Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Aplicación para la planificación de encuestas FRONTUR-CANARIAS del ISTAC

ISTAC'S FRONTUR-CANARIAS surveys scheduling application

Borja Guanche Sicilia

La Laguna, 8 de julio de 2022

D. **Juan José Salazar González**, con N.I.F. 43356435D, Catedrático de Universidad y Director del Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna, como tutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Aplicación para la planificación de encuestas FRONTUR-CANARIAS del ISTAC"

ha sido realizada bajo su dirección por D. **Borja Guanche Sicilia**,

con N.I.F. 79.087.185-Y.

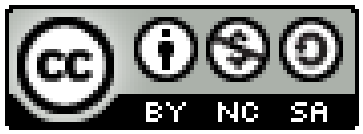
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 8 de julio de 2022

Agradecimientos

A D. Juan José Salazar González por su la labor realizada como tutor, su dedicación y sus consejos.

A todas las personas que me han acompañado y apoyado en estos cuatro años, en especial a mis familiares y amigos.

Licencia



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

Resumen

El Instituto Canario de Estadística (ISTAC) realiza mensualmente en los aeropuertos canarios la encuesta FRONTUR-Canarias. Dicha encuesta es realizada a aquellos pasajeros no residentes en el archipiélago canario, ya sea que residen en el extranjero o en otra comunidad autónoma, y que tienen como origen un aeropuerto de las Islas Canarias. La encuesta busca estudiar los movimientos en las fronteras de Canarias, con el fin de conocer datos, como por ejemplo: la edad, su procedencia, el tipo de alojamiento, cuánto han gastado en restaurantes, o si ha visitado otra isla. Se busca que dichas encuestas sean lo más representativas posibles, por lo que se tiene que maximizar el número de encuestas a realizar, teniendo en cuenta que se deben incluir a personas de todos los destinos posibles que visitan las islas. Además, se pretende aprovechar al máximo los recursos humanos, es decir, los encuestadores que el ISTAC dedique a efectuar las encuestas.

El objetivo de este trabajo fin de grado es realizar una aplicación que permita al ISTAC poder planificar con antelación y de manera óptima las encuestas FRONTUR-Canarias que se han de realizar, de manera que se cumplan los requisitos de obtención de un número de encuestas para un origen determinado. Se pretende que a dicha aplicación se le facilite la información necesaria (parámetros) para ser introducirlos en el modelo matemático y obtener así la solución óptima.

Palabras clave: ISTAC, Python, Programación Lineal, optimización, ETL, solucionador

Abstract

The Instituto Canario de Estadística (ISTAC) carries out at Canary Islands airports the FRONTUR-Canarias survey monthly. This survey is conducted to those passengers who are not residents at the Canary Islands, whether because they live in a foreign country or in another autonomous community, and they have a Canary Islands airport as their origin. This survey is looking for study the movements at Canary Islands borders, with the purpose of knowing different data about passengers, for example: the age, where are they from, the kind of accommodation, how much money they have spent at restaurants, or if they have visited another island. This survey should be representative, so the number of surveys should be maximized and this survey should include passengers from all destinations as possible. In addition, ISTAC's employees must be optimized.

The objective of this project is to make an application that ISTAC can use to scheduling the surveys of FRONTUR-Canarias, where the results must be optimized. Besides, the result must get the minimum number of survey necessary from each origin. To get the optimum solution, the application needs some inputs to being introduced on the mathematical model.

Keywords: ISTAC, Python, Linear Programming, optimization, ETL, solver

Índice general

Índice de figuras

Índice de tablas

Capítulo 1 **Descripción del problema**

En este primer capítulo, se describe en profundidad el problema a abordar, que a grandes rasgos consiste en la planificación óptima de la Encuesta de Movimientos Turísticos en Frontera de Canarias (FRONTUR-Canarias). Con este capítulo se busca dar a conocer los entresijos del problema y ofrecer así una visión panorámica de este.

1.1 Antecedentes

El Instituto Nacional de Estadística (INE) desde el año 2015 es el responsable a nivel nacional de la operación de estadística de Movimientos Turísticos en Fronteras (FRONTUR). Desde ese mismo año, el Instituto Canario de Estadística (ISTAC), tiene un convenio de colaboración con el propio INE, mediante el cual se busca ampliar la muestra de FRONTUR en el archipiélago canario, de manera que la muestra total para dicha región queda constituida por la que realice el INE y una muestra complementaria que lleva a cabo el ISTAC. Cabe destacar que las muestras son realizadas de manera separadas y debido a esto, el INE remite al ISTAC la muestra que va a realizar con el fin de que se integre con la muestra que realice el ISTAC, de manera que queden complementadas.

Actualmente, el proceso de selección de los vuelos a encuestar lo lleva a cabo una empresa subcontratada por el ISTAC. Cabe mencionar que no se tiene constancia de que se esté empleado un modelo de optimización, con el fin de optimizar algunos de los recursos (v.gr maximizar el número de encuestas a realizar, minimizar el número de encuestadores/as a emplear, etc.).

1.2 Introducción al problema

A grandes rasgos, el objetivo de la encuesta FRONTUR-CANARIAS consiste en obtener datos del turismo entrante en las islas de: *Lanzarote, Fuerteventura, Gran Canaria, Tenerife y La Palma* [1]. Este objetivo se consigue de acuerdo con las siguientes premisas:

- La estimación del número de turistas procedentes del extranjero que entran en cada una de las islas.
- La estimación del número de turistas procedentes del territorio español que entran en cada una de las islas.
- La estimación del número de excursionistas que entran en cada una de las islas.

La encuesta se realiza una vez al mes y será efectuada a turistas que tengan como origen un aeropuerto canario y no sean residentes en el propio archipiélago [2].

La empresa Aeropuertos Españoles y Navegación Aérea (AENA) se encarga de decidir qué semana se realiza la encuesta, así como de facilitar con antelación el fichero GESLOT, el cual contiene la información que se muestra en la **Figura 1.1** (v.gr los números de vuelo, las fechas de operación, los orígenes y destinos, etc.).

		Destino	Codigo	Dia_semana	Opera_desd	Opera_hasta	Hora_Salida	Aeronave	Num_vuelo	Pais	Escala	Origen
W21	Destino	ALICANTE-EL	ALC	L	01/11/2021	21/03/2022	9:05	73H	RYR4696	ESPAÑA		ACE
W21	Destino	ALICANTE-EL	ALC	V	31/12/2021	25/03/2022	18:25	73H	RYR4696	ESPAÑA		ACE
W21	Destino	ALICANTE-EL	ALC	S	05/03/2022	05/03/2022	19:50	320	VLG3169	ESPAÑA		ACE
W21	Destino	AMSTERDAM	AMS	S	08/01/2022	26/03/2022	10:30	73H	TRA5684	HOLANDA		ACE
W21	Destino	AMSTERDAM	AMS	M	02/11/2021	22/03/2022	10:50	73H	TRA5686	HOLANDA		ACE
W21	Destino	AMSTERDAM	AMS	J	17/02/2022	24/03/2022	10:50	73H	TRA5686	HOLANDA		ACE
W21	Destino	AMSTERDAM	AMS	D	13/03/2022	13/03/2022	11:05	7M8	TFL686	HOLANDA		ACE
W21	Destino	AMSTERDAM	AMS	V	18/02/2022	25/03/2022	11:30	320	VLG8443	HOLANDA		ACE
W21	Destino	AMSTERDAM	AMS	J	03/03/2022	24/03/2022	11:45	320	EJU7270	HOLANDA		ACE
W21	Destino	AMSTERDAM	AMS	S	06/11/2021	26/03/2022	12:00	320	EJU7270	HOLANDA		ACE

Figura 1.1: Fichero GESLOT AENA

1.3 Diseño muestral

El número de encuestas que se requiere para cada destino viene definido en función del número de pasajeros total que haya en la semana indicada para cada uno de los destinos. Además, el total de encuestas que se realice debe de cumplir con los tamaños mínimos muestrales que aparecen en la **Tabla 1.1**.

Calcular el número de pasajeros que habrá para un destino es complicado, dado que el número real de pasajeros no se conoce hasta que el avión ya está embarcado, de manera que se tiene que realizar una estimación. Dicha estimación se realiza siguiendo los siguientes pasos:

1. De acuerdo con los datos que ofrece el fichero GESLOT, para cada región de destino se podrá obtener el máximo de pasajeros que habrá, dado que para cada vuelo se incluye el modelo de aeronave que se empleará.
2. Teniendo los modelos de aviones que se van a emplear, como se puede ver en el apéndice **9.1 Tabla modelos de aviones**, para cada modelo de avión se tiene un número de asientos (que viene definido por el propio fabricante), por lo que de manera sencilla se puede calcular el número máximo de pasajeros (teniendo la ocupación al 100%) que podrá haber para una región la semana que se realice la encuesta.
3. Dado que no se puede suponer que los aviones irán al máximo de su capacidad, se realiza una estimación la cual como se indica en la sección **3.1 Parámetros**, se corresponde con uno de los parámetros que debe de introducir el/la técnico/a del ISTAC.
4. Además, existe otro parámetro con el cual se estiman las personas que participarán en la encuesta.

Número de pasajeros al destino	*Tamaño muestral mínimo*
Menos de 60	20
De 60 a 999	80
De 1000 a 9999	200
De 10000 a 24999	400
De 25000 a 39999	500
De 40000 a 59000	600
60000 y más	700

Tabla 1.1: Tamaños muestrales mínimos

A continuación, se muestran las dos situaciones en las que no se logra alcanzar el tamaño mínimo muestral:

- Dado unos parámetros de entrada determinados, la solución que aporta el modelo matemático no logra obtener el mínimo de encuestas requeridas, ya sea en una parte o en su totalidad para una cierta región de destino.
- A la hora de realizar las encuestas en los aeropuertos, las estimaciones tanto de ocupación como de éxito que se han realizado son inferiores al total de pasajeros que accede a realizar la encuesta, por lo que no se pueden obtener las encuestas necesarias.

En cualquiera de los casos anteriores y con el fin de lograr obtener el tamaño mínimo muestral que recoge la **Tabla 1.1**, se recurre a registros donantes. Estos registros donantes son datos que se tienen de encuestas realizadas con anterioridad, cuya procedencia se puede obtener de distintas maneras según recoge el *Informe de metodología para la estadística de movimientos turísticos del ISTAC* [2].

1.4 Alcance

Dada la casuística que presenta el problema, el objetivo del presente trabajo es diseñar una aplicación que facilite la toma de decisiones sobre qué vuelos encuestar en el ámbito de FRONTUR-CANARIAS. Esta aplicación debe de contener una interfaz gráfica intuitiva y fácil de usar, además de tener implementado el modelo matemático de optimización que se encarga de decidir a qué vuelos encuestar y a qué encuestadores emplear. Asimismo, se debe de dar la posibilidad de poder emplear un solucionador de pago y otro gratuito para resolver el problema dado.

1.5 Fases y desarrollo del proyecto

A continuación, se presenta un diagrama de Gantt donde se recogen todas las fases en las que se ha dividido el proyecto, así como la duración que estas han tenido:

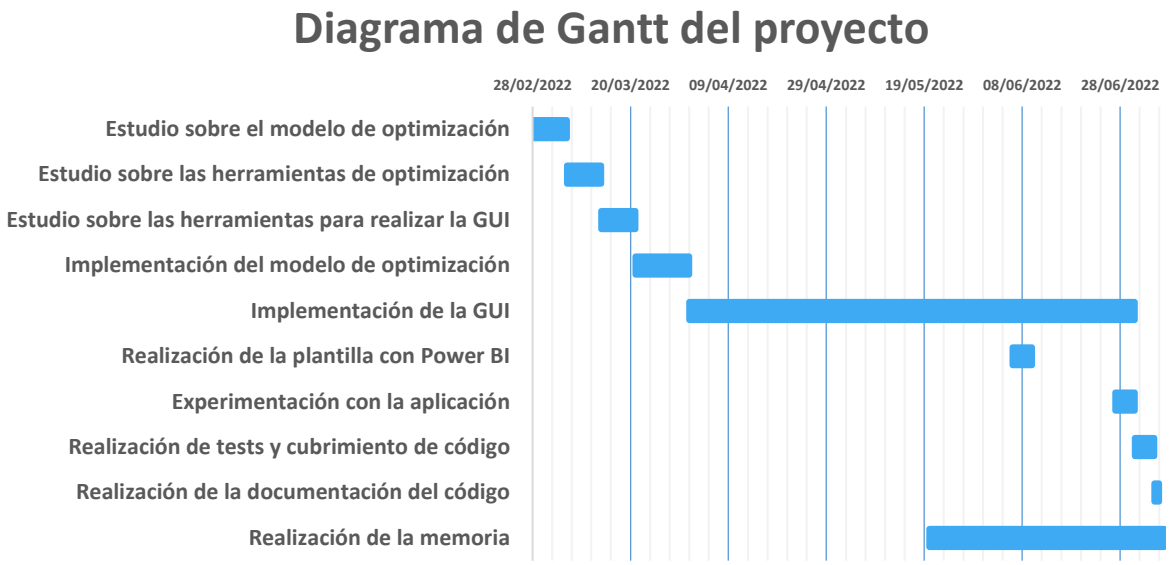


Figura 1.2: Diagrama de Gantt

Capítulo 2 **Conceptos matemáticos**

A modo de puesta en contexto de este proyecto, se comentarán los principales conceptos matemáticos que subyacen detrás de los modelos matemáticos de optimización, que no es más que la Programación Matemática, y que es la base sobre la que se sustenta el presente trabajo.

2.1 Programación Matemática

Los modelos matemáticos de optimización se emplean para representar un sistema real, donde dado un problema se busca optimizar un recurso. Para elaborar un modelo es necesario ejecutar las fases de: *análisis, formulación, solución, validación* y *puesta en práctica*. Además, todos los modelos matemáticos de optimización están formados por los siguientes tres elementos:

- **Conjunto de variables:** Son valores numéricos que se conocen e indican las alternativas que lleva consigo el problema.
- **Función objetivo:** Indica aquel recurso que se quiera *maximizar* o *minimizar*.
- **Conjunto de restricciones:** Son los elementos (ecuaciones y/o inecuaciones) que rigen el comportamiento del modelo e indican las limitaciones de los recursos que conlleva el problema.

Cuando se busca solución a un modelo de optimización, se exploran aquellos valores de las variables que cumplan todas las restricciones del modelo y que obtengan un valor óptimo para la función objetivo. En base a este procedimiento, el problema puede tener las siguientes características:

- **Problema no factible:** El espacio de soluciones está vacío, por lo tanto, no existe una solución. Esto indica que ninguna variable verifica todas las restricciones.
- **Problema factible:** En el espacio de soluciones sí existen soluciones que verifiquen todas las restricciones, sin embargo, con respecto a la búsqueda del valor óptimo se pueden dar dos situaciones:
  - **Problema no acotado:** La solución óptima se encuentra en el infinito porque en el espacio de soluciones hay valores que aumentan o disminuyen (según se quiera maximizar o minimizar) indefinidamente el valor de la función objetivo.
  - **Problema con óptimo:** Existe una solución que consigue que la función objetivo sea mínima o máxima de entre todas las soluciones.

2.2 Programación Lineal

La Programación Lineal es la Programación Matemática cuando las variables son números reales (continuos) y tanto la función objetivo como las restricciones son expresiones lineales en las variables. En este caso los modelos matemáticos se pueden representar como:

$$\begin{aligned} min\ \ & c^T x \\ s.a:\ & Ax = b \\ & x \geq\ 0 \end{aligned}$$

Figura 2.1: Modelo de programación lineal en notación matricial

Los elementos que conforman esta notación son los siguientes:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Figura 2.2: Matriz de coeficientes tecnológicos

$$b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Figura 2.3: Vector de recursos

$$c^T = (c_1, c_2, \cdots, c_n)$$

Figura 2.4: Vector de coeficientes de costes

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Figura 2.5: Vector de variables de decisión

2.3 Programación Lineal Entera

En muchas ocasiones se puede dar la necesidad de que todas o algunas de las variables tengan un carácter entero, por ejemplo, cuando se trata con seres u objetos indivisibles, como son: *las personas, los animales o las máquinas*. Por lo que la propia naturaleza del problema exige que las variables tomen valores enteros y por ende las soluciones que se buscan tengan que ser enteras.

La Programación Lineal Entera funciona de la misma manera que la Programación Lineal, pero las variables son números enteros (discretos) en lugar de números reales (continuos). La formulación de un problema de Programación Lineal Entera es la siguiente:

$$
\begin{aligned}
&min\ \ c^T x \\
&s.a:\ Ax = b \\
&\qquad x \geq 0 \wedge x_j \in Z :\ \forall_j \in J\ \subseteq\ \{1, \ldots, n\}
\end{aligned}
$$

Figura 2.6: Modelo de programación lineal entera en notación matricial

Existen dos tipos de problemas en la Programación Lineal Entera y dependiendo de si todas las variables son enteras o no, estos pueden ser los siguientes:

- **Problemas de Programación Lineal Entera Pura:** Si todas las variables del modelo son enteras.

$$
\begin{aligned}
&min\ \ c^T x \\
&s.a:\ Ax = b \\
&\qquad x \geq 0 \wedge x_j \in Z :\ \forall_j \in J\ =\ \{1, \ldots, n\}
\end{aligned}
$$

Figura 2.7: Modelo de programación lineal entera pura en notación matricial

- **Problemas de Programación Lineal Entera Mixta:** Si algunas de las variables del modelo son enteras.

$$
\begin{aligned}
&min\ \ c^T x \\
&s.a:\ Ax = b \\
&\qquad x \geq 0 \wedge x_j \in Z :\ \forall_j \in J \subset \{1, \ldots, n\}
\end{aligned}
$$

Figura 2.8: Modelo de programación lineal entera mixta en notación matricial

2.4 Técnicas de resolución en Programación Lineal Entera

Como se ha visto en el punto anterior, los problemas de Programación Lineal Entera son un caso particular de los problemas de Programación Lineal, donde todas o algunas de las variables son enteras. Cabe destacar que la búsqueda de la solución óptima en estos problemas es bastante costosa, puesto que en general, cuando se resuelve la relajación lineal no se permite redondear las componentes que sean fraccionarias [3]. A continuación, se presentan los métodos que se emplean para la resolución de problemas de Programación Lineal Entera.

2.4.1 Método de Hiperplanos de Corte

El método de *Hiperplanos de Corte* consiste en resolver problemas de Programación Lineal, de manera que se va redefiniendo la región factible del problema hasta que se encuentre la solución óptima entera. Esto supone que cada vez que no todas las componentes en la solución óptima del problema sean enteras, se tenga que agregar una nueva restricción [4]. Estas restricciones son los denominados cortes. Cabe destacar que las nuevas restricciones cumplen con la condición de integridad.

El algoritmo de *Hiperplanos de Corte* sigue la siguiente estrategia:

1. Se resuelve la relajación lineal del problema inicial. Se pueden dar dos situaciones:
   1.1. La solución que se obtiene es entera, por lo que se finaliza el proceso de búsqueda porque se encontró la solución óptima del problema.
   1.2. El nuevo problema es no factible. Se finaliza el proceso porque no existe solución.
2. ***Adición de un nuevo corte:*** Se agrega una nueva restricción que cumpla la condición de integridad, de manera que no se desechen las posibles soluciones enteras. Ir al paso 3.
3. ***Resolución del nuevo problema:*** Se resuelve el problema con la nueva restricción. Se pueden dar 3 situaciones:
   3.2. La nueva solución que se obtiene es entera, por lo que se finaliza el proceso de búsqueda porque se encontró la solución óptima del problema.
   3.3. La nueva solución que se obtiene es no es entera. Ir al paso 2.
   3.4. El nuevo problema es no factible, por lo que el problema actual también lo es. Se finaliza el proceso porque no existe solución.

2.4.2 Método de ramificación y acotación

El método de *Ramificación y Acotación* consiste en resolver una serie de problemas de Programación Lineal, hasta que se encuentre la solución óptima entera. Estos problemas, se van obteniendo al relajar las restricciones que presenta el problema de Programación Lineal Entera original. Cabe destacar que a medida que el algoritmo va trabajando, el número de restricciones que van

apareciendo va aumentando, puesto que estas restricciones permiten ir dividiendo la región factible del problema hasta que se encuentre la solución óptima entera.

El algoritmo de *Ramificación y Acotación* sigue la siguiente estrategia:

1. Se establecen el valor de la cota superior ($\tilde{z}$) a ∞.
2. ***Acotación:*** Se selecciona uno de los problemas de Programación Lineal que están en la lista de problemas a resolver ($\mathcal{L}$). Se resuelve la relajación lineal del problema $\mathcal{L}_i$ (problema seleccionado).
    2.1. $\mathcal{L}_i$ tiene solución óptima (x^*) entera:
        2.1.1. Si x^* perfecciona el mejor valor que hay hasta ahora, se actualizan el valor de la cota superior ($\tilde{z} = c^T x^*$), y el mejor valor óptimo pasa a ser ($\tilde{x} = x^*$).
        2.1.2. Si $\mathcal{L}$ está vacía, se finaliza el proceso.
        2.1.3. En otro caso: Volver al paso 2.
    2.2. En otro caso: Ir al paso 3.
3. ***Ramificación:*** Si en la solución óptima x^*, existe una componente (x_j^*) que no es entera, se añaden dos nuevos subproblemas a $\mathcal{L}$ (ramificación) donde:
    - El primer problema tendrá la restricción: $x_j \leq \lfloor x_j^* \rfloor$.
    - El segundo problema tendrá la restricción: $x_j \geq \lceil x_j^* \rceil$.

Algunos detalles a tener en cuenta cuando se emplean los métodos de *Hiperplanos de Corte* o de *Ramificación y Acotación* son los siguientes:

- Al resolver la primera relajación lineal del problema, se pueden dar las siguientes dos situaciones por las que finalizaría el algoritmo:
    - Si la solución que se obtiene es entera, esta será la solución óptima del problema.
    - Si el problema relajado es no factible, indica que el problema base también es no factible, y por lo tanto no tiene solución.
- Además, para el método de *Ramificación y Acotación*, el valor óptimo del problema base será el mínimo o máximo valor entero que se obtenga, según el problema sea de minimización o de maximización respectivamente. Y en caso de que al resolver alguno de los sucesivos problemas de Programación Lineal, el óptimo que se obtenga sea peor, se finaliza el algoritmo, puesto que no se encontrará ninguna solución óptima mejor.

Capítulo 3 **Modelo matemático**

En el primer capítulo, se presentó el problema fruto del presente trabajo, y en el capítulo anterior se dieron ciertas nociones acerca de los modelos de Programación Lineal, así como, métodos de resolución. A continuación, se presenta el modelo matemático de optimización que modeliza al problema objeto de este trabajo, indicándose: los parámetros, las variables, la función objetivo y las restricciones que tiene el propio modelo.

3.1 Parámetros

En primer lugar, los parámetros que intervienen en el modelo pueden ser diferenciados en cuatro grupos:

- Por un lado, se tienen los parámetros que introduce el/la técnico/a del ISTAC a través de la interfaz gráfica, y son con los que se puede parametrizar el modelo:
  - ***Aeropuertos canarios*** $= \{Lanzarote, Fuerteventura, \dots, La\ Palma\} \equiv$ 'Conjunto de todos los aeropuertos canarios en los que se realiza la encuesta'.[1]
  - ***Fechas seleccionadas*** $\equiv$ 'Conjunto de días seleccionados para realizar la encuesta'.
  - ***Número de entrevistadores*** $\equiv$ 'Número total de entrevistadores/as destinados a cada aeropuerto'.
  - $\boldsymbol{K} = \{\mathbf{1}, \dots, \boldsymbol{Numero\_entrevistadores}\} \equiv$ 'Conjunto de entrevistadores asociados a un aeropuerto canario'.
  - ***Jornada laboral*** $\equiv$ 'Número de horas que permanecerán los entrevistadores/as realizando encuestas'. Se expresa en horas.
  - ***Descanso*** $\equiv$ 'Número de minutos mínimos entre que se termina de encuestar un vuelo, y se empieza a entrevistar al siguiente'. Se expresa en minutos.
  - ***Velocidad*** $\equiv$ 'Tiempo que se requiere para realizar una encuesta'. Se expresa en minutos.
  - ***Ocupación*** $\equiv$ 'Estimación de la ocupación que tendrá cada vuelo'. Se expresa en tanto por ciento.

[1] Todos los parámetros de este primer grupo, pueden ser personalizados para cada uno de los aeropuertos que esté en el conjunto de aeropuertos canarios en los que se realice la encuesta.

- ***Éxito*** ≡ 'Estimación del éxito que se tendrá al realizar la encuesta a cada vuelo'. Se expresa en tanto por ciento.

- Por otro lado, se tienen los parámetros que se extraen del fichero GESLOT[2]:
  - ***Número de vuelos*** ≡ 'Número total de vuelos (tanto nacionales como internaciones) que tienen como origen un aeropuerto canario'.
  - ***Número de países*** ≡ 'Número total de países o regiones que tienen como destino un aeropuerto canario'.
  - $\boldsymbol{I} = \{\mathbf{1}, \ldots, \boldsymbol{Numero\ de\ vuelos}\}$ ≡ 'Conjunto de vuelos (tanto nacionales como internaciones) que tienen como origen un aeropuerto canario'.
  - $\boldsymbol{P} = \{\mathbf{1}, \ldots, \boldsymbol{Numero\ de\ paises}\}$ ≡ 'Conjunto de países o regiones que tienen como destino un aeropuerto canario'.
  - $\boldsymbol{I_p} \subseteq \boldsymbol{I}$ ≡ 'Subconjunto de vuelos que tienen como destino al país o región $\mathrm{p} : \mathrm{p} \in \mathrm{P}$ '.
  - $\boldsymbol{Numero\_max_p}$ ≡ 'Número máximo de pasajeros que viajan en las fechas seleccionadas al país o región $\mathrm{p} : \mathrm{p} \in \mathrm{P}$ '.
  - $\boldsymbol{dia_i}$ ≡ 'Día en que se realiza el vuelo $i : i \in I$ '.
  - $\boldsymbol{hora_i}$ ≡ 'Hora en que despega el vuelo $i : i \in I$ '.

- Además, también intervienen dos parámetros especiales:
  - $\boldsymbol{Numero\_min_p}$ ≡ 'Número mínimo de encuestas requeridas en las fechas seleccionadas al país o región $\mathrm{p} : \mathrm{p} \in \mathrm{P}$ '. Este parámetro depende de los tamaños muestrales mínimos que se muestra en la **Tabla *1.1***.
  - $\boldsymbol{INE_P}$ ≡ 'Número de encuestas realizas por el INE en las fechas seleccionadas al país o región $\mathrm{p} : \mathrm{p} \in \mathrm{P}$ '. Este parámetro es desconocido, por lo que se presupone que será nulo.

- Finalmente, se dispone de un fichero con información acerca de los distintos modelos de aviones comerciales que operan hoy en día. Para cada vuelo que esté en el fichero GESLOT, este tendrá asignado un modelo de avión, y gracias a este fichero, se conoce el número de asientos que dispone el avión. Esto conlleva a que se tengan los siguientes parámetros calculados:
  - $\boldsymbol{Numero\ de\ asientos_i}$ ≡ 'Número de asientos que dispone el avión que realiza el vuelo $i : i \in I$ '.
  - $\boldsymbol{encuestas_i} = num\_asientos_i \cdot ocupacion \cdot exito$ ≡ 'Número de encuestas que se estiman para el vuelo $i : i \in I$ '.

[2] Para cada aeropuerto canario donde se realice la encuesta, los parámetros de este grupo serán distintos, puesto que no todos los vuelos, destinos, horarios, etc., son iguales.

- $\boldsymbol{consumo_i} = \text{encuestas}_i \cdot velocidad \equiv$ 'Tiempo necesario para realizar las encuestas que se estiman para el vuelo $i : i \in I$ '.

Algunos detalles a tener en cuenta son los siguientes:

- El/la técnico/a del ISTAC decidirá qué aeropuertos del conjunto de aeropuerto canarios van a ser introducidos en el modelo, y para cada uno de ellos, todos los parámetros que estén bajo su elección los podrá ajustar según considere oportuno. Esto es así, para dotar a la herramienta de una mayor versatilidad, de manera que sirva como banco de pruebas para poder tomar decisiones más rápidas para cada aeropuerto.

3.2 Variables

Las variables que va a tener el problema serán variables de decisión (variables binarias), y son las siguientes:

$$x_{ik} = \begin{cases} 1 & Si\ el\ vuelo\ i\ es\ encuestado\ por\ el\ encuestador\ k \\ 0 & en\ otro\ caso \end{cases} \quad \forall\, i \in I \ \wedge\ \forall\, k \in K \quad (3.1)$$

$$y_i^1 = \begin{cases} 1 & Si\ el\ vuelo\ i\ es\ encuestado\ por\ un\ solo\ encuestador \\ 0 & en\ otro\ caso \end{cases} \quad \forall\, i \in I \quad (3.2)$$

$$y_i^2 = \begin{cases} 1 & Si\ el\ vuelo\ i\ es\ encuestado\ por\ mas\ de\ un\ encuestador \\ 0 & en\ otro\ caso \end{cases} \quad \forall\, i \in I \quad (3.3)$$

$$z_p = \begin{cases} 1 & Si\ no\ se\ cumple\ con\ el\ mínimo\ de\ encuestas\ requeridas\ para\ la\ región\ p \\ 0 & en\ otro\ caso \end{cases} \quad \forall\, p \in P \quad (3.4)$$

Con la variable (3.1) se pretende seleccionar el encuestador o encuestadores que se encargan de realizar la encuesta al vuelo i.

Con las variables (3.2 y 3.3) se pretende indicar si el vuelo i será encuestado por un solo encuestador (variable y_i^1 a 1) o por varios (variable y_i^2 a 1).

Con la variable (3.4) se pretende indicar aquellas regiones que son penalizadas por no cumplir con el número mínimo de encuestas requeridas de acuerdo con la **Tabla 1.1**.

3.3 Función objetivo

Con respecto a la función objetivo, cabe destacar que al tratarse de un modelo de optimización se requiere maximizar o minimizar algún recurso. En este caso, el recurso que se requiere maximizar es el número de encuestas que se realicen. Se presentan tres funciones objetivo distintas, que aparentemente son iguales, a excepción del coeficiente que acompaña a las variables y_i^1e y_i^2.

Claramente, cada una de las funciones objetivo puede ser diferenciada en dos partes:

- En la primera parte, se tiene un sumatorio que es el encargado de para cada vuelo i, sumar todas las encuestas que se realicen, tanto vuelos que sean encuestados por 1 solo encuestador/a como para los que sean encuestados por varios/as.
- En la segunda parte, se impone una penalización mediante un coeficiente negativo que puede ser modificado, pero a efectos prácticos se ha querido abstraer al técnico/a del ISTAC de esta decisión. Con dicha penalización, el efecto que se consigue es que con el empleo de la variable z_p, no se produzcan problemas no factibles, de manera que siempre va a existir una solución. Sin embargo, esto conlleva a que se obtenga un peor valor óptimo debido a que para el país o región p no se logran el mínimo número de encuestas requeridas.

A continuación, se presentan las tres posibles funciones objetivo que se ponen a disposición del técnico/a del ISTAC:

- Esta primera función objetivo presenta una neutralidad en cuanto a la importancia de emplear uno o varios/as encuestadores para encuestar a cada vuelo i. A partir de esta función objetivo se construyen las siguientes, sufriendo algunos cambios que harán que el modelo se comporte de forma diferente.

$$max \sum_{i \in I} encuestas_i \cdot (y_i^1 + y_i^2) - 100 \cdot \sum_{p \in P} Num\_max_p \cdot z_p \quad (3.5)$$

- Con esta segunda función objetivo, se presenta un interés en cuanto a la importancia de emplear varios/as encuestadores frente a uno/a para encuestar a cada vuelo i. De manera que se impone un factor multiplicativo que dota de mayor significación a la variable y_i^2.

$$max \sum_{i \in I} encuestas_i \cdot (y_i^1 + 2 \cdot y_i^2) - 100 \cdot \sum_{p \in P} Num\_max_p \cdot z_p \quad (3.6)$$

- Por último, con esta función objetivo también se presenta un interés en cuanto a la importancia de emplear varios/as encuestadores frente a uno/a. Sin embargo, en este caso se emplea un factor multiplicativo negativo a la variable y_i^1, dotando de mayor significación a la variable y_i^2.

$$max \sum_{i \in I} encuestas_i \cdot (-y_i^1 + y_i^2) - 100 \cdot \sum_{p \in P} Num\_max_p \cdot z_p \quad (3.7)$$

Más adelante, en el **Capítulo 5** se mostrará los efectos tanto en tiempo como en resultados obtenidos, que supone emplear cada una de las funciones objetivo.

3.4 Restricciones

Por último, se presenta el conjunto de restricciones que deben satisfacer las variables:

- Esta primera restricción se encarga de indicar que todo vuelo deberá ser encuestado por 1, varios o ningún encuestador, dado que solo puede ser seleccionada una de las variables de decisión $y_i^j : j \in \{1,2\}$.

$$y_i^1 + y_i^2 \leq 1 \quad \forall\, i \in I \quad (3.8)$$

- Esta segunda restricción se emplea para garantizar que el número de encuestadores asignados para cada vuelo i se corresponde con lo establecido, es decir, en el caso de que la variable y_i^1 tome el valor 1, el número de encuestadores asignados al vuelo i será 1, mientras que, si la variable y_i^2 toma el valor 1, todos los encuestadores disponibles para ese aeropuerto serán asignados al vuelo i.

$$\sum_{k \in K} x_{ik} = y_i^1 + |K| \cdot y_i^2 \quad \forall\, i \in I \quad (3.9)$$

- Esta tercera restricción es la más importante del modelo, puesto que es la que activa o no la variable z_p, en función de si se cumple con el tamaño mínimo muestral. Para ello, se selecciona el subconjunto de vuelos que tengan destino la región p, y de este subconjunto, se suman el total de encuestas potenciales que se pueden obtener empleando tanto un solo encuestador como más de uno, para todos los vuelos que tengan como destino esa región p:

$$\sum_{i \in I_p} encuestas_i \cdot (y_i^1 + y_i^2) \geq (\text{Numero\_min}_\text{p} - INE\_p) \cdot (1 - z_p) \;\forall\, p \in P \quad (3.10)$$

- A continuación, se tiene un conjunto de restricciones de acuerdo con las siguientes condiciones: $\forall\, k \in K \;\wedge\; \forall\, i,j \in I : \{i \neq j \;\wedge\; dia_i = dia_j \;\wedge\; hora_i \leq hora_j\}$:

  - Si $hora_i > hora_j - \frac{consumo_j}{Numero\_entrevistadores} - descanso$:

$$x_{ik} + x_{jk} \leq 1 \quad (3.11)$$

  - Si $hora_i - \frac{consumo_i}{Numero\_entrevistadores} + jornada < hora_j$:

$$x_{ik} + x_{jk} \leq 1 \quad (3.12)$$

  - Si $hora_i > hora_j - consumo_j - descanso$:

$$x_{ik} + x_{jk} \leq 1 + \sum_{l \in K \setminus \{k\}} x_{jl} \quad (3.13)$$

  - Si $hora_i - consumo_i + jornada < hora\_j$:

$$x_{ik} + x_{jk} \leq 1 + \sum_{l \in K \setminus \{k\}} x_{il} \quad (3.14)$$

- Por último, se tiene que imponer la restricción de integridad. Esto quiere decir que todas las variables del modelo son variables enteras y que tomarán valores de 0 o 1:

$$x_{ik},\, y_i^1,\, y_i^2,\, z_p \in \{0,1\} \quad \forall\, i \in I, \forall\, k \in K, \forall\, p \in P \quad (3.15)$$

Capítulo 4 Implementación de la aplicación

En este capítulo, se detalla la aplicación que se ha desarrollado para ayudar en la toma de decisiones sobre qué vuelos encuestar. Se detallan las tecnologías y herramientas empleadas, la arquitectura que se ha seguido, la implementación del modelo matemático, etc.

4.1 Software, herramientas y librerías utilizadas

La mayoría de las herramientas utilizas para llevar a cabo el proyecto son de software libre, y aquellas que no, disponen de licencias para estudiantes o versiones gratuitas. En el apéndice **9.3 Herramientas utilizadas para el proyecto** se detallan todas las herramientas que se emplearon para el desarrollo del proyecto.

4.2 Arquitectura de la aplicación

La arquitectura que se ha empleado para la aplicación es el patrón Modelo Vista Controlador (MVC) [5]. Este patrón separa el código de la aplicación en las siguientes tres secciones:

- **Modelo:** Contiene los datos de la aplicación, así como las operaciones para modificar dichos datos. Esto es conocido como la lógica de negocio de la aplicación.
- **Vista:** Se trata de la parte de la aplicación que ve el usuario y con la que interactúa directamente. Se emplea para mostrar los datos del modelo y para facilitar mecanismos con los que el usuario invoque a los eventos que realiza el controlador.
- **Controlador:** Se emplea para llevar a cabo los eventos que el usuario solicita a través de la vista. Además, se encarga de realizar las actualizaciones de la vista y de indicar los cambios a los datos del modelo.

Se ha decidido seguir el patrón MVC dado que con la separación de componentes se encapsulan las distintas actividades que realiza la aplicación, permitiendo un mejor mantenimiento, la reutilización de componentes y la facilitación de la escalabilidad de la aplicación.

En la implementación que se ha realizado, el modelo está compuesto por tres objetos que encapsulan los datos de la aplicación, y son: uno para los parámetros

introducidos por el usuario, otro para los ficheros y otro para almacenar la solución que se obtiene del modelo matemático. Además, estos contiene las herramientas necesarias para acceder y manipular los datos.

Para la vista se han empleado un total de cuatro vistas principales y una auxiliar, que se corresponde con una vista que contiene una barra de progreso para mostrar el estado de la acción que se está realizando. Las vistas de las que se compone la aplicación son las siguientes:

- ***Vista general (v_general.py):*** Se corresponde con la vista principal de la aplicación, desde donde se acceden a las vistas secundarias que dan acceso a los menús que permiten la personalización y configuración de los parámetros de entrada del modelo.

- ***Vista menú de parámetros (v_parametros.py):*** Contiene los componentes necesarios para poder introducir los parámetros que pueden personalizar el usuario.

- ***Vista menú ETL (v_etl.py):*** Contiene los componentes necesarios para poder introducir los ficheros de GESLOT y de los aviones.

- ***Vista menú para planificar (v_planificar.py):*** Contiene los componentes necesarios para poder introducir las configuraciones del modelo, como son la función objetivo y el solucionador a emplear.

- ***Vista barra de progreso (v_barra_progreso.py):*** Se trata de una vista auxiliar que contiene los componentes necesarios que informan del estado de los procesos que requieren un cierto tiempo para llevarse a cabo, como son: el ELT y la resolución del problema.

Para cada vista se implementó un controlador específico, encargado de:

- Gestionar el contenido de cada vista.
- Comunicarse con el modelo para realizar las operaciones de acceso y manipulación de los datos.

Además, cabe mencionar que debido a que la mayoría de los componentes que conforman las vistas (v.gr botones, etiquetas, checklist, etc.) son repetidos, se confeccionaron objetos para cada uno de ellos. De esta manera, se permite aún más la reutilización de los componentes, logrando así:

- Personalizar el contenido que tiene el propio componente.
- Indicar la ubicación que tendrá en la vista.
- Emplear métodos específicos para poder interactuar con ellos.

Por otro lado, cabe mencionar que se ha empleado un hilo diferente al que utiliza la aplicación para que se encargue de controlar la vista de la barra de progreso. Esto es así debido a que los procesos de ETL y de resolución del problema requieren un cierto tiempo y la interfaz gráfica se queda en espera realizando dichos procesos. Es por ello por lo que se ha decidido utilizar otro hilo que controle la ventana auxiliar y que en ella se muestre el estado del proceso que se está llevando a cabo, de manera que el usuario sepa en todo momento en que punto se encuentra dicho proceso.

Por último, en el apéndice **9.2 Flujograma de la aplicación** se incluye un flujograma donde se describen la secuencia de pasos a seguir para utilizar la aplicación.

4.3 Proceso de extracción, transformación y carga (ETL)

Un proceso ETL [6] consiste en:

- ***Extract:*** Extraer datos de uno o varios orígenes con formatos iguales o distintos.
- ***Transform:*** Transformar los datos mediante la aplicación de acciones como combinación, limpieza, validación, ordenación, etc.
- ***Load:*** Cargar los datos transformados en una nueva fuente de datos.

Para el presente trabajo, se precisó realizar el proceso ETL que describe la **Figura 4.1**, a fin de preparar los parámetros de entrada necesarios al modelo. En líneas generales, se trabaja sobre el fichero GESLOT (que es el que más información contiene), de manera que se busca obtener un dataframe con la información que requiere el modelo matemático en función de los parámetros y fichero de aviones que haya introducido el usuario. Para realizar este proceso, así como muchas operaciones internas de la aplicación, se emplearon las librerías de Pandas [7] y Numpy [8], que permiten realizar un buen tratamiento sobre los datos y contienen numerosas funciones optimizadas.

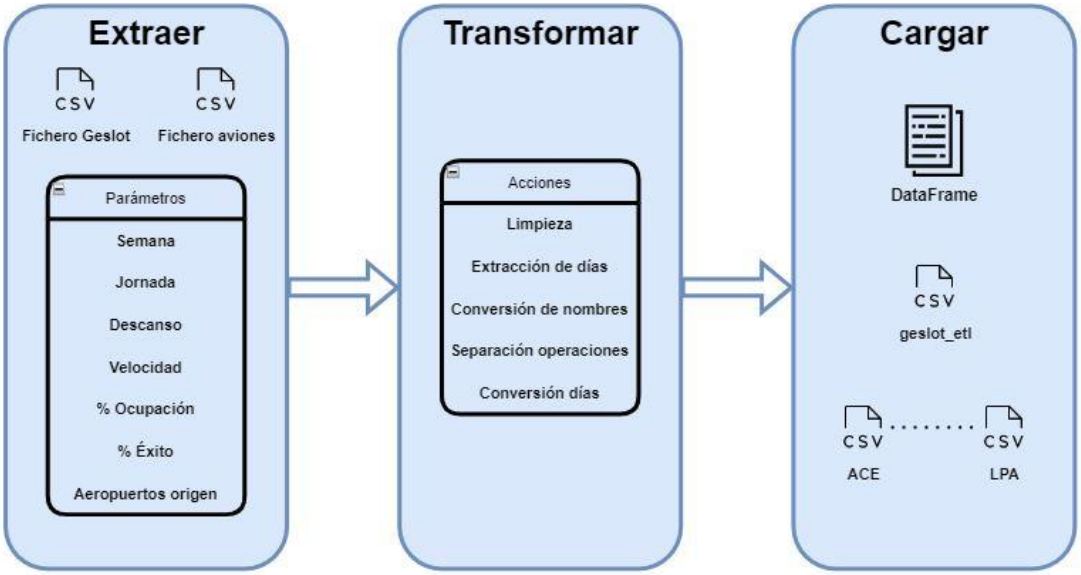


Figura 4.1: Diagrama del ETL implementado

Las acciones de las que está compuesta la fase de transformación del ETL que se implementó son las siguientes:

- ***Limpieza:*** Se eliminan las columnas innecesarias y las filas que tienen como aeropuerto de origen un aeropuerto que no forma parte de los seleccionados en los parámetros.
- ***Extracción de días:*** En primer lugar, se realiza una transformación (al tipo de dato *date*) tanto de los días seleccionados en los parámetros, como en los datos de las columnas que contienen las fechas con el inicio (opera desde) y el fin (opera hasta) de las operaciones para cada vuelo. A continuación, se obtiene el primer y último día introducido, que serán el mínimo y máximo valor de la semana (o días seleccionados) respectivamente. Por último, para las fechas que recogen las columnas del inicio y fin de operaciones, se eliminan las filas que queden fuera del intervalo [primer día, último día].
- ***Conversión de nombres:*** Para evitar problemas en los formatos de codificación, se cambian algunos nombres de regiones de destino (v. gr España, Federación Rusa, República Checa, etc.). Cabe mencionar que este proceso es revertido en una de las operaciones de transformación que se realiza en el Informe de Power BI (que se verá en el punto **5.1** de la presente memoria), de manera que se muestre el nombre del destino o región tal como es.
- ***Separación de operaciones:*** Se codifica en formato numérico los días en que se realizan las operaciones (1 es lunes y 7 es domingo), dándose casos en que por ejemplo si el vuelo realiza operaciones los martes, miércoles y domingos, se correspondería al valor *237*. De esta manera, se va a dividir el dataframe en dos, por un lado, se van a almacenar las filas que contengan vuelos que realicen operaciones una vez a la semana (se comprueba si el día de la semana es <= 7), y otro dataframe contendrá las filas que contengan vuelos que realicen operaciones varias veces a la semana (se comprueba si el día de la semana es > 7). Para este último caso, se crearán tantas nuevas filas como dígitos tenga la columna *día_sem*. Por último, se juntarán ambos dataframes.

- ***Conversión de días:*** Como para cada fila ya se tiene un único día de la semana, con este paso se cambiará el día que opera por la fecha que corresponda, es decir, si un vuelo opera un lunes (*día_sem = 1*), dicho lunes se codificará con el formato en fecha del día que corresponda para la semana seleccionada por el usuario.

4.4 Implementación del modelo matemático

El modelo matemático que modeliza el problema fruto del presente trabajo puede ser implementado en cualquier lenguaje, librería o herramienta de optimización.

En primer lugar, la idea que se tuvo inicialmente fue implementar el modelo con la herramienta OR-TOOLS [9], que es la suite de optimización de Google. Esto fue así debido a que las garantías y calidad que ofrece una librería que recibe soporte de una compañía como Google son mayores que cualquier otras. Además, uno de los objetivos que busca este trabajo, es poder dar la posibilidad de invocar a distintos solucionadores (gratuitos y con licencia), a modo de dotar a la aplicación de una mayor versatilidad. OR-TOOLS ofrece la posibilidad de poder llevar esto a cabo, sin embargo, no es posible realizarlo de manera sencilla, siendo necesario llevar a cabo un proceso bastante laborioso y sin éxito asegurado.

El modelo se implementó utilizando la librería de OR-TOOLS, sin embargo, solo se permitía invocar al solucionador de COIN-OR [10] de manera predeterminada, no siendo posible invocar a otro solucionador de la misma manera.

Debido a los problemas que esto acarreó, se decidió implementar el modelo con PuLP [11], una librería de software libre para Python que tiene el soporte de la comunidad de COIN-OR, y que permite modelar problemas de Programación Lineal. Además de poder codificar modelos también funciona como una API y ofrece la posibilidad de conectarse a multitud de solucionadores, entre ellos: GUROBI, GLPK, COIN-OR, CPLEX, etc.

La implementación del modelo con PuLP es muy natural, como se muestra en la **Figura 4.2**, de manera que viendo la codificación se puede interpretar fácilmente qué es lo que se lleva a cabo. Además, ofrece dos modos de poder conectarse a los distintos solucionadores, y son a través de:

- La interfaz de línea de comandos del solucionador.
- Una librería en Python del propio solucionador.

Se dan estas dos posibilidades debido a que la mayoría de solucionadores disponen de una interfaz por línea de comandos, en cambio, no todos disponen de una librería propia en Python.

```
self.__y1 = {}
for i in self.__I:
    self.__y1[i] = LpVariable("y1(" + str(i) + ")", 0, 1, LpBinary)

self.__y2 = {}
for i in self.__I:
    self.__y2[i] = LpVariable("y2(" + str(i) + ")", 0, 1, LpBinary)
```

Figura 4.2: Definición de algunas de las variables de decisión del modelo

4.5 Test y cobertura

A la mayoría de las funciones y métodos que contiene la aplicación, se le realizaron sus correspondientes pruebas para comprobar el correcto funcionamiento de estas. Dichas pruebas se realizaron mediante la librería de Pytest [12], uno de los módulos que permiten la elaboración de testeos o pruebas a programas codificados en Python.

Un ejemplo de cómo se han realizado las pruebas a las funciones, se muestra en la **Figura 4.3**. Como se verá en la sección **4.7.1 Workflows**, se ha llevado a cabo una integración continúa empleando GitHub Actions, de manera que las pruebas que se realizaron con Pytest se ejecutan en un workflow en GitHub.

```
def test_get_dias():
    assert controlador_parametros.get_modelo_parametros().get_dias() == [
        "01/03/2022",
        "02/03/2022",
        "03/03/2022",
    ]


def test_get_jornada():
    assert controlador_parametros.get_modelo_parametros().get_jornada() == 8


def test_get_descanso():
    assert controlador_parametros.get_modelo_parametros().get_descanso() == 10


def test_get_velocidad():
    assert (
        controlador_parametros.get_modelo_parametros().get_velocidad() == 0.5
    )


def test_get_ocupacion():
    assert (
        controlador_parametros.get_modelo_parametros().get_ocupacion() == 0.80
    )


def test_get_exito():
    assert controlador_parametros.get_modelo_parametros().get_exito() == 0.60
```

Figura 4.3: Ejemplo de testeo de los métodos del controlador de parámetros

Para ejecutar las pruebas y ver la cobertura de código, se desarrolló un script que automatizaba las pruebas, el cual se encuentra disponible en el siguiente enlace. Como resultado final, se obtuvo un cubrimiento de código del 86%.

4.6 Documentación

La mayoría de las funciones y clases que contiene la aplicación fueron documentadas mediante la librería de Pydoc [13], que es el módulo estándar de documentación de Python. Se pueden emplear varios estilos o formatos para elaborar la documentación, pero se decidió emplear estilo que utiliza en SciPy [14] dado que es muy visual, claro y fácil de entender. Dicho estilo sigue la siguiente estructura:

- Breve descripción
- Parámetros:
  - Se indican los parámetros.
  - Se realiza una breve descripción de los parámetros.
- Valores de retorno:
  - Se indican los valores de retorno.
  - Se realiza una breve descripción de los valores de retorno.

Un ejemplo de cómo ha quedado comentado el código se muestra en la **Figura 4.4**. Al realizar la documentación de un proyecto en Python y emplear la librería Pydoc, esta puede ser mostrada por la consola o guardada en archivos HTML generados automáticamente.

```
def conversor_aeropueros(lista_original):
    """
    Se realiza la conversión de los nombres de los aeropuertos en sus
    correspondientes códigos de la IATA.

    Parameters
    ----------
    lista_original : list
        Lista que contiene nombres de aeropuertos

    Returns
    -------
    list
        Lista que contiene los aeropuertos convertidos a código IATA.
    """
```

Figura 4.4: Ejemplo de función comentada

4.7 GitHub

Como sistema de control de versiones se ha empleado Git [15], y como plataforma para alojar el código GitHub [16] debido a las ventajas que posee, como son: es gratuito, permite compartir y tener en cualquier lugar el código desarrollado o las GitHub Actions para llevar a cabo automatizaciones.

Al código desarrollado se puede acceder a través del siguiente enlace: Código fuente

4.7.1 Workflows

Para llevar a cabo un proceso de integración continua, se han empleado las GitHub Actions creando un flujo de trabajo (workflow). El workflow que se empleó fue para ejecutar de manera automática las pruebas realizadas con Pytest. Esto se realizó mediante un fichero YAML, en el que se especificó la configuración workflow en cuestión, de manera que cada vez que se realiza una actualización al repositorio se activa el evento que pone en funcionamiento la automatización de las pruebas.

4.8 Distribución de la aplicación

La aplicación desarrollada cuenta con dos versiones diferentes, lo que permite que sea funcional en los sistemas operativos de Windows y GNU/Linux.

Para poder obtener el ejecutable de la aplicación se empleó la librería de PyInstaller [17], que se encarga de empaquetar la aplicación junto a todas las dependencias en un ejecutable. Esto permite que en el ordenador donde se quiera usar la aplicación, no sea necesario disponer del intérprete de Python ni ninguna dependencia que requiera la aplicación.

Cabe destacar que Pyinstaller no es un compilador cruzado [18], por lo que no es capaz de crear un ejecutable de la aplicación para Windows desde un sistema GNU/Linux, por lo que para obtener cada una de las versiones, se debió tener el proyecto en ambos sistemas operativos y así poder invocar al módulo de Pyinstaller en cada uno de ellos, obteniendo así el ejecutable con toda la lógica que requiere para cada uno de los sistemas operativos.

A continuación, se ponen a disposición del lector los enlaces para obtener los ejecutables:

- Versión para Windows (testeada en Windows 10): Versión Windows
- Versión para GNU/Linux (testeada en Ubuntu 18.04): Versión GNU/LINUX

Capítulo 5 **Experimentación**

En este capítulo se exponen las herramientas empleadas para la fase de experimentación de la aplicación, así como una serie de experiencias probando la aplicación y conclusiones sobre estas.

5.1 Introducción

Para llevar a cabo la fase de experimentación se tomaron como ejemplos una serie de parámetros de entrada tal y como lo haría un/a técnico/a del ISTAC de cara a realizar la planificación de las encuestas. Dichos parámetros de entrada fueron los siguientes:

- **Aeropuertos de origen:** La Palma, Tenerife Sur, Tenerife Norte, Gran Canaria, Fuerteventura y Lanzarote.
- **Semana:** Del 07/03/2022 al 13/03/2022
- **Jornada laboral:** 8 horas
- **Descanso:** 10 minutos
- **Velocidad:** 0.5 minutos
- **Ocupación:** 80%
- **Éxito:** 60%
- **Fichero Geslot:** Fichero Geslot de marzo de 2022
- **Fichero aviones: Ver 9.1 Tabla modelos de aviones**

Para llevar a cabo el proceso de planificación, el flujo de trabajo que se sigue se muestra en la **Figura 5.1**, que como se puede ver queda diferenciado en dos partes:

- Trabajo con la aplicación:
  - Recoger los parámetros de entrada.
  - Realizar el proceso ETL sobre los parámetros.
  - Cargar los parámetros en el modelo.
  - Invocar al solucionador que se quiera emplear (de entre los dos disponibles).
  - Obtención de los ficheros de salida formateados.
- Visualizar las soluciones obtenidas.

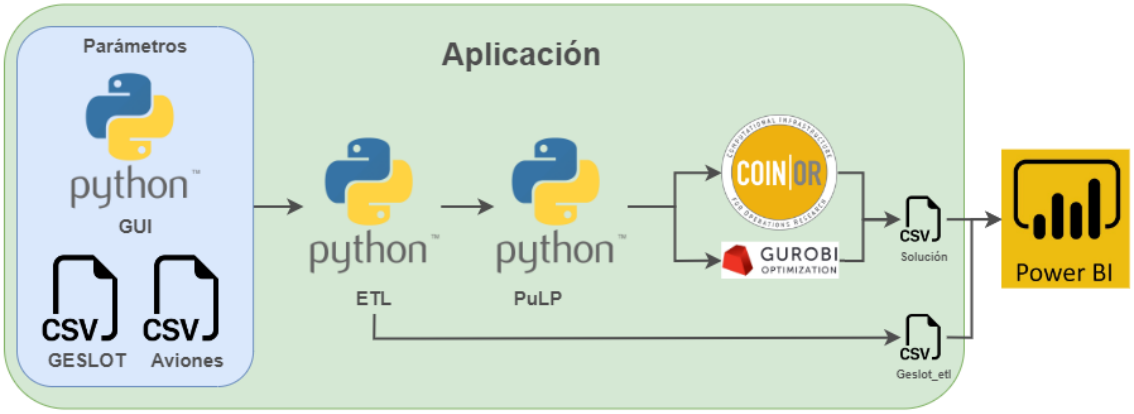


Figura 5.1: Esquema funcionamiento

Por otro lado, con el objetivo de visualizar los resultados y facilitar la toma de decisiones se ha elaborado un informe con Power BI [19], de manera que funcione como una plantilla de visualización. En la **Figura 5.2** se muestran los orígenes de datos con los que se nutre el informe, los cueles contienen los siguientes datos:

- **Geslot ETL (geslot_etl.csv):** Fichero que contiene los datos cargados tras realizar el proceso de ETL.
- **Solución modelo (solucion.csv):** Fichero que contiene los resultados obtenidos por el modelo matemático. Se emplean columnas calculadas para obtener porcentajes y para obtener qué rango de encuestas.
- **Tiempos de cómputo (tiempos_computo.csv):** Fichero que contiene los tiempos de cómputo que se han necesitado para la resolución de los problemas.
- **Aeropuertos Canarias:** Tabla elaborada desde el propio Power Bi con la que se relacionan las otras dos fuentes de datos.

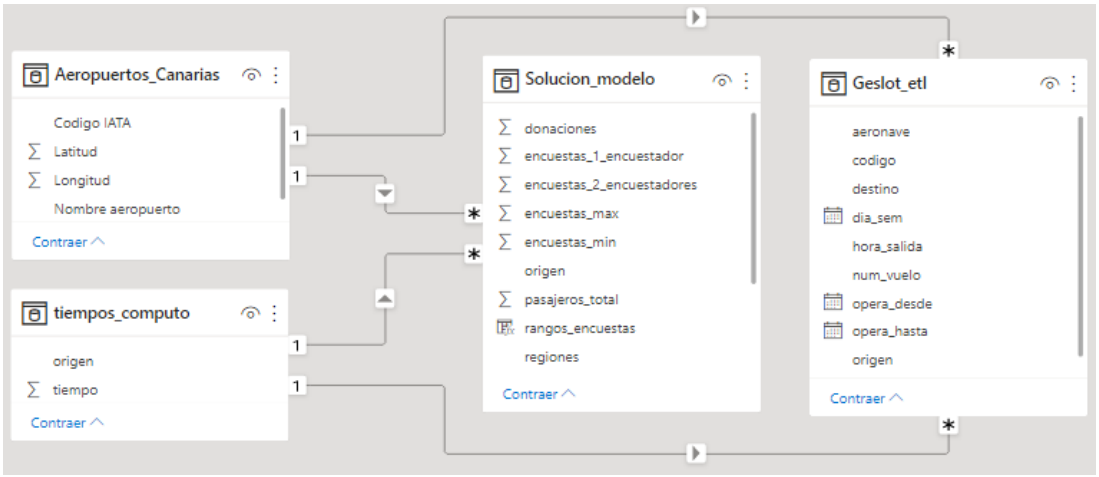


Figura 5.2: Relaciones entre las fuentes de datos

El informe de Power BI está compuesto por las siguientes páginas:

- **Encuestas por región:** Se trata de la página inicial, y contiene un resumen de las encuestas que se deben realizar. Se puede filtrar por regiones de destinos, de manera que los datos quedan sesgados en función de dichos filtros.
- **Vuelos a encuestar por región:** Sigue el mismo esquema que la página anterior, pero contiene un resumen de los vuelos a los que se deben realizar las encuestas. De igual manera, se puede filtrar por regiones de destinos.
- **Aeropuerto:** Dispone de un filtro mediante el cual se selecciona el aeropuerto de origen que se quiera visualizar. Contiene un resumen general tanto de vuelos como de encuestas
    - **Aeropuerto – Encuestas:** En función del aeropuerto que se ha seleccionado en la página *Aeropuerto*, se realizan una serie de visualizaciones para profundizar sobre las encuestas a realizar.
    - **Aeropuerto – Vuelos:** En función del aeropuerto que se ha seleccionado en la página *Aeropuerto*, se realizan una serie de visualizaciones para profundizar sobre los vuelos a encuestar.

El informe de Power BI se emplea tras haber trabajado sobre la aplicación y haber obtenido los resultados. Gracias a la posibilidad que ofrece Power BI de actualizar los datos, se pueden cargar de manera automática los ficheros que contiene la solución, de manera que los datos que se visualicen en el informe queden actualizados. Esto se consigue depositando los ficheros en la misma ruta que se indicó en el informe, desde donde este extrae los datos y los carga.

5.2 Solucionador de Gurobi

Gurobi [20] es un solucionador de optimización matemática de pago capaz de resolver problemas de Programación Lineal (en todas sus versiones). Dispone de licencias de pago y licencias académicas, las cuales se han empleado para poder resolver los problemas.

Para poner a prueba la aplicación empleando el solucionador de Gurobi[3], se han realizado dos experimentos (Gurobi I y Gurobi II) en los que se han utilizado los mismos parámetros de entrada, y que se corresponden a los indicados en la sección **5.1 Introducción**. Lo que diferencia un experimento del otro es la función objetivo empleada, sin embargo, se puede afirmar que el objetivo que persiguen es el mismo, dado que se busca que en las soluciones que se obtengan se priorice emplear dos encuestadores. Por un lado, en Gurobi I se ha empleado la función objetivo que priorizando emplear dos encuestadores y, por otro, en Gurobi II, se ha utilizado la función objetivo que penaliza utilizar un encuestador.

A continuación, se presentan los resultados obtenidos en los experimentos con Gurobi:

[3] Para poder invocar al solucionador de Gurobi, se debe disponer de una licencia para dicho software instalada y verificada en el ordenador donde se quiera utilizar.

- Experimento Gurobi I:

Función objetivo: *Priorizando emplear 2 encuestadores*				
Aeropuerto:	Tiempo de cómputo:	Total encuestas 1 encuestador:	Total encuestas 2 encuestadores:	Donaciones:
ACE	0,7086 seg	267	7218	32
FUE	0,3284 seg	0	7285	0
LPA	57,004 seg	1473	6390	0
TFN	0,4924 seg	0	5776	10
TFS	3,7440 seg	0	8270	0
SPC	0,0519 seg	0	3633	0
TOTAL:	64,910 seg	1740	38572	42

Tabla 5.1: Resultados experimento Gurobi I

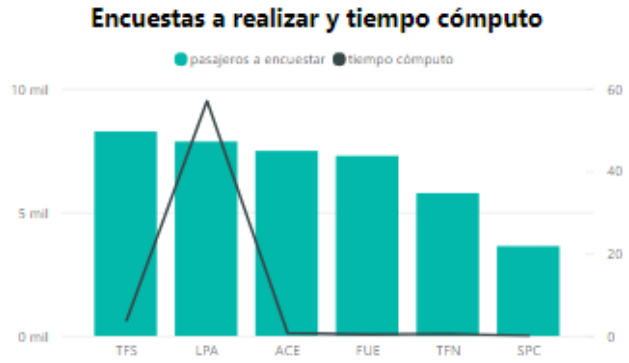


Figura 5.3: Gráfica encuestas a realizar y tiempos de cómputo Gurobi I

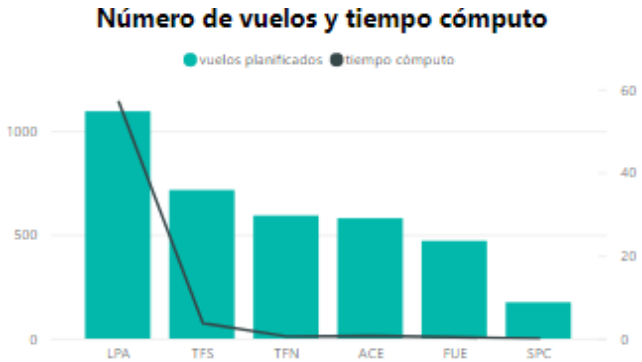


Figura 5.4: Gráfica vuelos planificados y tiempos de cómputo Gurobi I

- Experimento Gurobi II:

Función objetivo: *Penalizando emplear 1 encuestador*				
Aeropuerto:	Tiempo de cómputo:	Total encuestas 1 encuestador:	Total encuestas 2 encuestadores:	Donaciones:
ACE	0,5637 seg	140	7218	32
FUE	0,3545 seg	0	7285	0
LPA	8,3373 seg	463	6278	0
TFN	0,4668 seg	0	5776	10
TFS	1,8583 seg	0	8270	0
SPC	0,0488 seg	0	3633	0
TOTAL:	11,629 seg	603	38460	42

Tabla 5.2: Resultados experimento Gurobi II

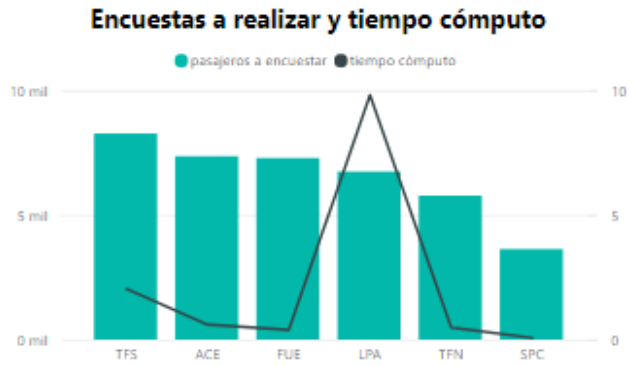


Figura 5.5: Gráfica encuestas a realizar y tiempos de cómputo Gurobi II

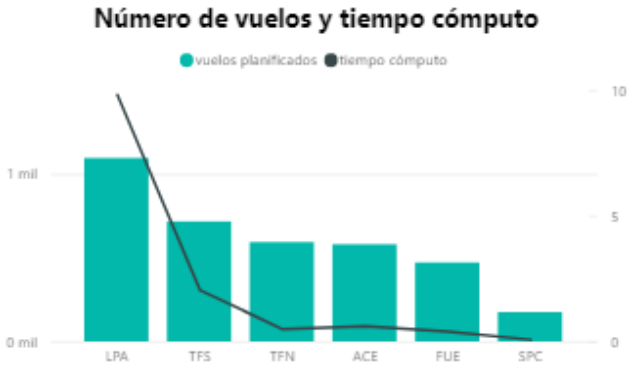


Figura 5.6: Gráfica vuelos planificados y tiempos de cómputo Gurobi II

Como se puede observar en la **Tabla 5.1** y en la **Tabla 5.2**, para ambos experimentos se han encontrado las soluciones óptimas en tiempos razonables, siendo el mayor de ellos el caso en que se prioriza emplear dos encuestadores para el aeropuerto de Gran Canaria, con 57,004 segundos.

Un dato a destacar es que para ambos casos el aeropuerto de Tenerife Sur es donde se obtienen el mayor número de encuestas a realizar (8270 encuestas), sin embargo, como se muestra en la **Figura 5.4** y la **Figura 5.6** Tenerife Sur no es el aeropuerto con más vuelos planificados, ni el aeropuerto con más volumen de pasajeros y, por ende, no es donde a priori hay un mayor número de encuestas a realizar. Es el aeropuerto de Gran Canaria al que se le atribuyen todas estas características, y que pese a tener un mayor número de vuelos y de pasajeros, se obtienen diferencias con el aeropuerto Tenerife Sur de 407 encuestas y 1521 encuestas para los experimentos de Gurobi I y Gurobi II respectivamente.

Adicionalmente, se han agregado las figuras **Figura 5.3** y **Figura 5.5**, donde se muestran el total de encuestas a realizar frente al tiempo de cómputo empleado. De acuerdo con lo comentado en el punto anterior y como se puede observar en dichas figuras, en el experimento Gurobi II para los aeropuertos de Lanzarote, Fuerteventura y el propio Tenerife Sur, se obtienen más encuestas a realizar que el aeropuerto de Gran Canarias. Esto puede venir motivado a que los slots de tiempo que hay en estos aeropuertos están mejor planificados que los del aeropuerto de Gran Canaria, ya sea porque en el aeropuerto de Gran Canaria están muy separados o muy juntos, provocando que con dos encuestadores no se consiga poder planificar la realización de un mayor número de encuestas.

Por otro lado, como se puede ver en la **Tabla 5.1** y en la **Tabla 5.2**, las donaciones que se requieren son las mismas. Además, este número de donaciones o registro donantes no es muy elevado, y representa el 0,1041% y el 0,1075% del total de encuestas a realizar para Gurobi I y Gurobi II respectivamente.

Por último, partiendo de la premisa que el objetivo principal es maximizar el número de encuestas a realizar, y que se cumpla los tamaños mínimos muestrales, el experimento con el que se consiguieron más encuestas fue el Gurobi I, con una diferencia de 112 encuestas con Gurobi II.

5.3 Solucionador de COIN–OR

COIN–OR son las siglas de *Computational Infraestructure For Operations Research*, una fundación sin ánimo de lucro que se encarga de administrar el proyecto con su mismo nombre, y que ofrece el solucionador de programación lineal entera COIN-OR Branch and Cut (CBC).

Al igual que con Gurobi, para poner a prueba la aplicación empleando el solucionador de CBC, también se han realizado dos experimentos (CBC I y CBC II) en los que se han utilizado los mismos parámetros de entrada que en el caso de Gurobi. Además, respecto a las funciones objetivo, se emplearon de la misma manera que con Gurobi.

A continuación, se presentan los resultados obtenidos en los experimentos con CBC:

- CBC 1:

Función objetivo: *Priorizando emplear 2 encuestadores*				
Aeropuerto:	Tiempo de cómputo:	Total encuestas 1 encuestador:	Total encuestas 2 encuestadores:	Donaciones:
ACE	33,699 seg	267	7218	32
FUE	11,571 seg	0	7285	0
LPA	-	-	-	-
TFN	19,334 seg	0	5776	10
TFS	-	-	-	-
SPC	0,5338 seg	0	3633	0
TOTAL:	65,137 seg	267	23912	42

Tabla 5.3: Resultados experimento CBC I

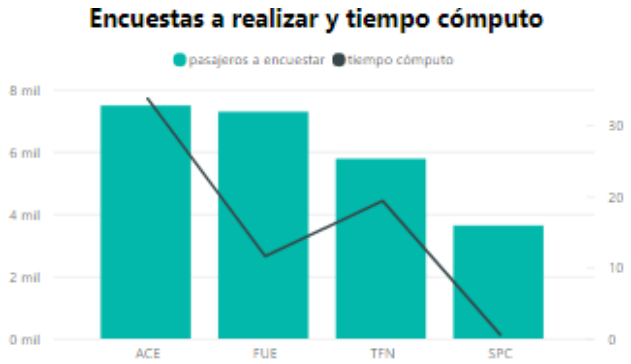


Figura 5.7: Gráfica encuestas a realizar y tiempos de cómputo CBC I

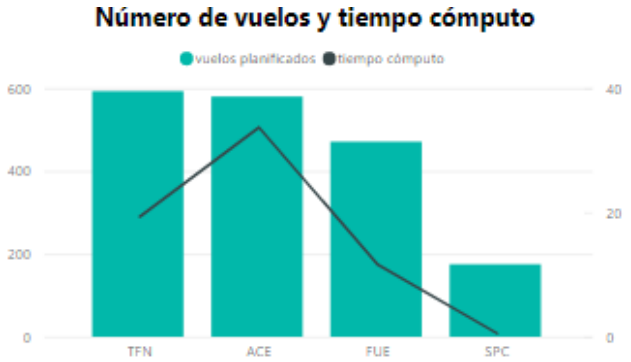


Figura 5.8: Gráfica vuelos planificados y tiempos de cómputo CBC I

- CBC 2

Función objetivo: *Penalizando emplear 1 encuestador*				
Aeropuerto:	Tiempo de cómputo:	Total encuestas 1 encuestador:	Total encuestas 2 encuestadores:	Donaciones:
ACE	18,957 seg	140	7218	32
FUE	11,070 seg	0	7285	0
LPA	-	-	-	-
TFN	12,494 seg	0	5776	10
TFS	425,95 seg	0	8270	
SPC	0,2289 seg	0	3633	0
TOTAL:	468,69 seg	140	32182	42

Tabla 5.4: Resultados experimento CBC II

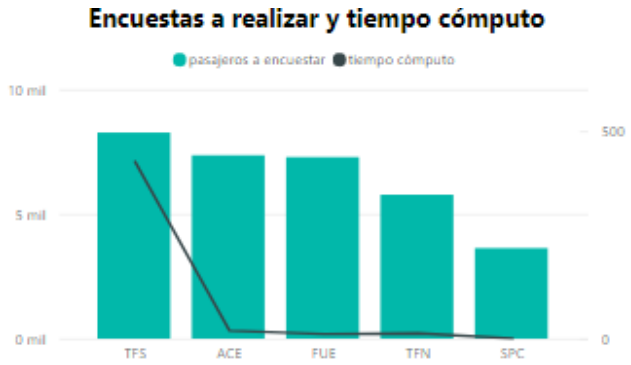


Figura 5.9: Gráfica encuestas a realizar y tiempos de cómputo CBC II

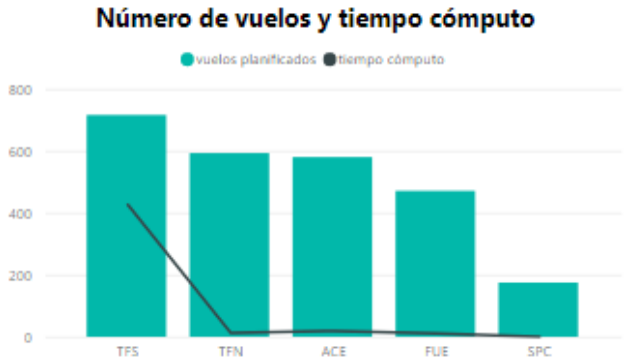


Figura 5.10: Gráfica vuelos planificados y tiempos de cómputo CBC II

Como se puede observar, para el experimento CBC I, no se han obtenido soluciones óptimas para todos los aeropuertos, y esto es debido a que para los aeropuertos de Tenerife Sur y Gran Canaria la búsqueda de la solución óptima llevó más de 10 minutos, y se decidió parar la aplicación. Sin embargo, en el experimento CBC II, sí se encontró solución para el aeropuerto de Tenerife Sur en un tiempo de 7 minutos aproximadamente.

Cabe mencionar que cuando se invoca al solucionador de CBC, no se le pasa ningún parámetro, sin embargo, se ofrecen varias formas de personalizar las decisiones que tome el solucionador, de manera que quizás se pueda obtener una solución óptima más rápido. Entre las personalizaciones que se pueden llevar a cabo están:

- Indicar las variables que deben ramificarse
- Emplear algún tipo de heurística para encontrar soluciones factibles más rápido.
- Seleccionar el siguiente nodo a considerar en el árbol de búsqueda.

Por otro lado, como se puede ver en la **Tabla 5.3** y en la **Tabla 5.4** las donaciones que se requieren son las mismas, y además coinciden con las donaciones requeridas en los experimentos realizados con Gurobi.

Realizando la comparativa del número de encuestas que se obtuvieron para ambos experimentos, e ignorando que para CBC II sí se obtuvo solución en un tiempo razonado para Tenerife Sur, la diferencia entre el número de encuestas a realizar entre ambos experimentos es de 127 encuestas, mientras que la diferencia entre los tiempos de ejecución es de 22,4 segundos.

Por último, con las experiencias realizadas tanto con Gurobi como con CBC, se pueden apreciar las diferencias que presentan un solucionador de pago y uno gratuito, que a grandes rasgos se remite a la capacidad de resolución y los métodos que emplee el propio solucionador para encontrar la solución óptima. Esto conlleva a que en el caso de los experimentos empleando Gurobi si se puedan encontrar soluciones para todos los aeropuertos y para ambas funciones objetivo en tiempos razonables, mientras que para los experimentos realizados con CBC no pueda ser así. En definitiva, Gurobi al ser un solucionador de pago recibe soporte y mantenimiento continuo de la compañía dueña de este producto, mientras que CBC forma parte de un proyecto de software libre y únicamente recibe soporte de la comunidad.

Capítulo 6 **Conclusiones y líneas futuras**

Con la herramienta que se ha implementado el ISTAC puede llevar a cabo un banco de pruebas para planificar la encuesta FRONTUR-CANARIAS, pudiendo responder a la pregunta de qué vuelos encuestar y obtener la solución óptima. Como se expone en el **Capítulo 5** , se pueden dar situaciones en las que dados unos parámetros de entrada y una función objetivo no se encuentre una solución óptima en un tiempo razonable. Esto no quiere decir que no exista solución, dado que gracias al empleo de la variable z_p no se pueden dar situaciones en las que no se encuentre una solución al problema dado, sino que es necesario más tiempo para encontrar la solución óptima.

Hay que tener en cuenta que el tiempo necesario para encontrar la solución óptima no depende de la implementación del modelo ni de la aplicación, sino que viene dado por la complejidad que presentan los problemas de optimización combinatoria y el empleo de métodos exactos para resolver este tipo de problemas. Además, como se vio en el **Capítulo 5** , el empleo de una u otra función objetivo dan soluciones distintas y tiempos de resolución más o menos razonables, por lo que una de las tareas a las que se enfrentan los investigadores de operaciones es buscar funciones objetivo que se ajusten a los requisitos del problema y que, además encuentren la solución óptima en un tiempo moderado.

Por lo tanto, una línea futura puede ser mejorar los tiempos de cómputo mediante el empleo de nuevas funciones objetivo, de tal forma que dependiendo de las necesidades que tenga el ISTAC se puedan implementar en el modelo e incluirlas en la aplicación.

Otra línea futura, puede ser emplear métodos heurísticos para encontrar una solución en un tiempo más razonable, sin embargo, el empleo de algún método que siga una estrategia heurística no aseguraría que la solución obtenida pueda ser la óptima pero sí una aproximada.

Por último, este trabajo ha servido para poner en práctica muchos de los conocimientos que se han visto y aprendido en estos cuatro años de grado, así como otros tantos que han sido necesarios para la realización del proyecto. Además, desarrollar un proyecto como este, ha ayudado a conocer el potencial que generan herramientas de este tipo, donde a través de una aplicación con interfaz gráfica, pueden subyacer implementados modelos matemáticos que den soporte a la toma de decisiones en numerosos campos.

Capítulo 7 **Summary and Conclusions**

The main goal of this project has been completed and ISTAC can use this application to make a testbench and obtain different planning of FRONTUR-CANARIAS. However, there are cases where input parameters and an objective function can arise a conflict situation, where an optimal solution is not found in a reasonable time. Nevertheless, it doesn't mean that there isn't an optimal solution, because with the z_p variable always gets a solution but to find it, more time is needed.

The time required to solve a problem doesn´t depend on the implementation of the model or the application. This time is linked to the complexity of combinatorial optimization problems and the use of exact methods to solve them. As we can see at the chapter 5, the use of an objective function it can provide different solutions and different times, so that one of the most important tasks of operations researchers is to define a good objective functions.

A future line could be to design new objective functions that reduce time and obtain better solutions. These new objective functions can be coded on the application.

Another future line could be to use heuristic methods to find solutions more quickly. However, the use of heuristics doesn’t ensure that the given solution is the optimal solution.

In conclusion, much knowledge learned along these four years of the degree, was needed in this project, as well as other specific skills was needed too. To make a project like this, helps to know the real potential that have the applications which implements optimization models. Besides this type of applications can help to support the decision making in many fields.

Capítulo 8 **Presupuesto**

Para llevar a cabo este proyecto todas las herramientas que se emplearon son de software libre, a excepción del solucionador de Gurobi que se necesitó una licencia académica. Por otro lado, para llevar a cabo las tareas de documentación y consulta, se emplearon libros de la biblioteca de la Escuela Superior de Ingeniería y Tecnología de la Universidad de La Laguna, así como del portal O'Reilly [21] (con acceso a través de la cuenta institucional). A continuación, se detallan los costes asociados al desarrollo de este proyecto:

Tipos	**Descripción**	**Total**
Equipo de trabajo	Portátil MSI PL62 7RC	900 €
Honorarios	400 horas de trabajo (estipulando la hora de trabajo de un ingeniero informático a 20€).	8000 €
Total		8900 €

Tabla 8.1: Resumen de tipos

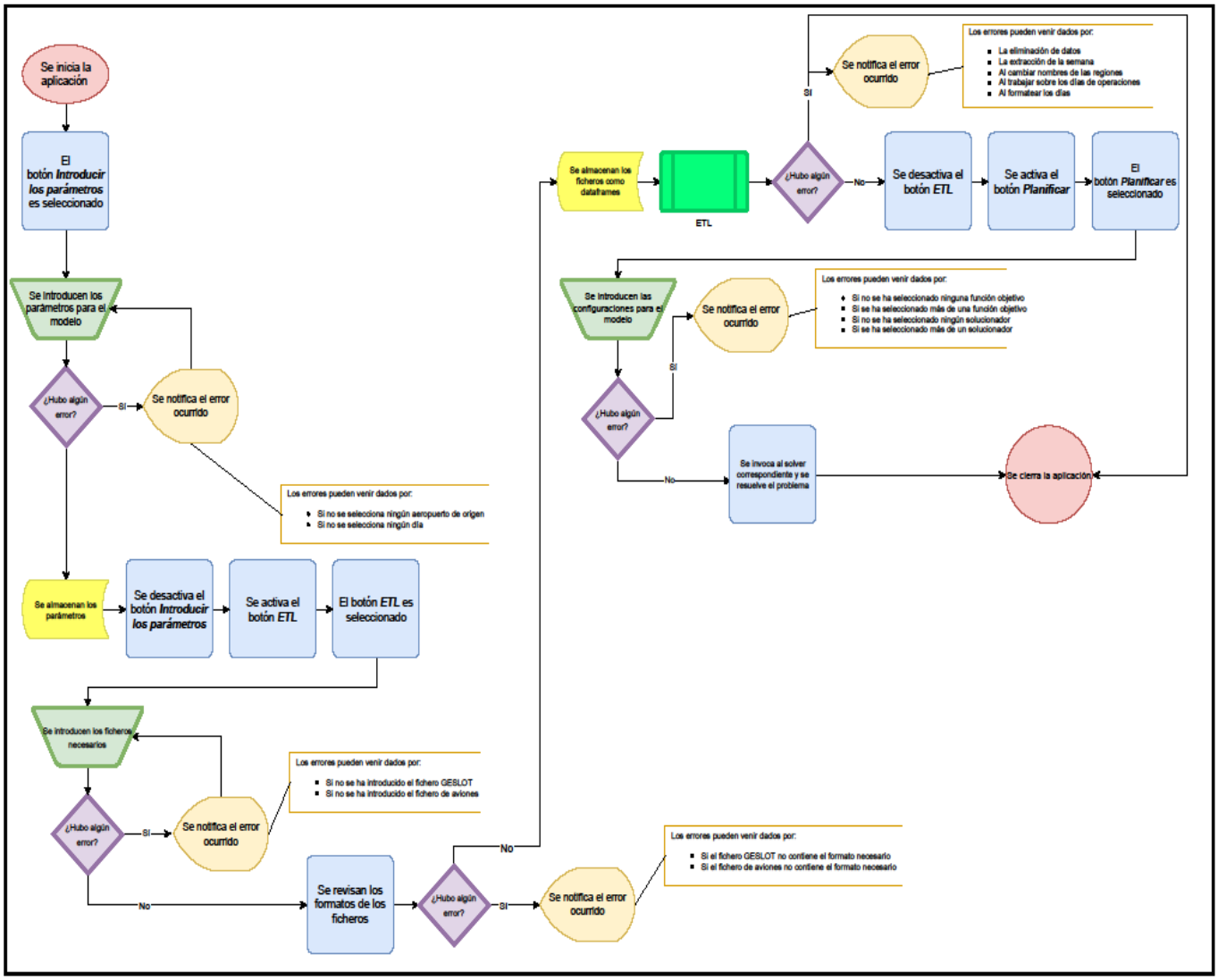
Capítulo 9 **Apéndice**

9.1 Tabla modelos de aviones

Código IATA:	Número asientos:	Modelo:
223	200	Airbus A220-300
290	200	Embraer E190-E2
295	200	Embraer E195-E2
319	141	Airbus A319
320	180	Airbus A320-100/200
321	200	Airbus A321-100/200
32A	220	Airbus A320(sharklets)
32B	220	Airbus A321(sharklets)
32Q	220	Airbus A321neo | LR | XLR
32N	220	Airbus A320neo
32S	220	Airbus A318/319/320/321
332	288	Airbus A330-200
333	295	Airbus A330-300
343	295	Airbus A340-300
717	106	Boeing 717
735	106	Boeing 737-500
73C	189	Boeing 737-300 Winglets
73G	189	Boeing 737-700 | 737-700ER
73H	189	Boeing 737-800 Winglets
73J	189	Boeing 737-900 Winglets
73W	189	Boeing 737-700 Winglets
738	189	Boeing 737-800

7S8	189	Boeing 737
737	189	Boeing 737
734	189	Boeing 737-400
752	239	Boeing 757-200
75T	239	Boeing 757
75W	239	Boeing 757
767	278	Boeing 767
76W	278	Boeing 767
788	200	Boeing 787-8
7M8	200	Boeing 787-8
7M9	200	Boeing 787-9
789	200	Boeing 787-9 pax
AT7	72	Aerospatiale/Alenia ATR72
ATR	72	Aerospatiale/Alenia ATR42 / ATR72
BE4	10	Beechcraft
CRK	50	Canadair Regional Jet
DH4	68	DeHavilland Canada DHC-8-400 Dash8Q
E75	100	Embraer E-175
E90	100	Embraer E-190
E95	122	Embraer E-195
EP3	100	Embraer E-190
GJ5	100	Gulfstream V
H28	100	British Aerospace (HawkerSiddeley)
M83	172	McDonnell Douglas MD83

9.2 Flujograma de la aplicación



9.3 Herramientas utilizadas para el proyecto

Nombre	Tipo	Descripción / Uso
Visual Studio Code	Software	Editor de código empleado para desarrollar la aplicación.
Python	Lenguaje de programación	Lenguaje de programación empleado para para desarrollar la totalidad del código.
tkinter	Librería	Paquete empleado para implementar la interfaz gráfica.
tkcalendar	Librería	Módulo de Python que proporciona un componente calendario para tkinter.
os	Librería	Módulo de Python que proporciona funciones para poder interactuar con el sistema operativo.
Numpy	Librería	Módulo de Python empleado para computación científica.
Pandas	Librería	Módulo de Python empleado para manipular y analizar datos.
datetime	Librería	Módulo de Python empleado para manipular fechas y horas.
functools	Librería	Módulo de Python empleado para trabajar con funciones de orden superior.
threading	Librería	Módulo de Python empleado para trabajar con hilos.
math	Librería	Módulo de Python empleado para trabajar con funciones matemáticas.
pulp	Librería	Módulo de Python mantenido por COIN-OR para implementar modelos de programación lineal.
Coin-or branch and cut	Solucionador	Solucionador de programación lineal gratuito mantenido por la organización de COIN-OR.
Gurobi	Solucionador	Solucionador de programación lineal de pago, que dispone de licencias gratuitas para poder ser utilizadas en el ámbito académico

Bibliografía

[1] BOC - 2018/053, «Gobiernodecanarias.org,» 15 Marzo 2018. [En línea]. Available: http://www.gobiernodecanarias.org/boc/2018/053/010.html. [Último acceso: 9 Junio 2022].

[2] Instituto Canario de Estadística, 2016. [En línea]. Available: http://www.gobiernodecanarias.org/istac/descargas/E16028B/metodologia_FRONTUR.pdf. [Último acceso: 9 Junio 2022].

[3] J. J. S. González, Programación matemática, San Cristóbal de La Laguna: Díaz de Santos, 2001.

[4] E. R. Méndez, PROGRAMACIÓN LINEAL Y ENTERA, España: Sanz y Torres, 2019.

[5] I. Sommerville, Ingeniería de Software, México: Pearson Educacion, 2011.

[6] Microsoft, «Docs Microsoft,» [En línea]. Available: https://docs.microsoft.com/es-es/azure/architecture/data-guide/relational-data/etl. [Último acceso: 23 Junio 2022].

[7] Pandas, «Pandas.pydata,» [En línea]. Available: https://pandas.pydata.org/. [Último acceso: 26 Junio 2022].

[8] Numpy, «Numpy.org,» [En línea]. Available: https://numpy.org/. [Último acceso: 26 Junio 2022].

[9] Google, «Google OR-Tools,» [En línea]. Available: https://developers.google.com/optimization. [Último acceso: 26 Junio 2022].

[10] COIN-OR Organization, «Coin-or.org,» [En línea]. Available: https://www.coin-or.org/. [Último acceso: 27 Junio 2022].

[11] COIN-OR Organization, «PuLP,» [En línea]. Available: https://coin-or.github.io/pulp/. [Último acceso: 27 Junio 2022].

[12] Pytest, «Pytest.org,» [En línea]. Available: https://docs.pytest.org/en/7.1.x/index.html. [Último acceso: 26 Junio 2022].

[13] Python Software Foundation, «Python.org,» [En línea]. Available: https://docs.python.org/es/3/library/pydoc.html. [Último acceso: 25 Junio 2022].

[14] SciPy, «SciPy.org,» [En línea]. Available: https://scipy.org/. [Último acceso: 27 Junio 2022].

[15] Git, «Git-scm,» [En línea]. Available: https://git-scm.com/. [Último acceso: 27 Junio 2022].

[16] GitHub Inc, «GitHub,» [En línea]. Available: https://github.com/. [Último acceso: 27 Junio 2022].

[17] D. Cortesi, «pyinstaller.org,» [En línea]. Available: https://pyinstaller.org/en/stable/index.html. [Último acceso: 26 Junio 2022].

[18] Wikipedia, «Wikipedia.org,» 9 Febrero 2021. [En línea]. Available: https://es.wikipedia.org/wiki/Compilador_cruzado. [Último acceso: 26 Junio 2022].

[19] Microsoft, «Powerbi.microsoft,» [En línea]. Available: https://powerbi.microsoft.com/es-es/. [Último acceso: 27 Junio 2022].

[20] Gurobi, «Gurobi optimization,» [En línea]. Available: https://www.gurobi.com/. [Último acceso: 28 Junio 2022].

[21] O'Reilly, «O'Reilly,» [En línea]. Available: https://www.oreilly.com/. [Último acceso: 28 Junio 2022].