo incondicional no podría haberlo conseguido: mis amigas del Palm-Mar y mis amigos de la carrera. Dispuestos de manera desinteresada a acompañarme en este largo camino, en cada larga y agotadora noche de estudio.

La universidad por recibirme con los brazos abiertos y por darme la oportunidad de realizar este trabajo tutorizado por un gran profesional como es D. Juan José Salazar González que me ha brindado conocimientos y un apoyo incomparable.

Gracias por siempre desear y anhelar siempre lo mejor para mí, los consejos y por cada una de sus palabras que me guiaron y me seguirán guiando.

Licencia

© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.

Resumen

El objetivo principal que se persigue con el desarrollo de este proyecto es aportar mejoras en uno de los pilares básicos de la operativa del transporte aéreo, la programación comercial de vuelos. Estas mejoras mencionadas permitirán la realización de los siguientes objetivos: aumentar la seguridad en el entorno aeroportuario, aumentar la productividad en el transporte de pasajeros en las empresas que forman de la cadena de valor, y aumentar la satisfacción final del pasajero.

Palabras clave: Programación de vuelos, sistema de inteligencia artificial, logística.

Abstract

The main objective pursued with the development of this project is to bring improvements to one of the basic pillars of air transport operations, the commercial scheduling of flights. These improvements will enable the following objectives to be achieved: to increase safety in the airport environment, to increase productivity in passenger transport in the companies that form both a direct and indirect part of the value chain, and to increase final passenger satisfaction.

Keywords: Flight scheduling, artificial intelligence system, logistics.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción.

1.1. Motivación.

El servicio regional de vuelos requiere a las empresas de transporte que su oferta se adecúe de manera acorde a la temporalidad de la demanda de los pasajeros. Esta temporalidad es tan granular que llega incluso a necesitar ser diferente dependiendo de la franja horaria dentro de un mismo día, ya que no es lo mismo la necesidad de un pasajero que viaja por negocios de lunes a viernes que la del pasajero que lo hace por placer, detectándose incluso variaciones de temporalidad en el pasajero de placer dependiendo de las épocas del año o incluso periodos o días concretos. Las empresas de transporte deben adecuarse a esta demanda variable optimizando además los recursos, tanto humanos como físicos, necesarios para atender dicha demanda.

Este planteamiento obliga a la empresa regional a continuos ajustes en su programación para atender de manera eficaz la variabilidad en la demanda con la subsiguiente carga de trabajo en las áreas afectadas que deben trabajar de manera coordinada, como perfectos engranajes dentro de la cadena de valor, con otras áreas como pueden ser de programación de tripulantes, programación de flota, áreas de mantenimiento de aeronaves, etc.

Estas tareas, con el aumento en la producción se hacen cada vez más complejas, necesitando cada vez más tiempo para realizar programaciones que puedan dar la respuesta adecuada a la demanda manteniendo un alto grado de productividad.

1.2. Antecedentes y estado actual del tema.

1.2.1. Antecedentes.

La necesidad de las compañías áreas para resolver estos problemas logísticos complejos viene ligada a lo caro que resultan los recursos de los aviones y la tripulación. Estos recursos deben sujetarse a varias reglas para que cada vuelo esté cubierto por un avión y una tripulación.

La planificación de los dos recursos nombrados (enrutamiento de aeronave y emparejamiento de la tripulación) se realiza comúnmente de manera separada. La implementación

de un único modelo que se ajuste a los recursos de las tripulaciones y sus limitaciones daría lugar a un problema intratable, ya que el modelo se tendría que ajustar a cada una de las normas de cada empresa.

El artículo “Approaches to solve the fleet-assignment, aircraft-routing, crew-pairing and crew-rostering problems of a regional carrier” (Salazar-González, 2014) presenta el resultado de un Proyecto de investigación para una compañía aérea que opera servicios interinsulares en las Islas Canarias (España), Marruecos y Portugal. El objetivo principal era resolver problemas de asignación de flotas, rutas de aeronaves, emparejamiento de tripulaciones y reagrupamiento de tripulaciones a partide de datos de casos reales. El artículo describe un algoritmo heurístico basa en un modelo de programación.

Otro trabajo que ha sido pionero es Cordeau et al. (2001) presenta una formulación de variables de ruta en la que todas las rutas de las aeronaves y también todas las rutas de la tripulación representan columnas en un modelo de programación lineal entera mixta (MILP). Proponen además un enfoque heurístico que emplea un modelo de descomposición de Benders para resolver la relajación de programación lineal del modelo.

1.2.2. Comercialización de vuelos programados.

La programación comercial de vuelos es un aspecto clave de la cadena aeronáutica. Para ello, el objetivo de este proyecto es el estudio de la optimización de la misma utilizando una herramienta de Inteligencia Artificial.

Dicha herramienta para la programación de vuelos requerirá de la implementación de un motor de Inteligencia Artificial que permita a través de algoritmos de aprendizaje de datos, la realización de propuestas optimizadas de vuelos en base a la demanda. Los factores principales de innovación con relación a herramientas ya que existentes serán la de poder generar escenarios con múltiples operadores, flotas y compañías comercializadoras, que tengan en cuenta no sólo el factor de ocupación de los vuelos sino también el condicionante de las flotas a utilizar, factores de coste de producción y las condiciones particulares del transporte aéreo regional como puede ser las líneas de obligado servicio.

En el mercado ya existen herramientas de optimización de vuelos, pero estas no se adaptan a las necesidades particulares de las aerolíneas regionales. La particularidad de las aerolíneas regionales es que la demanda es muy variable en cortos espacios de tiempo, por lo que es necesario, casi semanalmente, ajustar la programación dependiendo de las necesidades de la población y de los servicios dependientes de la llegada de los pasajeros.

La volatilidad de la demanda, así como la necesidad de las aerolíneas de optimizar al máximo sus recursos, hace indispensable disponer de herramientas que permitan de manera dinámica, el ajuste del número de plazas ofertadas.

La granularidad en la realización de estos ajustes debe llegar a nivel de día o periodo específico (festividades, eventos puntuales, puentes. . .) con el objetivo de dar respuesta a las necesidades de los pasajeros.

1.3. Objetivos.

1.3.1. Objetivos generales.

En el presente problema se detallan las acciones necesarias para la investigación de las soluciones de Inteligencia Artificial definidas en el apartado anterior, el objetivo que se pretende mejorar y la tecnología aplicada a cada una de las acciones.

Los objetivos generales que se persiguen a la hora de realizar este Trabajo de Fin de Grado son:

- Una primera toma de contacto de problemas lineales en el mundo real.
- Aprender y entender el problema de enrutamiento de aeronaves.
- Aprender y entender el problema de emparejamiento de tripulación.
- Aprender y entender el problema de asignación de flota.
- Aprender y entender las distintas técnicas de aprendizaje por refuerzo.
- Proponer una herramienta útil para la programación comercial de vuelos.

1.3.2. Objetivos específicos.

Los objetivos específicos que se persiguen a la hora de elaborar este Trabajo de Fin de Grado son los siguientes:

- Desarrollar un pseudocódigo de aprendizaje por refuerzo.
- Analizar las distintas técnicas de programación de vuelos.

1.4. Estructura de la memoria.

En esta memoria del Trabajo de Fin de Grado se va a desarrollar un estudio sobre la programación comercial de vuelos y las técnicas más importantes para el desarrollo de una inteligencia artificial utilizando métodos de aprendizaje por refuerzo.

Se ha estructurado de la siguiente manera:

- El primer capítulo es introductorio donde se explica brevemente el problema, los antecedentes y los objetivos que se plantea para el desarrollo del Trabajo de Fin de Grado.
- El segundo capítulo redacta los fundamentos teóricos en los que se basa este Trabajo de Fin de Grado.

- El tercer capítulo se redacta la definición del problema, así como su modelo matemático y el enfoque de la solución.
- El cuarto capítulo redacta el aprendizaje por refuerzo, la estrategia de búsqueda y el enfoque de la solución.
- El quinto capítulo se muestra el mockup diseñado para una aplicación gráfica para usuario, ademas de la implementación de una aplicación gráfica de usuario real.
- El sexto capítulo se redactan los resultados obtenidos de las distintas evaluaciones de los algoritmos.
- El séptimo y octavo capítulo se redactan las conclusiones y líneas futuras de este proyecto, en español y en inglés respectivamente.
- El noveno capítulo redacta el presupuesto de la realización de este Trabajo de Fin de Grado.

1.5. Herramientas utilizadas.

En el siguiente apartado se explican las distintas herramientas que se han utilizado para le desarrollo del estudio del presente Trabajo Fin de Grado.

1.5.1. Entorno de programación.

Para el desarrollo de este proyecto he utilizado la herramienta de *Google colab*, también conocida como *'Colaboraty'*. Esta herramienta permite ejecutar código arbitrario en el navegador y esta diseñada especialmente para la programación de tareas de aprendizaje automático y análisis de datos.

Por otro lado, para el desarrollo de la herramienta útil he utilizado el editor de código fuente Visual Studio Code. Fue desarrollado por Microsoft y te permite instalarlo en los distintos sistemas operativos tanto Windows, Linux, macOsc y Web. Esta herramienta es bastante versátil porque te permite realizar depuración de código, control integrado de Git, así como otras funcionalidades que hacen cómoda la programación.

1.5.2. Lenguaje de programación.

Se ha hecho uso del lenguaje de programación Python. El lenguaje de programación Python es un lenguaje interpretado de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, la programación funcional.

En este trabajo se ha hecho uso de la programación funcional, en la parte de desarrollo donde he utilizado la herramienta *Google colab*, haciendo uso de las celdas que nos proporciona para la ejecución de pequeños trozos de código. También se ha hecho uso de la programación orientada objeto en el desarrollo de la interfaz gráfica de usuario (GUI).

1.5.3. Implementación de la interfaz gráfica de usuario.

Para la implementación de la interfaz gráfica de usuario, herramienta útil, he hecho uso de PyQt5 (Meier, 2019). Esta herramienta es una de las ultimas versiones de un conjunto de herramientas widgets GUI desarrollado por Riverbank Computing.

Se trata de una interfaz Python para Qt, una de las librerías GUI multiplataforma más completas y populares. Finalmente, PyQt5 es una gran combinación del lenguaje de programación Python y la librería Qt.

Capítulo 2

Fundamentos teóricos.

En este capítulo se hará una breve descripción del campo en el que se ha trabajado a lo largo del Trabajo de Fin de Grado el cual es la investigación de sistemas de inteligencia artificial en la programación comercial de vuelos, así como la aplicación de distintas técnicas que existen.

2.1. Problema de programación de vuelos.

El problema de la programación de vuelos estriba en que los aviones y las tripulaciones son recursos caros que necesitan una utilización eficiente. Estos recursos deben asignarse a un horario de coste mínimo sujetos a varias reglas, de modo que cada uno de los vuelos sea cubierto por un solo avión y una sola tripulación. Por ello, en la mayoría de los casos a la hora de resolver el problema se resuelven de manera separada, y la ruta de la aeronave se resuelve primero y posteriormente el emparejamiento de tripulación.

Este problema de optimización completo, en la que en cada partición es una colección de secuencia de vuelos. Los vuelos de una secuencia deben ser asignados al mismo operador aéreo. Encontrar la partición de la ruta de la aeronave se llama ruta de la aeronave (*crew-route*), encontrar la partición de la ruta de la tripulación se llama emparejamiento de la tripulación (*crew-pairing*), y todo el problema de optimización (incluyendo también la asignación de la flota) se llama problema de enrutamiento (*routing problem*).

2.1.1. Problema de enrutamiento de aeronaves.

La definición al problema de enrutamiento viene dada por un conjunto de tramos de vuelo que deben ser volador por un único tipo de aeronave (Mirjafari et al., 2020). Este problema de programación de rutas de aeronaves y tripulaciones consiste en determinar un conjunto de rutas de aeronaves y emparejamientos de tripulaciones de coste mínimo, de forma que cada tramo de vuelo sea cubierto por una aeronave y una tripulación, y se satisfagan las restricciones laterales.

Dado un conjunto de tramos de vuelo que deben ser volados por un único tipo de aeronave, el problema de programación simultánea de rutas de aeronaves y tripulaciones

consiste en determinar un conjunto de rutas de aeronaves y emparejamientos de tripulaciones de coste mínimo, de forma que cada tramo de vuelo sea cubierto por una aeronave y una tripulación, y se satisfagan las restricciones laterales.

Se entiende como restricción lateral a aquellas impuestas por las operadoras, como el tiempo máximo de vuelo y los requisitos de mantenimiento que solo afectan a las tripulaciones o a las aeronaves. Por otro lado, existen restricciones de enlace que son los que imponen los tiempos mínimos de conexión para las tripulaciones que dependen de las conexiones de las aeronaves. Para tratar estas restricciones de enlace, se propone un enfoque de solución basado en la descomposición de Benders.

El proceso de solución itera entre un problema maestro que resuelve el problema de las rutas de las aeronaves y un subproblema que resuelve el problema de los emparejamientos de las tripulaciones. Debido a su particular estructura, ambos problemas se resuelven por generación de columnas. Se utiliza un método heurístico de branch-and-bound para calcular las soluciones enteras. En un conjunto de instancias de prueba basadas en datos proporcionados por una compañía aérea, el enfoque integrado produjo un importante ahorro de costes en comparación con el proceso de planificación secuencial utilizado habitualmente en la práctica. La instancia más grande resuelta contiene más de 500 tramos de vuelo durante un periodo de 3 días.

2.1.2. Problema de emparejamiento de tripulación.

El problema de la programación de la tripulación, debido a su complejidad, se divide en dos subproblemas: el emparejamiento de las tripulaciones y la asignación de la tripulación (Gélvez, 2018).

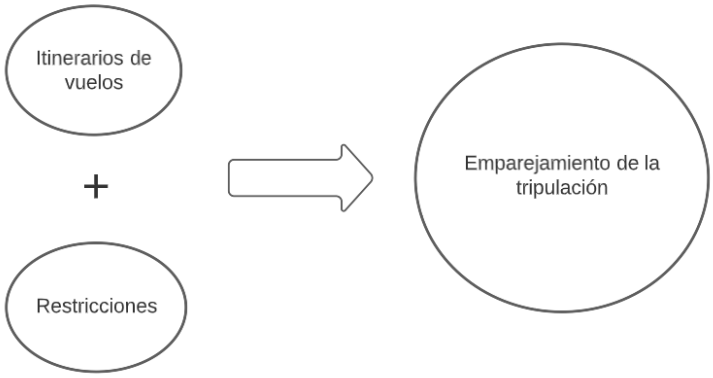

Figura 2.1: Componentes del problema de emparejamiento de tripulación.

El problema de emparejamiento de tripulación se puede enunciar como un par *(pairing)* de secuencias de vuelos que inician y finalizan en una misma base. Des esta forma, en la secuencia el aeropuerto de destino coincide con el aeropuerto de origen del siguiente vuelo. Cada uno de los emparejamiento tiene un costo asociado.

EL objetivo de este problema es encontrar un subconjunto de estas parejas cuyo coste sea mínimo, y que a su vez cubran todas las rutas programadas. El problema de

emparejamiento de tripulaciones tiene además que tener en cuenta una serie de normas y restricciones, bien dadas por las operadoras de la aerolínea.

2.1.3. Problema de asignación de flota.

En el problema de asignación de flota, los aviones disponibles por cada operadora son asignados a diferentes rutas de vuelo. Cada tramos de vuelo tiene asignado unos ingresos propios que dependen de la demanda del mercado.

El objetivo de ese problema consiste en maximizar las ganancias que se obtienen por cada vuelo, teniendo en cuenta que los vuelos solo pueden estar cubiertos por los aviones disponibles (Gélvez, 2018).

Capítulo 3

Definición del problema.

3.1. Definición del problema.

El problema al que nos enfrentamos estriba en una compañía de vuelos regional que oferta una determinada cantidad de vuelos en un día, la cual dese encontrar un itinerario de rutas de coste mínimo que satisfaga todos los vuelos. Esta compañía de vuelos dispone dos operadores de vuelos distintas (OP1, OP2). Cada una de las operadoras cuenta con sus propias tripulaciones y aviones. Una de las principales restricciones es que las tripulaciones no pueden ser intercambiadas, es decir, las tripulaciones de una operadora no pueden realizar vuelos en los aviones de la otra operadora.

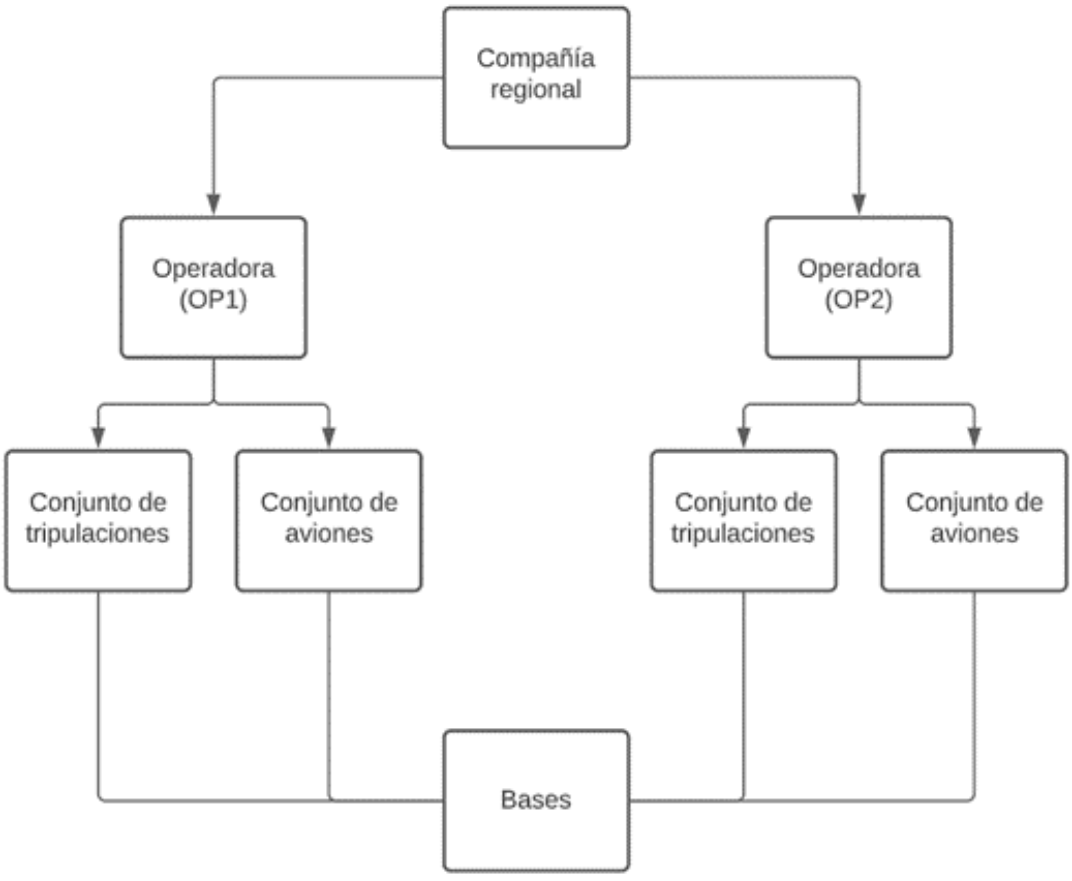

Figura 3.1: Desglose de la distribución de la compañía.

La compañía opera sus vuelos interinsulares conectando pequeños aeropuertos. Estos aeropuertos tienen un horario especifico en el que todos se encuentran cerrados por la noche, esto significa que los vuelos también estarán planificados dentro de un marco horario (entre las 07:00 y las 23:00). En el problema se ven implicados siete aeropuertos:

- Aeropuerto de Las Palmas de Gran Canaria: LPA.
- Aeropuerto de Tenerife Norte: TFN.
- Aeropuerto de Tenerife Sur: TFS.
- Aeropuerto de La Palma: SPC.
- Aeropuerto del Hierro: VDE.
- Aeropuerto de Lanzarote: ACE.
- Aeropuerto de Fuerteventura: FUE.

Aunque se cuente con siete aeropuertos solo dos aeropuertos de los nombrados anteriormente están determinados como bases, el aeropuerto LPA y TFN. Esta especificación es importante ya que se convierte en una restricción para el problema, esto significa que todas las rutas han de empezar en una base y además tienen que terminar la ruta obligatoriamente en la misma base de donde se partió.

Por otro lado, las rutas de las tripulaciones también cuentan con unas restricciones específicas que están definidas por cada operadora. Por ejemplo, se determina la cantidad máxima de vuelos que puede contener una ruta, así como la duración máxima de cada una de las rutas. Además, el número de rutas activas en un determinado instante de tiempo ha de ser menor que la cantidad de tripulaciones disponibles que había en la base al comienzo del día.

Para este problema el tiempo mínimo que puede transcurrir entre dos vuelos de una misma ruta es de 20 minutos (tiempo necesario para preparar un vuelo). El tiempo de conexión entre dos vuelos no puede ser mayor a 3 horas (180 minutos). Los tiempos de conexión se pueden dividir en dos tipos, cortas y largas. Los tiempos de conexión se consideran cortos cuando el tiempo es mayor a 20 minutos y menos a 30 minutos, y la diferencia con los tiempos de conexión larga es que en estas últimas la tripulación puede efectuar un cambio de avión antes de continuar con el siguiente vuelo de la ruta. Este cambio de avión provoca una penalización en el coste de esta debido al tiempo que se le añade al realizar el cambio de la tripulación de un avión a otro.

En conclusión, el problema estudiado en este trabajo busca la solución de una asignación factible de vuelos a operadoras, así como las rutas de los aviones y las tripulaciones de manera que se satisfagan todos los vuelos ofertados, con el objetivo de minimizar el coste total y maximizar la robustez de la planificación.

3.2. Modelo matemático.

En este capítulo se explicará el modelo matemático de variables de arco para aviones y de camino para tripulaciones presentado en el artículo científico Salazar-González (2014).

3.2.1. Introducción a conceptos del modelo matemático.

La formulación del modelo viene dada por dos grafos acíclicos: el primero gráfico de aviones (*aircraft graph*) y el segundo gráfico de rutas (*crew graph*). Detalles del modelo pueden encontrarse en el artículo científico Cacchiani and Salazar-González (2020). El gráfico de aviones representa la secuencia de vuelos válidas para las rutas de los aviones, mientras que el gráfico de rutas representa una secuencia de vuelos válida para rutas de tripulaciones. Decimos que una ruta de vuelos es factible para una ruta de avión si el tiempo de conexión entre los dos vuelos es mayor a 20 minutos. Por otro lado, una secuencia de vuelos es considerada factible para una ruta de tripulación si el tiempo de conexión es mayor a 20 minutos y menor a 180 minutos.

Dada la descripción, el grafo de tripulaciones está contenido en el grafo de enriamiento, ya que el primero es una versión más restrictiva del segundo. Los dos grafos cuentan con el mismo conjunto N de nodos, dados por la unión de los del conjunto de nodos N^f (representa los vuelos), $N^b d$ (representa las bases de salida) y $N^b a$ (representando las bases de llegada).

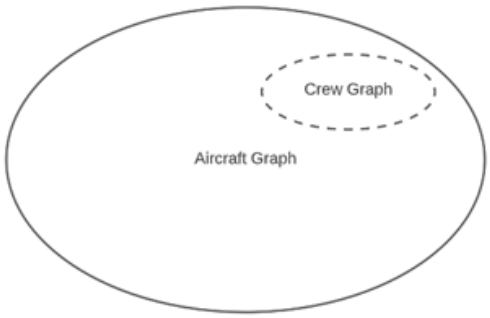

Figura 3.2: Esquema de *Aircraft graph*.

Para nuestro problema, podemos observar en la siguiente figura un ejemplo de aircraft graph (G^a). En este grafo podemos ver seis instancias de con vuelos (nodos blancos), que conectan tres aeropuertos: TFN, LPA, SPC.

En el grafo estan representados como T, L, S respectivamente. Los nodos con un grosor mayor al resto son los aeropuertos bases. El par de letras en los nodos representa el par de vuelos, de manera que la primera letra representa el aeropuerto de destino y la segunda letra representa el aeropuerto de destino (origen-destino) de cada uno de los vuelos.

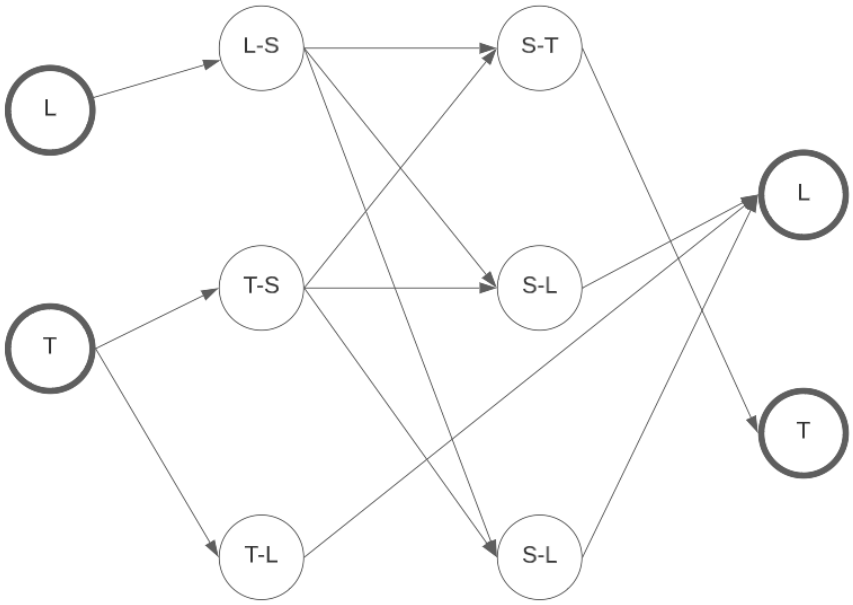

Figura 3.3: *Aircraft graph* para un caso ejemplifico.

El *aircraft graph* viene dado por $G^a = (N, A^a)$, de igual forma el *crew graph* viene dado por $G^c = (N, A^c)$. El primer grafo contiene un arco $(i, j) \in A^a$, de igual forma el segundo grafo también contiene un arco $(i, j) \in A^c$, siempre y cuando se cumpla alguna de estas restricciones:

- $(i, j) \in N^f$ representa a dos vuelos que pueden ir en secuencia en una ruta.
- Correspondiéndose la componente $i \in N^b d$ con una base y $j \in N^b d$ y la componente $j \in N^f$ siendo el primer vuelo de la ruta que parte de esa misma base.
- Correspondiéndose la componente $j \in N^b a$ con una base y la componente $i \in N^f$ siendo el último vuelo de la ruta que termina en esa base.

Por otro lado, y como se ha mencionado en el anterior apartado, la aerolínea cuenta con dos operadoras que van a estar representadas como el conjunto K. Con ello podemos definir a n_c^{kl} y n_a^{kl} para $k \in K$, como el número de tripulaciones y aviones respectivamente, que se encuentran para la operadora k en la base l.

Dada la necesidad de modelar la naturaleza del problema a través de una única función objetivo se consideró la definición de los cuatro siguientes pesos:

- Definimos α como el peso sobre la suma de los tiempos de conexión sobre las rutas de las tripulaciones.
- Definimos β_k como el peso sobre el número de rutas de tripulación para cada operadora la operadora $k \in K$.
- Definimos γ_k como el peso sobre el número de las rutas de los aviones para cada operadora k.
- Definimos δ como el peso sobre el número de cambios de avión durante las rutas de las tripulaciones.

3.2.2. Descripción del modelo matemático.

La formulación matemática para el problema minimiza:

$$\sum_{k\in K, l\in N^{nb}, R\in R_c^{kl}} (\alpha \cdot c_R + \beta_k) \cdot x_R + \sum_{k\in K, l\in N^{nb}, i\in N^{bd}, j\in N^f:(i,j)\in A^a} \gamma_k \cdot y_{ij}^{kl} + \delta \cdot \sum_{(i,j)\in A^c\ A_s^c} z_{ij}, (1)$$

Sujeta a las distintas restricciones en las tripulaciones:

$$\sum_{k\in K, l\in N^{bd}, R\in R_c^{kl}: i\in R} x_R = 1 \qquad \forall i \in N^f, (2)$$

$$\sum_{R\in R_c^{kl}} x_R \leq n_c^{kl} \qquad \forall k \in K, l \in N^b d, (3)$$

$$x_r \in \{0,1\} \qquad \forall k \in K, l \in N^{bd}, R \in R_c^{kl}, (4)$$

Además de las restricciones en los aviones:

$$\sum_{(i,k)\in A^a} y_{ij}^{kl} = \sum_{(i,k)\in A^a} y_{ij}^{kl} \qquad \forall k \in K, l \in N^{bd}, i \in N^f, (5)$$

$$\sum_{i\in N^{bd}, j\in N^f:(i,j)\in A^a} y_{ij}^{kl} \leq n_a^{kl} \qquad \forall k \in K, l \in N^{bd}, (6)$$

$$y_{ij}^{kl} \in \{0,1\} \qquad \forall k \in K, l \in N^{bd}, (i,j) \in A^a, (7)$$

Y restricciones en el enlace entre las rutas de las tripulaciones y las rutas de los aviones, en las conexiones cortas y en los cambios de avión:

$$\sum_{l\in N^{bd}, R\in R_c^{kl}: i\in R} x_R = \sum_{l\in N^{bd}, (i,j)\in A^a} y_{ij}^{kl} \qquad \forall k \in K, i \in N^f, (8)$$

$$\sum_{l\in N^{bd}, R\in R_c^{kl}:(i,j)\in R} x_R = \sum_{l\in N^{bd}} y_{ij}^{kl} \qquad \forall k \in K, i, j \in N^f : (i,j) \in A_s^c, (9)$$

$$z_{ij} \in \{0,1\} \qquad \forall i, j \in N^f : (i,j) \in A^c\ A_s^c, (10)$$

El modelo utiliza caminos para representar las rutas de las tripulaciones y arcos para representar las rutas de los aviones, y es denominado modelo *Arc-Path*. La denotación R_c^{kl} define al conjunto de rutas factibles de tripulación del*crew graph* $(G_c = (N, A^c))$ para una operadora $k \in K$ partiendo de una base $l \in N^{bd}$. Se entiendo como ruta factible a aquella ruta que cumpla con las restricciones en la duración de la misma, el máximo número de vuelos ejecutables y en la secuencia de vuelo.

El coste c_R viene asociado a cada una de las rutas para así luego poder ser utilizados para penalizar aquellos tiempos que sean excesivamente largos o, por lo contrario, excesivamente cortos.

Definimos una variable binaria x_R para representar las rutas. De esta manera, $x_R = 1$ siempre y cuando la ruta R sea asignada a una tripulación perteneciente a la operadora $k \in K$ que tiene como base de salida a $l \in N^{bd}$

De igual forma, se define otra variable binaria y_{ij}^{kl} para cada arco $(i,j) \in A^a$, cada operador $k \in K$ y cada base de salida $l \in N^{bd}$. De esta manera, $y_{ij}^{kl} = 1$ si y solo si el arco (i,j) es operado por un avión perteneciente a la operadora k con base de salida l.

Por último, para poder definir los cambios de un avión en una ruta de tripulación se ha definido una variable binaria z_{ij}. De esta menra, por cada arco $(i,j) \in A^c/A_s^c$, z_{ij} siempre que un cambio de avión se produzca entre los juegos i y j, siendo estos $(i,j) \in N^f$

En el modelo matemático la función objetivo (1) presenta los cuatro criterios a través de una única función. A continuación se listan las definición de algunas las distintas restricciones:

- (2) impone que cada vuelo ha de ser asignado a una tripulación restrictivamente.
- (3) delimita el número de tripulaciones disponibles para cada operador en cada base.
- (5) impone la conservación del flujo en cada nodo correspondiente a un vuelo.
- (8) enlaza las rutas de las tripulaciones con las de los aviones.
- (9) prohíbe que se produzcan cambios en el avión cuando hayan conexiones cortas.

3.3. Enfoque de la solución.

Para la resolución se basó en un modelo de programación lineal mixta *(Mixed-integer linear programming - MILP)*. El enfoque para la resolución se descompuso en tres fases. La primera parte consiste en utilizar el proceso de generación por columnas para generar las rutas de la tripulación, resolviendo así la relajación de la programación lineal del modelo *MILP*. La segunda parte consiste en dos subfases: Cota inferior y calculo de la solución óptima de la resolución del problema (Pérez Hernández et al., 2020).

3.3.1. Generación por columnas

El método de generación de columnas es un proceso muy utilizado en los problemas con un gran número de variables pero con un pequeño número de restricciones (de la Fuente García and Moreno, 1996). Esta técnica consiste en resolver problemas de programación lineal donde las columnas son impracticas generarlas explícitamente, generalmente en problemas que tienen un número exponencial de variables.

Para la resolución de este tipo de problemas se parte de un problema maestro (relajación lineal del problema original), con una estructura relativamente simple. Después

hay un sub-problema *(sub-problema de princig)* que permite identificar estar columnas adicionales que no han sido incluidas en el problema maestro para que mejoren el valor de la función objetivo.

Este método de generación por columnas es ampliamente usado en una gran cantidad de problemas: problemas de rutas de vehículos, problemas de agente viajero, y como es el caso, problema de enrutamiento de aeronaves.

El procedimiento consiste en resolverel *pricing problem* que busca la ruta con el mínimo costo reducido y corresponde a un problmea ESPRC (*Elementary Shortest Path with elementary Constraints*) (Gélvez, 2018). Este *pricing problem* es resuelto por un procedimiento ed programación dinámica. El problema consiste en determinar para cualquier operador $k \in K$ y para cualquier $l \in N^{BD}$, aquella rutas tales que cumplan que todas las restricciones descritas anteriormente sean factibles y que:

$$\alpha \cdot c_R + \beta_k - \sum_{i \in R} (\varphi_i + \zeta_i^k) - \sum_{(i,j) \in R} \eta_{ij} - \sum_{(i,j) \in R} \xi_{ij}^k - \psi^{kl} < 0$$

De la anterior formulación, φ_i se trata de las variables duales de las restricciones descritas en (2), ψ^{kl} las variables duales des las restricciones (3), ζ_i^k las variables duales de las restricciones (8), ξ_{ij}^k las variables duales de las restricciones (9) y η_{ij} las variables duales de las restricciones (10).

Capítulo 4

Aprendizaje por refuerzo.

En este proyecto se ha investigado el aprendizaje por refuerzo (Reinforcement Learning). El aprendizaje por refuerzo es un área del aprendizaje automático que se ocupa de cómo los agentes inteligentes deben realizar acciones en un entorno para maximizar la noción de recompensa acumulativa. El aprendizaje por refuerzo es uno de los tres paradigmas básicos de aprendizaje automático, junto con el aprendizaje supervisado y el aprendizaje no supervisado.

Entendiendo como aprendizaje supervisado a cuando partimos de un conjunto de datos que esta previamente etiquetado, es decir, se conoce con anterioridad el valor del atributo objetivo para el conjunto de datos que disponemos. Sin embargo, el aprendizaje no supervisado parte de datos que no están etiquetados previamente (Morales and González, 2012).

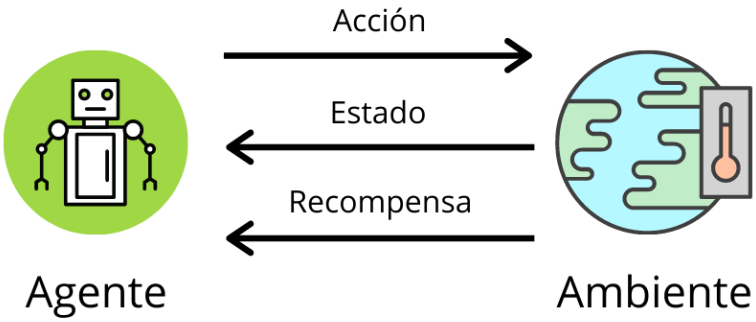

Figura 4.1: Funcionamiento de aprendizaje por refuerzo

El aprendizaje por refuerzo se estudia en diversas disciplinas, como la teoría de juegos, la teoría de control, la investigación de operaciones, la teoría de la información, la optimización basada en simulación, los sistemas multiagente, la inteligencia de enjambre y las estadísticas.

Los problemas de interés en el aprendizaje por refuerzo también han sido estudiados en la teoría del control óptimo. Esta teoría se ocupa principalmente de la existencia y caracterización de soluciones óptimas y algoritmos para su cálculo exacto, y menos del aprendizaje.

4.1. Problemas y algoritmos para el control del aprendizaje.

Este tipo de inteligencias requiere de mecanismos de exploración inteligentes. Si se utiliza una selección aleatoria de acciones, sin ningún tipo de referencia a una distribución de probabilidad estimada, esto daría resultados totalmente deficiente. Sin embargo, debido a la falta de algoritmos que se adapten bien al número de estados, los métodos de exploración simples son los más prácticos en este tipo de casos. Uno de los métodos de exploración simple más conocido y práctico es el algoritmo *Greedy*.

Incluso si se ignora el tema de la exploración e incluso si el estado fuera observable, el problema sigue siendo utilizar la experiencia pasada para descubrir qué acciones conducen a recompensas acumulativas más altas.

Existen varios caminos a lo que elección de algoritmos se refiere. Se podría implementar un algoritmo siguiendo el criterio de optimalidad o también siguiendo un enfoque de fuerza bruta. El enfoque de fuerza bruta implica dos pasos principalmente:

- Para cada una de las políticas posibles se devolverá una muestra.
- Se elige una póliza con la mayor rentabilidad esperada.

El principal problema con este tipo de algoritmos es el tiempo de ejecución, ya que se consiguen tiempo de ejecución muy elevados debido a la cantidad de pólizas elevadas. Estos problemas se pueden mejorar si asumimos algún estructura y permitimos que las muestras generadas a partir de una política influya en las estimaciones realizadas por otras. Los dos enfoques principales para conseguir esto es seguir la estimación de la función de valor y la búsqueda directa de políticas.

4.2. Algoritmo de búsqueda exhaustiva.

Una vez expuesto la teoría que supone llevar acabo el problema, es hora de ponerse con la resolución del mismo. Al llevar acabo la optimización de los vuelos, y haberlo solventado por los métodos descritos anteriormente, queda afrontar el problema a la hora de añadir pequeñas variaciones.

De los distintos algoritmos que son utilizados para el control del aprendizaje he optado por utilizar un enfoque llamado *Fuerza Bruta*. En estos enfoques el algoritmos se divide en dos etapas:

- Una primera fase para cada política posible, muestrea los resultados.
- Una segunda fase donde se devuelve aquella que obtiene una mayor política.

La búsqueda por fuerza bruta también recibe el nombre de búsqueda exhaustiva y se trata de una técnica que consiste en enumerar las distintas posibles candidatos para la solución de un problema. Después de enumerar los distintos candidatos en necesario comprobar que dicho candidato satisface la solución al problema.

Para la implementación de un algoritmo básico para la búsqueda exhaustiva es necesario implementar tres tipos de funciones distintas:

- $first(Problem)$: esta función generar una primera solución candidata para el problema.
- $next(Problem, candidate)$: esta función genera la siguiente solución candidata para el problema después de una soluciópn candidata dada.
- $valid(Problem, candidate)$: esta función se encarga de comprobar si la solución candidata generar es una solución valida para el problema.
- $show(Problem, candidate)$: esta función se encarga de mostrar la solución candidata para el problema.

Es cierto que hay secuencias en los problemas dados que no es posible encontrar una solución. Para solventar este tipo de casos y que el algoritmo no genere una primera solución no valida lo que se hizo fue que si el la función *first* no encontraba una solución al problema se devolviera un un valor nulo.

El algoritmo descrito anteriormente utiliza la función *show* para mostrar cada una de las soluciones candidatas al problema. Comúnmente, el algoritmo es modificado para que una vez encuentre una primera solución el algoritmo se detenga. La realidad es que para la implementación de una interfaz es necesario tener en cuenta el tiempo de CPU, por lo que se ha considerado una cantidad fija de este tiempo para la parada del algoritmo.

Como hemos visto, la búsqueda exhaustiva no es compleja de implementar, pero su coste de ejecución es proporcional al número de soluciones candidatas (Tello, 2018). Dicho número de soluciones es exponencialmente proporcional al tamaño del problema. Este tipo de búsquedas ha sido utilizado ya que previamente se ha reducido utilizando otro método heurístico.

4.3. Enfoque de la solución.

Como se describe en el anterior apartado, es necesario la implementación de cuatro funciones distintas. Para el problema solo ha sido necesario la implementación de tres de estas funciones. He decido descartar la función donde se verifica si es valida la solución candidata, ya que en nuestra función *first* genero directamente una solución candidata valida.

4.3.1. Función *first*.

Para la implementación de esta función se ha dividido en dos fases. Una primera fase donde se resuelve la cota inferior y una segunda fase donde se implementa el modelo MILP.

Para la implementación de la cota inferior se ha seguido el fundamento del siguiente algoritmo:

- Calculo de un conjunto de rutas iniciales.
- Relajación lineal.
- Programación dinámica.
- Repetimos los pasos 2 y 3 hasta que el coste sea positivo.

Como vemos en el fundamento, el primer paso es el calculo de la relación lineal con las rutas que se tengan en el momento.

Para la implementación del modelo MILP, que se ha definido en los anteriores apartados, se ha utilizado la librería *Pulp* para la utilización de una de sus clases, concretamente *LpProblem Class*. Esta clase genera un problema de programación lineal, y utilizando el método *lpSum* consigo calcular la suma de la lista de la expresiones lineales que están descritas en cada una de las restricciones anteriormente nombradas.

4.3.2. Función *next*.

Para la implementación de la función *next* he decido que en vez de buscar otra solución candidata, el problema tendrá determinado un tamaño de tiempo de ejecución que será directamente proporcional al tamaño de variables del problema. Este tiempo de ejecución previamente determinado será el elemento que decida cuando parar con la ejecución y determinar así si se ha encontrado la solución óptima al problema.

Para la implementación de este método lo primero es determinar el tiempo de ejecución máximo. Para ello he realizado un serie de ejecuciones donde se ha encontrado la solución óptima al problema y he comparado el número de variables en esa ejecución con el tiempo de ejecución.

Nº de variables	Tiempo de ejecución	Relación (var/s)
1674	0,8119	$4{,}8500 \cdot 10^{-4}$
3524	2,1579	$6{,}1234 \cdot 10^{-4}$
20566	67,9276	$3{,}3029 \cdot 10^{-3}$
22666	217,3488	$9{,}5891 \cdot 10^{-3}$
27105	246,3478	$9{,}0886 \cdot 10^{-3}$

Tabla 4.1: Relación variable y tiempo de ejecución.

Luego se determinó que el tiempo de ejecución máximo para el algoritmo vendría dado por:

$$t_{max.} = numero_{variables} \cdot 10^{-2}$$

En la siguiente figura podemos ver la relación del tiempo de ejecución con el número de variables mostrada en una gráfica utilizando los datos expuestos en la Tabla 4.1:

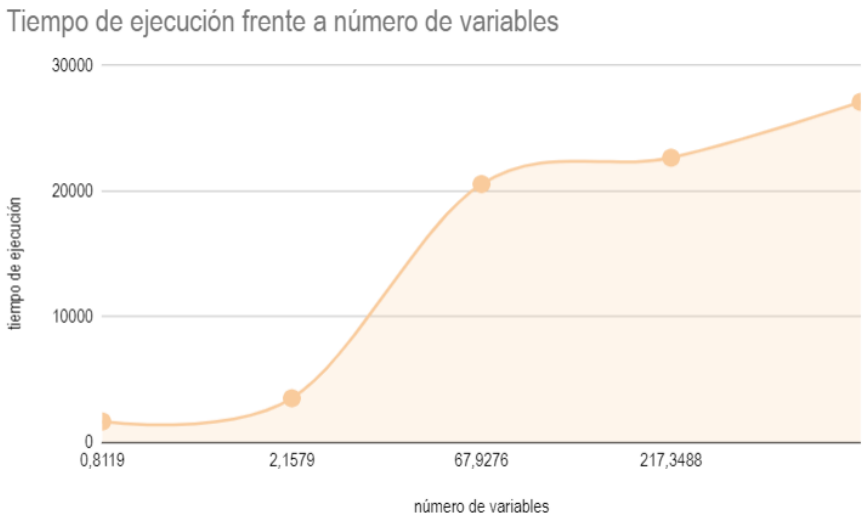

Figura 4.2: Gráfico de áreas del tiempo de ejecución frente al número de variables.

4.3.3. Función *show*.

La función *show* recordemos que mostrará la solución óptima al problema. Para la implementación de dicha función ha sido muy sencilla, ya que simplemente lo que nos devuelve la clase *LpProblem* es la solución óptima al problema en una lista de objetos.

Dicha lista de objetos que es devuelta es recorrida para contabilizar el número identificador de la ruta y posteriormente se le añade el coste y la duración de dicha ruta.

Para la salida del problema he decido dos formatos distintos de salida. El primer formato en el que se exporta la salida del programa es en un fichero de extensión JSON con la siguiente forma:

Fichero 4.1: Solución en formato JSON

```
{
    "Final Routes": [
        {
            "op-base": ["ATB","LPA"],
            "routes": [],
            "cost": 0.0,
            "duration": 0.0
        },
        {
            "op-base": ["ATB","TFN"],
            "routes": [],
            "cost": 0.0,
            "duration": 0.0
        },
        {
            "op-base": ["AC6","LPA"],
            "routes": [
                {
```

```
                    "id_routes": [200, 203, 123, 152, 516, 515, 222, 223],
                    "cost": 255.0,
                    "duration": 580.0
                },
                {
                    "id_routes": [ 500, 501],
                    "cost": 30.0,
                    "duration": 115.0
                },
                {
                    "id_routes": [ 309, 304, 210, 211, 358, 359],
                    "cost": 155.0,
                    "duration": 435.0
                },
                {
                    "id_routes": [ 107, 108, 510, 513, 280, 281],
                    "cost": 240.0,
                    "duration": 470.0
                }
            ]
        },
        {
            "op-base": ["AC6", "TFN"],
            "routes": [
                {
                    "id_routes": [605, 606, 651, 650, 110, 145, 631, 632],
                    "cost": 300.0,
                    "duration": 540.0
                }
            ]
        }
    ]
}
```


La solución en formato JSON, cuenta con cuatro diccionarios distintos. Cada diccionario cuenta con distintos atributos:

- ’op-base’: representa la pareja de operadora y base.
- ’routes’: se trata de una lista, donde se almacenan tanto diccionarios como rutas haya.
    - ’id_routes’: identificador de la ruta.
    - ’cost’: coste de la ruta.
    - ’duration’: duración de la ruta.

El segundo formato en el que se exporta la salida del programa es en un fichero con extensión excel (*.xlsx*) dedicada más a la parte del usuario. Esta salida tiene la siguiente forma:

	A	B	C	D
1	**Op-bases**	**Route**	**Cost**	**Duration**
2	('AC6', 'LPA')	[200, 203, 123, 152, 516, 515, 222, 223]	255	580
3	('AC6', 'LPA')	[500, 501]	30	115
4	('AC6', 'LPA')	[309, 304, 210, 211, 358, 359]	155	435
5	('AC6', 'LPA')	[107, 108, 510, 513, 280, 281]	240	470

Figura 4.3: Solución en formato hoja de cálculo.

La solución en formato hoja de cálculo se ha decidido implementar de tal forma que generen cuatro fichero distintos, uno por cada pareja de operadora-base (’AC6-LPA’, ’AC6-TFN’, ’ATB-LPA’, ’ATB-TFN’). En la figura 4.3 podemos ver un ejemplo de solución en dicho formato, dividido en cuatro columnas:

- ’Op-bases’: representa la pareja de operadora y base.
- ’Route’: se trata de la lista de los identificadores de la ruta.
- ’Cost’: coste de la ruta.
- ’Duration’: duración de la ruta.

De esta forma cada fichero contendrá las distintas listas de rutas asignadas a esa pareja de operadora-base, asociando cada lista de ruta a un coste y una duración.

Capítulo 5

Interfaz de Usuario.

Si es cierto que para la utilización del modelo hace falta su implementación, también es necesario que se disponga de una interfaz gráfica que sea capaz de ser utilizada por cualquier usuario. Por ello, se propone un diseño de interfaz dirigida al usuario para la utilización del modelo.

5.1. Mockup.

5.1.1. Herramienta utilizada.

Para la realización del mockup de la aplicación he utilizado Justinmind (Farrell-Vinary, 2011). Este software es una herramienta de prototipos y wireframing para la creación de prototipos de alta fidelidad de aplicaciones web y móviles. He escogido esta herramienta por su gran capacidad de generar versiones realistas de un producto terminado, además que el software permite crear funciones de colaboración, interacción y diseño.

5.1.2. Resultados obtenidos.

A continuación se expone la interfaz diseñada. En la primera figura vemos el *mockup* de la ventana principal del programa:

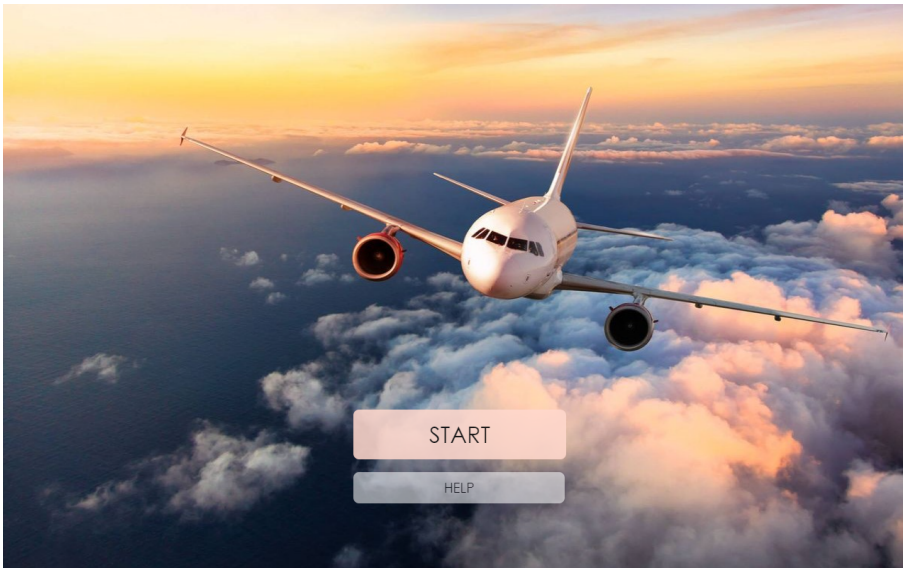

Figura 5.1: Pantalla principal de la aplicación

Una vez el usuario se encuentre en la pantalla principal del programa visualizara dos botones, *START* y *HELP*. Al pulsar uno de los dos el programa lo redirigirá a otra pantalla emergente, la primera para trabajar en el programa (*botón START*) o una segunda pantalla informativa (*botón HELP*).

La pantalla emergente informativa, como su nombre indica, informará al usuario sobre el funcionamiento básico de la aplicación.

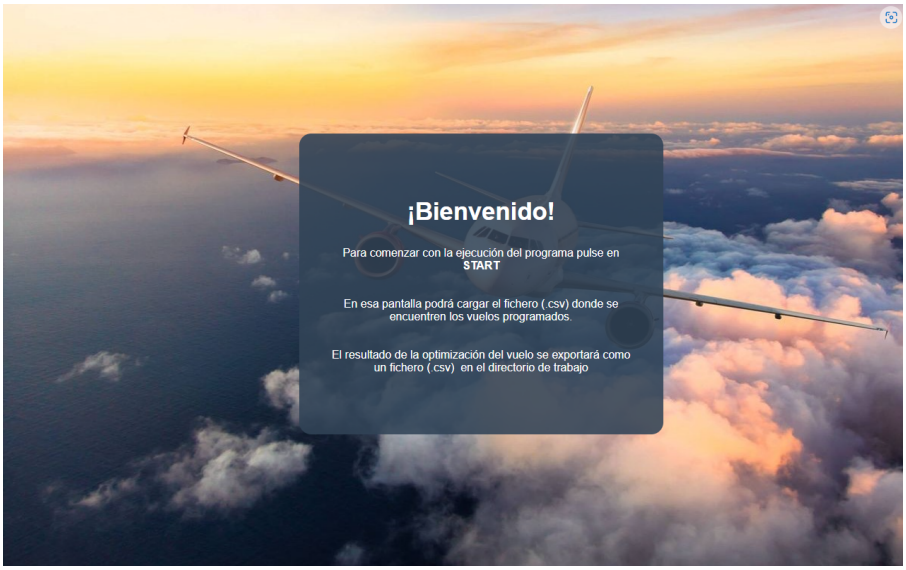

Figura 5.2: Pantalla informativa sobre el funcionamiento del programa

Cuando el usuario presione el botón *START* y se encuentre en la pantalla de desarrollo, podrá cargar el fichero donde se encuentran los vuelos. En esta pantalla, el usuario se encuentra con cuatro opciones. La primera opción y más llamativa es la central donde se encuentra un panel que el usuario puede pulsar para cargar el archivo de partida para el problema.

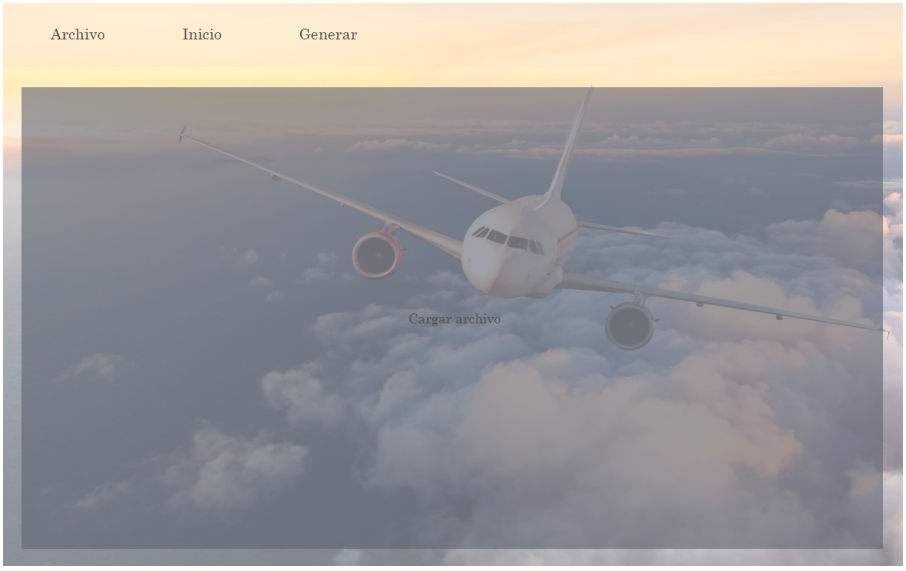

Figura 5.3: Pantalla generativa de rutas.

5.2. Interfaz gráfica.

5.2.1. Herramienta utilizada.

Para la realización de la interfaz gráfica de la aplicación he utilizado el lenguaje de programación Python. En especial he utilizado la librería PyQt5 (Meier, 2019), binding de la biblioteca gráfica QT para dicho lenguaje de programación.

5.2.2. Resultados obtenidos.

Una vez diseñado el mockup de la aplicación es momento de aplicar estas ideas a una interfaz de usuario real donde he decido que su implementación sería en inglés. En la siguiente figura vemos el resultado de la implementación de la pantalla principal de la aplicación que se había comentado en el anterior apartado.

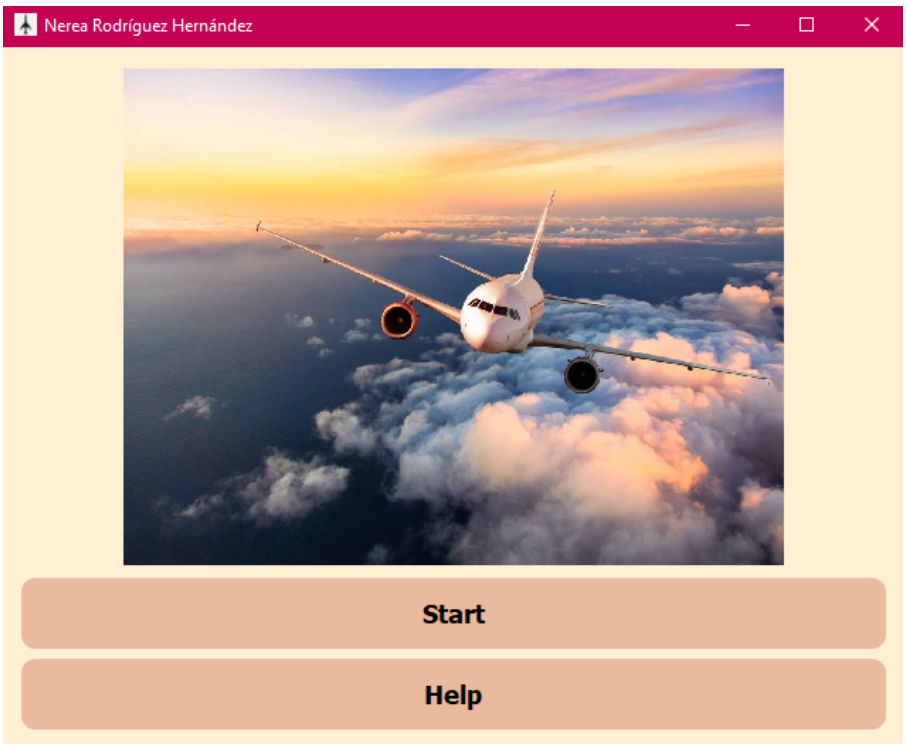

Figura 5.4: Pantalla principal real de la interfaz.

Igual que se explicaba anteriormente, cuando el usuario presione el botón 'Help', emergerá una nueva pantalla informativa. Como resultado de la implementación:

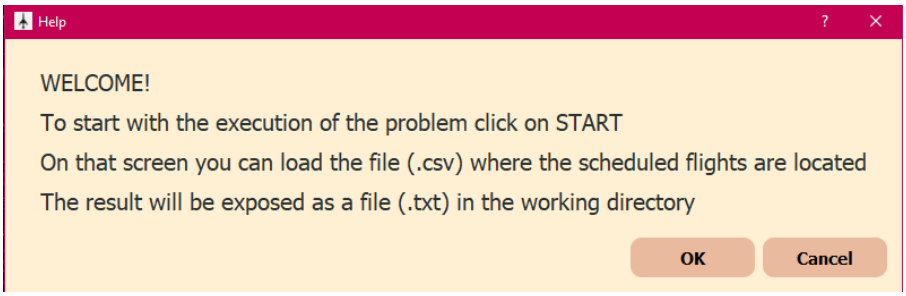

Figura 5.5: Pantalla informativa real de la interfaz.

De igual forma, cuando el usuario presione el botón 'Start', se le redirecciona a otra pantalla. Como resultado de la implementación:

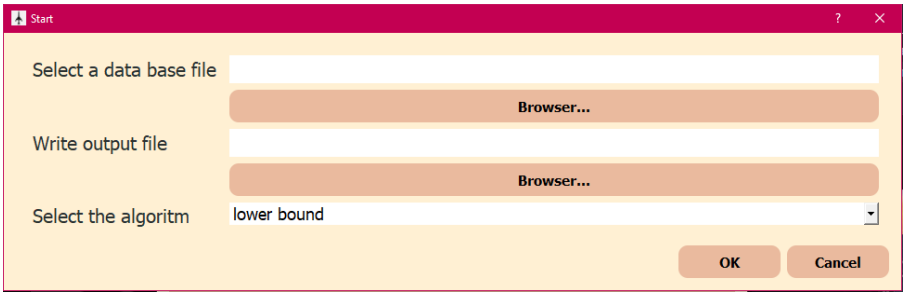

Figura 5.6: Pantalla *start* real de la interfaz.

Con respecto al mockup diseñado en una primera instancia, he decido cambiar el diseño de esta pantalla para que en ella el usuario directamente seleccione el fichero de

entrada donde se encuentran las distintas rutas. Así mismo, el usuario debe introducir el nombre del fichero de salida donde se almacenará al ruta final. Por último el usuario seleccionará el algoritmo a aplicar, añadiendo más opciones si se selecciona la opción de búsqueda exhaustiva.

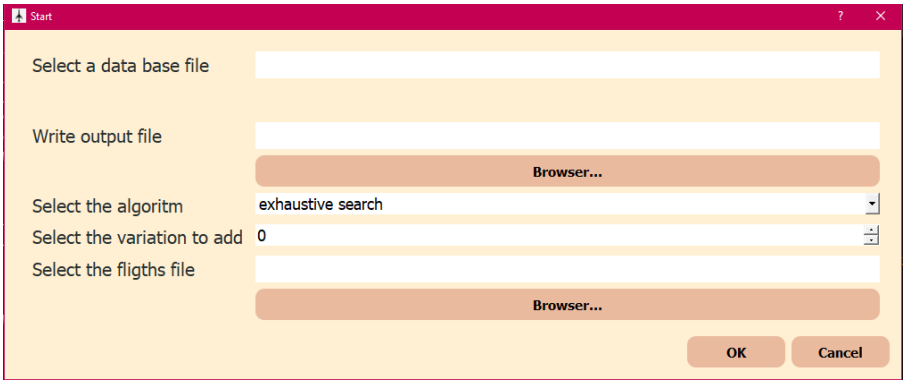

Figura 5.7: Pantalla *start* real con variaciones de la interfaz.

El usuario podrá seleccionar el número de vuelos a añadir o eliminar, seleccionando un número negativo para esto último, y por otro lado deberá introducir un fichero donde se encuentren los id de los vuelos a añadir o a eliminar.

5.2.3. Detalles de selección del algoritmo.

Como se ha expuesto en el anterior apartado 5.1.2, el usuario puede seleccionar entre tres algoritmos. Los dos primeros algoritmos se trata de una implementación que encontramos en Pérez Hernández et al. (2020), denominados cota inferior (*lower bound*) y MILP.

El algoritmo *lower bound* implementado en el trabajo de Pérez Hernández et al. (2020) se utiliza la librería *Pulp* para la utilización de sus clases, concretamente *LpProblem Class*. La clase *LpProblem Class* nos permite resolver problemas lineales, dicha implementación cuenta con una metodología nada simple. Este algoritmo es definido siguiendo el siguiente fundamento:

Algoritmo 1: Lower Bound

1 $routes = []$
2 $objetive_{value} = -1$
3 $problem \leftarrow LpProblema('LinearRelaxation', LpMinimize)$
4 $problem \leftarrow add\ restrictions$
5 $problem.solve()$
6 $optimizable \leftarrow funtion\_check\_optimizable$
7 $routes \leftarrow problem_{variables}$
8 $objetive_{value} \leftarrow value(problem.objetive)$

Por otro lado, el algoritmo MILP implementado en el trabajo de Pérez Hernández et al. (2020) también utiliza la librería *Pulp* para la utilización de la clase *LpProblem Class*. Cabe recordar que el modelo MILP se trataba de un modelo matemático conocido como

problema lineal mixto (MILP) del inglés *"Mixed Integer Linear Problems (MILP)"*. En este algoritmo se sigue el siguiente fundamento:

Algoritmo 2: Lower Bound

1 $routes = []$
2 $objetive_{value} = -1$
3 $problem \leftarrow LpProblema('UpperBound', LpMinimize)$
4 $problem \leftarrow add\ restrictions$
5 $problem.solve()$
6 $optimizable \leftarrow funtion\_check\_optimizable$
7 $routes \leftarrow problem_{variables}$
8 $objetive_{value} \leftarrow value(problem.objetive)$

Por último, la última elección del usuario es el algoritmo de búsqueda por fuerza bruta (*exhaustive search*). Para la implementación del mismo se ha seguido el fundamento descripto en el Apéndice A.2

Capítulo 6

Evaluación y resultados.

En este capitulo se expone el estudio de los datos obtenidos en el presente trabajo, así como un análisis de los resultados. Primero expongo los resultados de la optimización de los vuelos para poder compararlos con los resultados de las otras variaciones.

6.1. Resultado de la optimización.

En este apartado expongo los resultados obtenidos de los datos que corresponden con los vuelos ofertados por la compañía el 15 de marzo de 2020. Estos datos son reales obtenidos por una aerolínea regional establecida en las Islas Canarias.

inst.	Nº vuelos	Nº conex. cortas	Nº conex. largas
1	30	5	85
2	66	8	249
3	70	13	282
4	80	13	328
5	100	21	503
6	128	24	742
7	140	27	844
8	152	30	946

Tabla 6.1: Relación número de vuelos y conexiones.

En la siguiente gráfica podemos ver la relación entre el número de vuelos y el total de conexiones realizadas (conexiones cortas y conexiones largas) utilizando los datos expuestos en la Tabla 6.1:

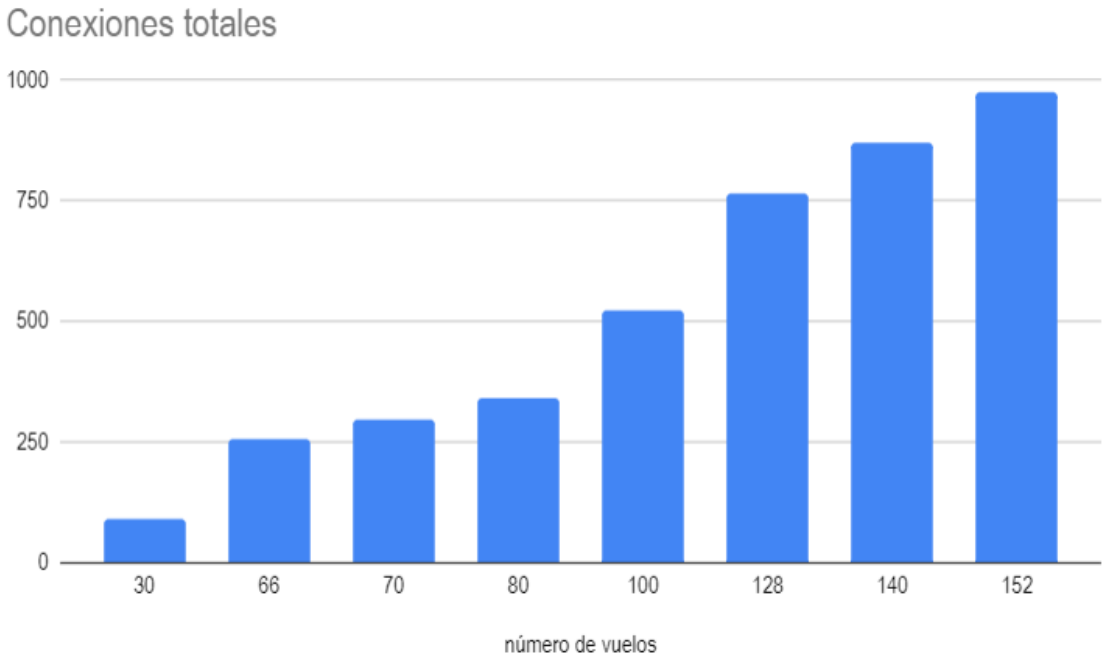

Figura 6.1: Relación del número de vuelos y conexiones totales.

Para estas instancias se han considerado los siguientes valores para los pesos en la duración de los tiempos de conexión entre las rutas:

- 2 para los tiempos de conexión entre 20 y 30 minutos.
- 1 para los tiempos de conexión entre 30 minutos y una hora.
- 3 para los tiempos de conexión entre 60 minutos y 120 minutos.
- 5 para los tiempo de conexión entre 120 y 180 minutos.

Para las instancias se han considerado los siguientes valores para los pesos de la función objetivo:

- λ con valor 1.
- β_k con valor 1000 para $k \in K$.
- γ_k con valor 10 para $k \in K$.
- δ con valor 100.

Al final el objetivo que perseguimos es minimizar el número de tripulaciones utilizadas para cubrir todos los vuelos, a pesar que también es importante conseguir minimizar los cambios de avión, los tiempos de conexión y el número de aviones usados para cubrir todos los vuelos.

Por otro lado, para cada un de las instancias anteriores, nos devuelve información sobre el número de variables de tripulación y de avión durante el proceso de resolución del problema.

Esta información que nos devuelve viene dada por:

- El número de variables de tripulación generadas $(x_R LP)$.
- El número de variables de aviones generadas $(y_a LP)$.

inst.	Nº vuelos	$x_R LP$	$y_a LP$
1	30	139	484
2	66	427	1936
3	70	974	1992
4	80	766	2732
5	100	880	4268
6	128	944	6524
7	140	1072	7628
8	152	972	8856

Tabla 6.2: Información sobre número de tripulación y aviones.

En la siguiente gráfica de barra apiladas podemos ver la relación entre el número de variables de tripulación y el número de variables de avión generadas con los datos expuestos en la Tabla 6.2:

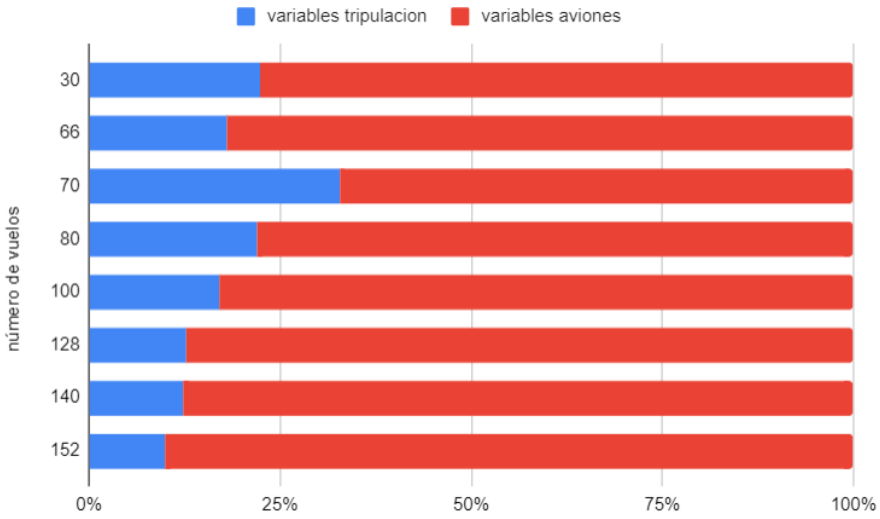

Figura 6.2: Relación del número de vuelos y conexiones totales.

Por último, se ha comparado como cambia el rendimiento del algoritmo según cada instancia de prueba. En esta tabla tenemos los tributos:

- El número de vuelo de la instancia de prueba *(Nº Vuelos)*.
- El valor objetivo en la función*first (first)*
- El tiempo de ejecución de la función *(T_{first})*.
- El tiempo de ejecución del modelo completo *(T_{total})*

instancia	Nº vuelos	T_{first}	first	T_{total}
1	30	2,03062	6030	2.29840
2	70	189.18620	13310	90.9369
3	78	189.18620	15340	76.38654
4	80	66.12619	16295	68.17263

Tabla 6.3: Rendimiento algoritmos de cada instancia.

En el anterior análisis se pueden ver los resultados de un mismo día con cantidades diferentes en el número de vuelos. A continuación se realiza un estudio de los distintos vuelos en cada día de una semana de un mismo mes y año. En el siguiente análisis se mantuvieron los pesos expuestos anteriormente.

El estudio fue realizado de la primera semana sobre el mes de septiembre de 2012:

inst.	Día	Nº vuelos	Nº conex. cortas	Nº conex. largas
1	01/09/12	100	18	452
2	02/09/12	130	32	764
3	03/09/12	110	28	596
4	04/09/12	119	38	629
5	05/09/12	121	40	628
6	06/09/12	124	39	642
7	07/09/12	140	52	945

Tabla 6.4: Relación conexiones en una semana.

En la siguiente gráfica de barra podemos ver la relación entre el número de vuelos y el número de conexiones totales que realiza en la primera semana de septiembre de 2012, datos expuestos en la Tabla 6.4:

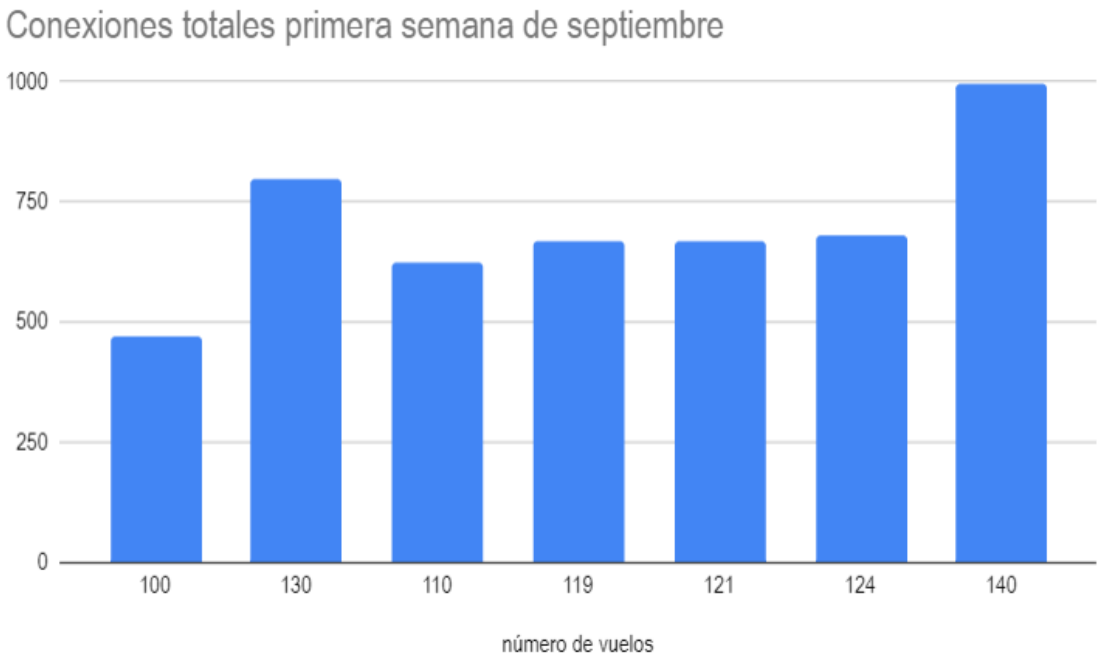

Figura 6.3: Relación del número de vuelos y conexiones totales en el mes de septiembre.

De la gráfica que vemos en la Figura 6.3 podemos sacar que existe una distribución homogénea entre el número de vuelos con el número de conexiones totales y que en una semana aproximadamente se realizan 700 conexiones entre vuelos.

Por último expongo la siguiente información sobre el número de variables de tripulación y de avión que se generan durante el proceso de resolución del problema:

inst.	Día	Nº vuelos	$x_R LP$	$y_a LP$
1	01/09/12	100	735	4092
2	02/09/12	130	947	6736
3	03/09/12	110	792	4891
4	04/09/12	119	532	5940
5	05/09/12	121	585	5940
6	06/09/12	124	1698	6212
7	07/09/12	140	1858	7564

Tabla 6.5: Información sobre número de tripulación y aviones.

6.2. Comparativa de resultados.

En este apartado expongo los resultado expuestos en Pérez Hernández et al. (2020) y realizo una comparativa con los resultados obtenidos en el presente trabajo. Es necesario mencionar que las pruebas que se realizaron en el trabajo de Pérez Hernández et al. (2020) en un portátil de uso personal con procesador i7-8550 y una memoria RAM de 8GB, mientras que la se realizaron en este trabajo también fueron realizadas en un portátil de uso personas con procesador i7-7700HQ y una memoria RAM de 8GB.

En la siguiente tabla se muestran los datos obtenidos por los distintos algoritmos. A continuación la leyenda de la tabla:

- inst.: identificador de la instancia.
- Nº vuelos: número de vuelos que se han utilizado en la prueba de ejecución.
- T_{LB}: tiempo de ejecución del algoritmo *Lower Bound*.
- LB: valor objetivo del problema utilizando el algoritmo *Lower Bound*.
- T_{MILP}: tiempo de ejecución del algoritmo *MILP*.
- MILP: valor objetivo del problema utilizando el algoritmo *MILP*.
- T_{total}: tiempo de ejecución total.
- first: valor objetivo del problema utilizando el algoritmo *First*.

inst.	Nº vuelos	T_{LB}	LB	T_{MILP}	MILP	T_{total}	first
1	30	0.79823	6015	0.07627	6015	2.29840	6030
2	70	142.02474	13310	0.46920	13310	90.93692	13310
3	78	182.99210	15435	1.45914	15445	76.38654	15340
4	80	160.66826	16235	1.88724	16375	68.17263	16295

Tabla 6.6: Comparación de rendimientos de algoritmos.

En la siguiente gráfica en la figura 6.4 vemos de manera visual la comparación de rendimiento de algoritmos con el valor objetivo de los distintos algoritmos:

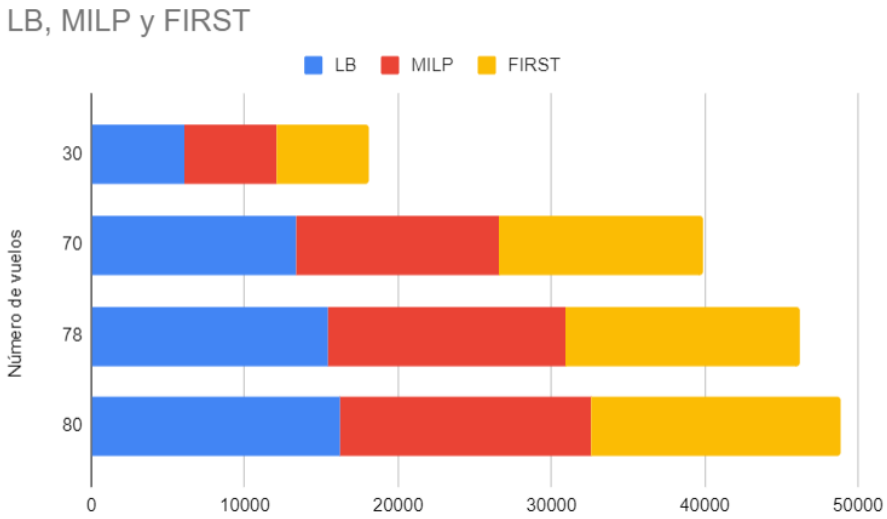

Figura 6.4: Comparación valores objetivo de los algoritmos.

De igual manera, se expone otra gráfica en la figura 6.5 donde podemos comparar los tiempos de ejecución:

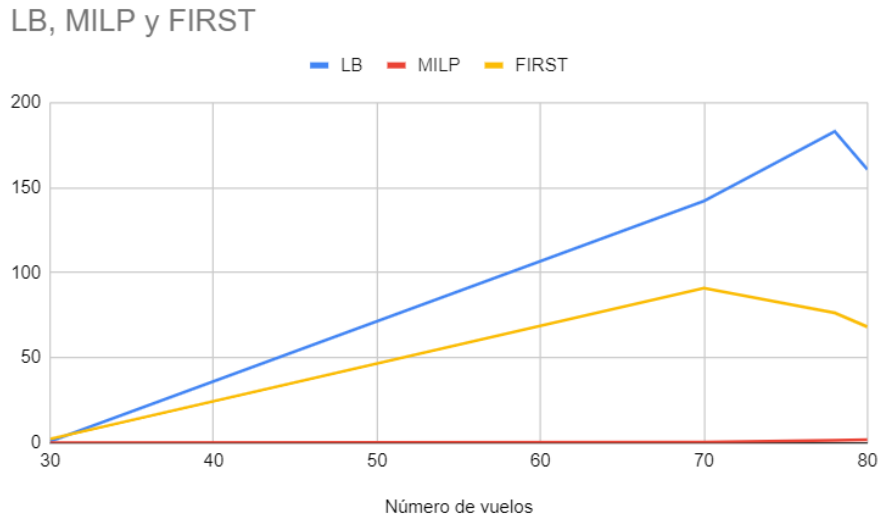

Figura 6.5: Comparación tiempos de ejecución de los algoritmos.

Capítulo 7

Conclusiones y líneas futuras.

En este apartado expone mi autocrítica y valoración, además de presentar una visión para la mejora de la herramienta desarrollada.

Este proyecto ha consistido en la investigación de problemas de optimización vinculados a la programación de vuelos. Además de analizar los problemas, hemos utilizado elementos de Inteligencia Artificial. Dada la importancia para cualquier aerolínea, existe mucha documentación para muchas variantes. Dado su auge en el mercado, es un problema vivo, que plantea muchos desafíos, y sobre el que se continua realizando investigación de alto nivel.

Desde el comienzo del proyecto se marcaron una serie de objetivos que llevarían a una meta. Dicha meta desde un punto de vista objetivo ha sido alcanzada. Se ha conseguido investigar y desarrollar una herramienta útil en el ámbito de la optimización de rutas de vuelos.

Desde un punto de vista computacional, en el desarrollo del algoritmo se trató llegar al mejor tiempo de computo, pero desafortunadamente no se llegó a mejorar con diferencia. Como línea de futuro en esta rama del problema estudiaría diferentes compiladores y/o lenguajes de programación que consiguieran un mejor tiempo de computo.

Desde un punto de vista dirigida al usuario, la herramienta gráfica desarrollada presenta una herramienta bastante básica e intuitiva. Como línea de futuro en esta rama del problema añadiría valor al diseño dirigido al usuario con una interfaz más amigable.

A nivel personal ha sido complejo partir de un proyecto ya anteriormente desarrollado, debido a la dificultad añadida de código no desarrollado por mi. Me siento realizada por haber podido estudiar esta rama de la informática, y haberme sumergido en los problemas de enrutamiento de aeronaves y emparejamiento de tripulación, de los que espero seguir investigando en un futuro no muy lejano.

Capítulo 8

Summary and Conclusions.

In this section, I will make a self-criticism and assessment, as well as a vision for the improvement of the developed tool.

This project has consisted of the investigation of flight scheduling problems, as well as the use of Artificial Intelligence. Currently, it is a subject that continues to be studied and for which there is a great deal of documentation. Due to its market growth, research is still being carried out on its realisation.

From the beginning of the project, a number of objectives were set that would lead to a goal. From an objective point of view, this goal has been achieved. A useful tool in the field of flight route optimisation has been researched and developed.

From a computational point of view, in the development of the algorithm we tried to reach the best computational time, but unfortunately we did not improve it by far. As a future direction in this branch of the problem, I would study different compilers and/or programming languages to achieve better computational time.

From a user-oriented point of view, the graphical tool developed presents a fairly basic and intuitive tool. As a future line in this branch of the problem it would add value to the user-driven design with a more user-friendly interface.

On a personal level it has been complex to start from a previously developed project, due to the added difficulty of code not developed by me. I feel fulfilled to have been able to study this branch of computer science, and to have immersed myself in the problems of aircraft routing and crew pairing, which I hope to continue researching in the not too distant future.

Capítulo 9

Presupuesto

En el trabajo redactado en el presente trabajo ha sido necesario invertir un total de 300 horas de trabajo. El salario medio por hora de un perfil técnico como el nuestro es de 16.60€ por hora. Además de las horas de trabajo, en el proyecto se ha utilizado un ordenador personal valorado en 2657,12€.

9.1. Detalles del presupuesto.

Recurso	Descripción	Cantidad	Coste
Jornada	Horas de trabajo	300	16,60€
Ordenador personal	Lenovo Legion Y520	1	2657,12€
		Coste total:	7637,12€

Tabla 9.1: Coste de Infraestructura y recursos humanos.

Apéndice A

Algoritmos implementados

A.1. Búsqueda exhaustiva

```
/*****************************************************************************
*
* Fichero main.py
*
*****************************************************************************
*
* AUTOR: Nerea Rodríguez Hernández
*
* FECHA: 21/06/2022
*
* DESCRIPCION: Encargada de generar y mostrar la solución del problema.
*
*****************************************************************************/
```

Algoritmo 3: Búsqueda

```
1 t_estimate ← n_variables · 10^-2
2 while t_max do
3 |   candidate ← first
4 |   t_max ← next(t_estimate)
5 |   show(Problem, candidate)
```

A.2. Función first

```
/*****************************************************************************************
 *
 * Fichero exhaustive_search.py
 *
 *****************************************************************************************
 *
 * AUTORA: Nerea Rodríguez Hernández
 *
 * FECHA: 21/06/2022
 *
 * DESCRIPCION: Encargada de generar la primera solución al problema
 *
 *****************************************************************************************/
```

Algoritmo 4: First

```
problem ← None
optimizable ← True
value_objetive ← 0
while optimizable == True do
    problem ← solve problem
    optimizable, routes ← solve small problem
    for key, routes ∈ route_list do
        for value ∈ routes do
            routes_real ← reassigment value
problem ← solve lineal problem minimizing
```

A.3. Función next

```
/************************************************************************** ************
 *
 * Fichero exhaustive_search.py
 *
 ********************************************************************************************
 *
 * AUTORA: Nerea Rodríguez Hernández
 *
 * FECHA: 21/06/2022
 *
 * DESCRIPCION: Encargada de establecer la condición de parada al algoritmo.
 *
 ********************************************************************************************/
```

Algoritmo 5: Next

```
1 t_estimate ← n_variables · 10^-2
2 t_START ← execution start time
3 t_NEXT ← execution total time
4 if t_estimate >= t_NEXT then
5 |   return true
```

A.4. Función show

```
/************************************************************************** *************
 *
 * Fichero exhaustive_search.py
 *
 ********************************************************************************************
 *
 * AUTORA: Nerea Rodríguez Hernández
 *
 * FECHA: 21/06/2022
 *
 * DESCRIPCION: Encargada de mostrar al usuario la solución al problema.
 *
 ********************************************************************************************/
```

Algoritmo 6: Show

```
1  id_routes ← []
2  for value ∈ variableofproblem do
3  |  if variable_name crewroutesvalue! = −1 then
4  |  |  id_routes insertvalue
5  aux ← 0
6  for op ∈ operator_list do
7  |  for base ∈ bases_list do
8  |  |  for route ∈ route_list do
9  |  |  |  if aux ∈ id_routes then
10 |  |  |  |  newRoutes[op, base]insertroute
11 |  |  |  aux+ = 1
12 exportdataasfile
```

Bibliografía

J.-J. Salazar-González, "Approaches to solve the fleet-assignment, aircraft-routing, crew-pairing and crew-rostering problems of a regional carrier," *Omega*, vol. 43, pp. 71–82, 2014.

J.-F. Cordeau, G. Stojković, F. Soumis, and J. Desrosiers, "Benders decomposition for simultaneous aircraft routing and crew scheduling," *Transportation science*, vol. 35, no. 4, pp. 375–388, 2001.

B. Meier, *Python GUI Programming Cookbook: Develop functional and responsive user interfaces with tkinter and PyQt5*. Packt Publishing Ltd, 2019.

M. Mirjafari, A. Rashidi Komijan, and A. Shoja, "An integrated model for aircraft routing and crew scheduling: Lagrangian relaxation and metaheuristic algorithm," *WPOM-Working Papers on Operations Management*, vol. 11, no. 1, pp. 25–38, 2020.

C. L. R. Gélvez, "Diseño de un algoritmo de generación de columnas para la programación de tripulación en logística aeroportuaria." Ph.D. dissertation, Universidad Industrial de Santander, 2018.

V. Cacchiani and J.-J. Salazar-González, "Heuristic approaches for flight retiming in an integrated airline scheduling problem of a regional carrier," *Omega*, vol. 91, p. 102028, 2020.

E. Pérez Hernández *et al.*, "Procedimiento informático para la planificación óptima de tripulaciones y aviones que cubran unos vuelos dados," 2020.

D. de la Fuente García and P. P. Moreno, *Programación lineal entera y programación no lineal*. Universidad de Oviedo, 1996.

E. Morales and J. González, "Aprendizaje por refuerzo," *Presentacion En Linea en: https://ccc. inaoep. mx/˜ emorales/Cursos/Aprendizaje2/Acetatos/refuerzo. pdf*, 2012.

E. R. Tello, "Algoritmos de búsqueda exhaustiva," 2018.

P. Farrell-Vinary, "Justinmind," *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 3, pp. 34–35, 2011.