

Treball de Fi de Grau

Grau en Enginyeria en Tecnologia Industrials

Gestión del tráfico aéreo. Modelo de optimización e implementación

MEMÒRIA

Autor: Guillem Carrion Teixidó
Director: Lluís Pérez Vidal
Convocatòria: Juny 2016



Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona



Resumen

El trabajo de fin de grado descrito a continuación desarrolla e implementa una solución para la gestión efectiva del tráfico aéreo. Se ha diseñado un modelo matemático capaz de optimizar las rutas de las aeronaves, así como el coste total de combustible y retrasos. Posteriormente, se ha propuesto una implementación de este modelo mediante herramientas estudiadas en la escuela. Finalmente, para comprobar la funcionalidad de la herramienta, se han diseñado escenarios de prueba para comprobar que los resultados de la simulación son los esperados.

El objetivo principal de este proyecto es básicamente plantear una solución a una problemática muy presente en nuestra sociedad, la gestión del tráfico aéreo. Este estudio podría servir de base para posteriores análisis más profundos, que, junto con el conocimiento necesario, podrían reducir notablemente las demoras en aeropuertos, el coste del transporte e incluso el impacto medioambiental.

Índice

Resumen	3
Índice	5
Lista de Figuras	7
1. Introducción	9
2. Análisis	11
3. Modelo de Optimización	15
3.1. Formulación	15
3.2. Parámetros	16
3.3. Variables de decisión	17
3.4. Función objetivo	18
3.5. Restricciones	19
4. Codificación del modelo	23
4.1. Programario utilizado	23
4.2. Estructura de la implementación	23
4.2.1. Input	23
4.2.2. Modelo	25
4.2.3. Escritura de la solución	27
4.2.4. Interfaz usuario	27
4.2.5. Diagrama flujo de información	29
4.3. Interpretación OPL del modelo matemático	30
4.3.1. Parámetros	30
4.3.2. Variables de decisión	34
4.3.3. Función objetivo	35
4.3.4. Restricciones	37
5. Resultados	41
5.1. Escenario ATM_SCN	41
5.1.1. Estructura	41
5.1.2. Planificación	42
5.1.3. Ejecución	42

5.1.4. Resultados	43
5.1.5. Visualización de los resultados	44
5.2. Escenario ATM_SPAIN	45
5.2.1. Estructura	45
5.2.2. Planificación	45
5.2.3. Ejecución	46
5.2.4. Resultados	47
5.2.5. Generación datos para <i>BlueSky</i>	48
5.2.6. Visualización de los resultados	49
6. Presupuesto	51
Conclusiones	53
Referencias	56
ANEXOS	57
A. Código OPL	57
B. Código VisualBasic	63

Índice de figuras

2.1. Tráfico anual mundial, en trillones de RPK (Revenue Passenger Kilometres)	11
3.1. Típica relación entre $x_{i,k}(t)$ y $z_{i,k}(t)$	20
4.1. Ejemplo de datos de entrada	24
4.2. Gestor de variables de Excel	25
4.3. Archivo OPS. Parámetros de la simulación	26
4.4. Opciones permitidas para el usuario	27
4.5. Ejecución del modelo via línea de comandos	28
4.6. Escritura de la solución	28
4.7. Estructura básica de la implementación	29
5.1. Espacio aéreo escenario ATM_SCN	41
5.2. Planificación escenario ATM_SCN	42
5.3. Botones de ejecución	42
5.4. Confirmación ejecución modelo	43
5.5. Confirmación ruta del programa	43
5.6. Nueva programación para el escenario ATM_SCN	44
5.7. Coste final del escenario ATM_SCN	44
5.8. Representación final para el escenario ATM_SCN	44
5.9. Nodos que conforman el espacio aéreo para el escenario ATM_SPAIN	45
5.10. Planificación escenario ATM_SPAIN	46
5.11. Botones de ejecución	46
5.12. Ejecución vía script del escenario ATM_SPAIN	47
5.13. Nueva programación para el escenario ATM_SPAIN	47
5.14. Estructura de ficheros del simulador <i>BlueSky</i>	48
5.15. Ejecución por comandos de <i>BlueSky</i>	49
5.16. Simulación del escenario ATM_SPAIN mediante <i>BlueSky</i>	49

1. Introducción

El objetivo principal de este trabajo de fin de grado, es el diseño y programación de una herramienta capaz de optimizar y simular rutas de vuelo.

A partir de un estudio exhaustivo del funcionamiento del ATM (Air Traffic Management), se ha programado un modelo de optimización multi-objetivo mediante herramientas estudiadas en la escuela (IBM ILOG CPLEX Optimization Studio).

Cabe decir que, los modelos de optimización multi-objetivos, son aquellos problemas que requieren la optimización simultánea de más de un objetivo. En nuestro caso, principalmente se pretende optimizar, de forma simultánea, el consumo de combustible y los retrasos en los aeropuertos, entre otros.

El trabajo se divide en tres grandes bloques; el análisis de antecedentes y posibles soluciones, la modelización y programación del modelo, y la comprobación de resultados.

En el primer bloque se estudia la situación actual del tráfico aéreo mundial y los hechos que han motivado este proyecto. Como se explicará más adelante, debido al aumento exponencial del transporte aéreo, es de primordial importancia disponer de una metodología eficaz y óptima que regule el tráfico aéreo.

En el segundo bloque, separado en dos secciones, se propone un modelo capaz de optimizar el tráfico aéreo, disminuyendo retrasos y costes. En la primera sección se formula (en lenguaje matemático) el modelo, mientras que en la segunda sección se explica cómo se ha codificado.

En el tercer bloque, en el último bloque se analiza la validez de los resultados en dos escenarios distintos, el segundo de los cuales pretende imitar a la realidad y se simula mediante la herramienta *BlueSky*.

Por último, se estima un presupuesto en caso de poner en funcionamiento real este proyecto.

2. Análisis

El transporte aéreo mundial ha aumentado durante las últimas décadas y se espera que siga creciendo en el futuro. De hecho, en los últimos diez años, este sector ha experimentado un crecimiento del 62 %, un 5,8 % anual en los últimos cinco. Se espera que su crecimiento sea del 4,6 % anual en los próximos veinte años [1].

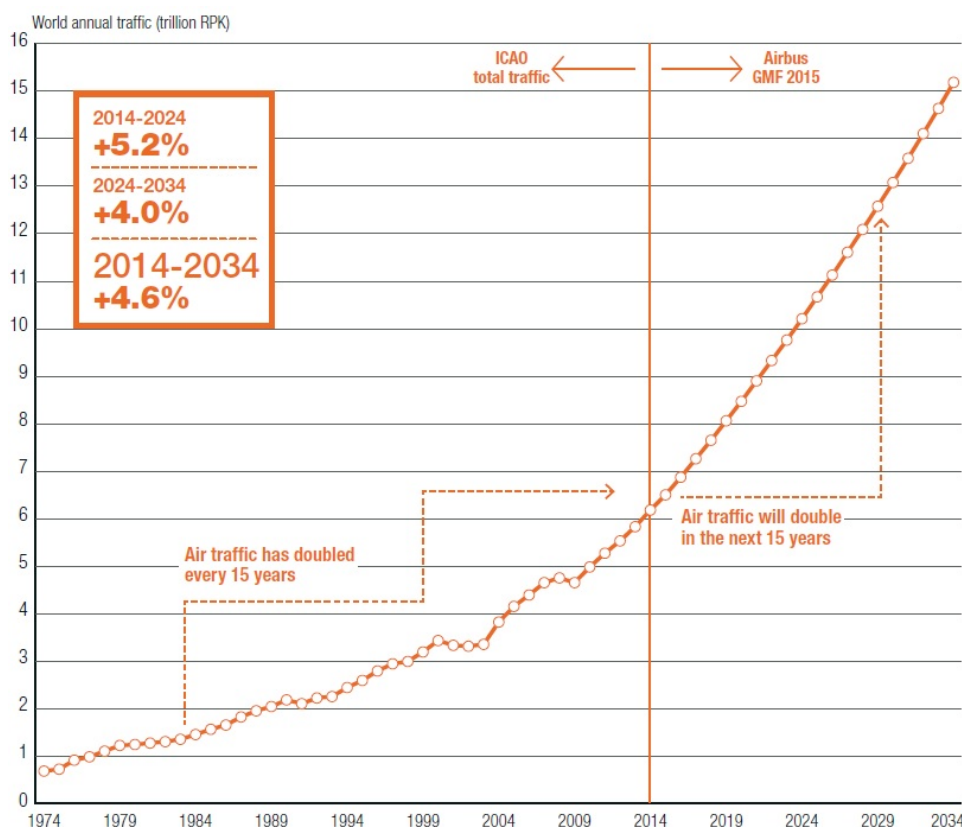


Figura 2.1: Tráfico anual mundial, en trillones de RPK (Revenue Passenger Kilometres¹)

La evolución tecnológica del sector y la globalización han provocado un abaratamiento importante de los costes. En consecuencia, ha aumentado significativamente la demanda de recursos al sistema de espacio aéreo.

Durante los últimos años, se han investigado y desarrollado técnicas de gestión dinámica para poder balancear la creciente demanda con los recursos limitados del espacio aéreo global. Mediante este tipo de gestión dinámica, se puede ampliar la capacidad del espacio aéreo, maximizando la eficiencia y flexibilidad de este. Esta gestión se caracteriza por ser centralizada, donde los distintos usuarios solicitan ventanas temporales para el uso del espacio aéreo. En general, los dos tipos de usuarios son los civiles y los militares y, en

¹EL RPK es el número total de kilómetros recorridos por todos los pasajeros.

muchos países, existen partes del espacio aéreo reservado solo al uso militar. Cuando hay un aumento temporal de la demanda de tráfico aéreo o una disminución de la capacidad de este, los controladores del tráfico aéreo civil deben solicitar rutas adicionales de los militares.

Actualmente ya existen modelos y algoritmos para la Gestión del Flujo del Tráfico Aéreo (ATFM, en inglés) con el objetivo de proponer una mejor utilización del espacio aéreo, a continuación, algunos ejemplos:

- Vossen & Michael, 2006 [2]; Modelo para reducir retrasos en los aeropuertos.
- Richetta & Odoni, 1993-1994 [3]; Modelo ATFM con restricciones en la capacidad de los aeropuertos.
- Bertsimas & Patterson, 1998-2000 [4], Lulli y Odoni, 2007 [5]; Modelo ATFM con restricciones simultaneas en la capacidad de los aeropuertos y sectores.

En todos estos estudios, la estructura del espacio aéreo se considera determinista e inmutable, es decir, aerovías², waypoints³, rutas, etc son estáticas y no se pueden modificar. Además, todos ellos siguen un procedimiento similar, ajustan dinámicamente y de forma optimizada los planes de vuelo mediante desvíos o retrasos en pista. De esta forma, se pueden afrontar posibles disminuciones de la capacidad del espacio aéreo, minimizando costes y esperas de los clientes. Aun así, pocos estudios se han centrado en cómo reconstruir una red de tráfico aéreo eficiente y cómo poder ajustar la red actual dinámicamente. En este trabajo se plantea un modelo y algoritmo que permite tomar decisiones dinámicas sobre cuándo y cómo ajustar la estructura del espacio aéreo, buscando el mínimo coste. Se introducen elementos de control para la apertura y cierre de segmentos de rutas aéreas además de la incorporación de nuevas restricciones, como el menor tiempo de ocupación para estas secciones temporales.

Finalmente, otro problema a tener en cuenta son las incertidumbres en la proyección futura del espacio aéreo, ya que dificultan en gran medida la gestión del mismo. Las metodologías actuales, impiden que los controladores obtengan una representación precisa de la futura situación del espacio aéreo y aumenta su carga de trabajo al producirse acontecimientos impredecibles con escaso tiempo de reacción. Es por ese motivo que el modelo se basa en la planificación durante la supuesta fase estratégica de control del

² Es el camino virtual predefinido (tanto en altura como en trazado) que sigue una aeronave que sale desde un punto A hasta un punto B.

³ Los waypoints son coordenadas para ubicar puntos de referencia tridimensionales utilizados en la navegación basada en GPS (global positioning system: sistema de posicionamiento global).

tráfico aéreo. Durante esta fase, una entidad centralizada programa los vuelos en puntos temporales a fin de reducir al mínimo algunos criterios de complejidad espacial o los retrasos en las llegadas de los vuelos.

La idea básica del modelo planteado es generar la programación de los vuelos de forma global i centralizada para así poder facilitar el trabajo de los controladores aéreos en la fase táctica, teniendo en mente la estructura dinámica del espacio aéreo. Debido a que el tiempo de reacción y rectificación es menor en la fase táctica que en la estratégica, es vital que sea en esta última dónde haya que minimizar el impacto de las futuras incertidumbres⁴.

⁴Estas incertidumbres pueden tener distintos orígenes, como variación en las corrientes de aire, resolución de conflictos entre aeronaves o incluso peticiones de cambio de ruta ante fuertes condiciones meteorológicas adversas.

3. Modelo de Optimización

Dado una red de tráfico aéreo compuesto por aeropuertos, waypoints, aerovías, rutas y un conjunto de vuelos planeados para volar en esta red, el modelo de optimización planteado toma decisiones dinámicas sobre cuándo y cómo ajustar la estructura de esta. Este ajuste se materializa mediante la apertura y cierre de aerovías temporales para así minimizar el coste total, incluyendo el coste de apertura y el coste de uso para estas aerovías, al igual que el coste de los retrasos para todos los aviones.

3.1. Formulación

Para poder implementar las mejoras mencionadas, hay que disponer de una base firme donde aplicar las condiciones. Por ese motivo, el modelo se construye sobre un conjunto de premisas y elementos que pretenden reproducir la realidad.

Para empezar, consideramos una red de tráfico aéreo \mathcal{R} , que contiene el conjunto de nodos \mathcal{N} y el conjunto de aerovías \mathcal{A} ; $\mathcal{R}(\mathcal{N}, \mathcal{A})$. El conjunto de nodos representa los aeropuertos y waypoints existentes, por otro lado, el conjunto de aerovías representa las conexiones entre los distintos nodos, ya sean fijas o temporales (se pueden usar temporalmente). La agrupación de nodos y aerovías nos proporciona la base sólida necesaria para poder definir parámetros y condiciones de control.

El modelo es del tipo *Multi-commodity flow problem*, o en español *Problema del flujo multi-producto*. Este tipo de problemas de flujo se caracterizan por tener distintos productos que se trasladan entre distintos nodos de origen y destino [6]. A continuación, las suposiciones a tener en cuenta para el modelo que nos ocupa.

- Todos los vuelos con mismo origen y destino pertenecen a la misma ruta.
- Dado que el valor de cada ruta es el número de vuelos, debe ser un entero no negativo.
- Todos los vuelos tienen que aterrizar en su origen en algún momento.
- Los aviones tienen las mismas características mecánicas, por lo que vuelan a la misma velocidad y recorren una determinada aerovía con el mismo tiempo. Al mismo tiempo, consumen el mismo combustible por cada km recorrido.
- Las capacidades de las aerovías son conocidas para todo el espectro temporal del modelo.

- Si se abre una aerovía temporal, debería permanecer abierta un mínimo de tiempo establecido.

3.2. Parámetros

A continuación, los parámetros del modelo definidos.

\mathcal{A}	conjunto de arovías
\mathcal{A}_{tmp}	conjunto de arovías que pueden ser abiertas temporalmente
\mathcal{K}	conjunto de rutas de vuelo
\mathcal{N}	conjunto total de nodos
\mathcal{N}_{orig}	conjunto de nodos de origen
\mathcal{N}_{dest}	conjunto de nodos de destino
\mathcal{N}_{mid}	conjunto de nodos intermedios, distintos a origen y destino
i	índice de nodo
(i, j)	índice de arovía
t	índice de intervalo de tiempo
k	índice de ruta de vuelo
$dis_{i,j}$	longitud de la aerovía (i, j)
$a_{i,j}$	tiempo de vuelo para aerovía (i, j)
$c_{i,j}(t)$	nº máx de vuelos que pueden entrar en la aerovía (i, j) en el intervalo t
$U_{i,j}(t)$	tiempo mínimo de ocupación para la aerovía (i, j)
$P_i^k(t)$	nº de vuelos programados de la ruta k , con origen i , en el intervalo t
$w_{in}(i)$	nº máximo de arovías entrantes para nodo i
$w_{out}(i)$	nº máximo de arovías salientes para nodo i
r	coste de volar 1 km
$d_i(t)$	coste de retrasar un vuelo al nodo de origen i en el intervalo t
$o_{i,j}(t)$	<i>opening fee</i> para aerovía temporal (i, j) en el intervalo t
$m_{i,j}(t)$	<i>usage fee</i> para aerovía temporal (i, j) en el intervalo t

3.3. Variables de decisión

Las variables representan las decisiones que se pueden tomar para afectar el valor de la función objetivo. Se clasifican en dos grupos, variables independientes (principales, de control) y variables dependientes (secundarias, de estado). Como bien indica el nombre, la finalidad es poder impactar la función objetivo mediante la variación de las variables de control. Paralelamente, y con menor importancia, las variables de estado o auxiliares presentes en la función objetivo y/o restricciones también se verán impactadas.

En el caso estudiado, podríamos afirmar que las tres variables introducidas son de control, aunque con connotaciones distintas; $x_{i,j}(t)$, $z_{i,j}(t)$ booleanas y $f_{i,j}^k(t)$ entera positiva.

$$x_{i,j}(t) = \begin{cases} 1, & \text{si la aerovía (i, j) se usa en intervalo t.} \\ 0, & \text{en caso contrario} \end{cases}$$

$$z_{i,j}(t) = \begin{cases} 1, & \text{si la aerovía (i, j) se abre en intervalo t.} \\ 0, & \text{en caso contrario} \end{cases}$$

$$f_{i,j}^k(t) \in \mathbb{Z}^+$$

Los conjuntos de variables $x_{i,j}(t)$ y $z_{i,j}(t)$ son bidimensionales. Cada uno de estos dos conjuntos, se extienden en dos dimensiones; aerovía y tiempo. Existen tantas variables en cada conjunto, como combinaciones de aerovías e intervalos de tiempo. Por un lado, La variable $z_{i,j}(t)$ tomará el valor 1 solo en el instante que se abra la aerovía. Por otro lado, $x_{i,j}(t)$ tomará el valor 1 cuando la aerovía esté en uso. Por ejemplo,

$$\begin{aligned} z_{A,B}(10) &= 1 & x_{A,B}(10) &= 1 \\ z_{A,B}(11) &= 0 & x_{A,B}(11) &= 1 \end{aligned}$$

Nos indica que la aerovía (A, B) se abre en el instante 10, y que sigue en uso en el instante 11. Cabe mencionar que la variable $x_{i,j}(t)$ aplica a todas las aerovías, por el otro lado, la variable $z_{i,j}(t)$ solo aplica a las aerovías temporales.

Finalmente, el conjunto de variables $f_{i,j}^k(t)$ se extiende por tres dimensiones; aerovía, ruta, tiempo. En consecuencia, hay tantas variables en este conjunto como combinaciones de las tres dimensiones. El valor de $f_{i,j}^k(t)$ define el n° de vuelos de la ruta k entrando en

la aerovía (i, j) en el intervalo de tiempo t . Por ejemplo,

$$f_{A,B}^{A \Rightarrow Z}(10) = 1$$

Nos indica que el total de aviones de la ruta $A \Rightarrow Z$ que recorren la aerovía (A, B) en el instante 10 es igual a 1.

Esta variable será que finalmente nos proporcione la información necesaria para obtener la programación definitiva, ya que nos indica qué vuelos salen de los nodos origen y a qué intervalo de tiempo.

3.4. Función objetivo

Conceptualmente, la función objetivo se define como la medida cuantitativa del funcionamiento del sistema que se quiere optimizar, ya sea para maximizar o minimizar. En el caso que nos ocupa, la función objetivo representa el coste total, y el objetivo es minimizarlo. Concretamente, la solución óptima pasará por aquel conjunto de valores que deben tomar las variables, tal que minimicen el valor de la función objetivo, siempre y cuando se cumplan las restricciones. Matemáticamente podemos expresarla de la siguiente forma:

$$\begin{aligned} \min Z = & \sum_{t,k,i \in N_{orig}} d_i(t) \cdot f_{i,i}^k(t) + \sum_{t,k,i \in A} r \cdot dis_{i,j} \cdot f_{i,j}^k(t) \\ & + \sum_{t,(i,j) \in A_s} o_{i,j}(t) \cdot z_{i,j}(t) + \sum_{t,(i,j) \in A_s} m_{i,j}(t) \cdot x_{i,j}(t) \end{aligned} \quad (3.1)$$

Como se puede observar, la función objetivo está compuesta por cuatro términos diferenciados. A continuación, una explicación ordenada de cada uno de estos elementos.

- El primer término representa el coste de todos los retrasos en los nodos de origen. En este caso, $f_{i,i}^k(t)$ es el total de vuelos que pasan del nodo i al nodo i . La aerovía (i, i) empieza y termina en el mismo nodo de origen i , por lo que tiene un tiempo de vuelo unitario y longitud 0, es decir, su salida se retrasa un intervalo de tiempo. Cada nodo de origen (o aeropuerto) tiene un coste determinado para cada avión que se retrasa una unidad de tiempo. Por ese motivo, se pondera el coste total mediante el parámetro $d_i(t)$.
- El segundo término representa el coste total de vuelo propiamente dicho. Se calcula multiplicando el coste de volar un km (r), por la distancia entre los nodos ($dis_{i,j}$) y por la variable $f_{i,j}^k(t)$. Al multiplicar por esta última variable, solo se computa el coste de los aviones que realmente están en vuelo.

- El tercer término representa el coste de apertura de las aerovías temporales. Se calcula multiplicando el coste de apertura de cada aerovía ($o_{i,j}(t)$) por las veces que se abre esta misma ($z_{i,j}(t)$).
- El cuarto término representa el coste de mantener abiertas las aerovías temporales. Se calcula multiplicando el coste de mantener cada aerovía temporal en funcionamiento durante un intervalo de tiempo ($m_{i,j}(t)$) por el tiempo total que permanece abierta ($x_{i,j}(t)$).

3.5. Restricciones

Las restricciones representan un conjunto de relaciones expresadas mediante ecuaciones e inecuaciones que determinadas variables deben cumplir. En el modelo descrito, las restricciones son las siguientes:

$$\sum_{j:(i,j) \in \mathcal{A}} f_{i,j}^k(t) - \sum_{j:(j,i) \in \mathcal{A}} f_{i,j}^k(t - a_{j,i}) = P_i^k(t) \quad \forall i \in \mathcal{N}_{orig}, t \in \mathcal{T}, k \in \mathcal{K} \quad (3.2)$$

$$\sum_{j:(i,j) \in \mathcal{A}} f_{i,j}^k(t) - \sum_{j:(j,i) \in \mathcal{A}} f_{i,j}^k(t - a_{j,i}) = 0 \quad \forall i \in \mathcal{N}_{mid}, t \in \mathcal{T}, k \in \mathcal{K} \quad (3.3)$$

$$\sum_{t,j:(j,i) \in \mathcal{A}} f_{j,i}^k(t) - \sum_{t,q \in \mathcal{N}_{orig}} P_i^k(t) = 0 \quad \forall i \in \mathcal{N}_{dest}, k \in \mathcal{K} \quad (3.4)$$

$$\sum_{k \in \mathcal{K}} f_{i,j}^k(t) \leq c_{i,j}(t) \cdot x_{i,j}(t) \quad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T} \quad (3.5)$$

$$U_{i,j}(t)[z_{i,j}(t) - (x_{i,j}(t) - x_{i,j}(t-1))] \leq \sum_{\tau \in (t-U_{i,j}, t)} x_{i,j}(\tau) \quad \forall (i,j) \in \mathcal{A}_{tmp}, t \in \mathcal{T} \quad (3.6)$$

$$x_{i,j}(t) - x_{i,j}(t-1) \leq z_{i,j}(t) \quad \forall (i,j) \in \mathcal{A}_{tmp}, t \in \mathcal{T} \quad (3.7)$$

$$\sum_j x_{j,i}(t) \leq w_{in}(i) \quad \forall i \in \mathcal{N}_{mid}, t \in \mathcal{T} \quad (3.8)$$

$$\sum_j x_{i,j}(t) \leq w_{out}(i) \quad \forall i \in \mathcal{N}_{mid}, t \in \mathcal{T} \quad (3.9)$$

$$x_{i,j}(t) \in \{0, 1\}, \quad z_{i,j}(t) \in \{0, 1\}, \quad f_{i,j}^k(t) \in \mathcal{Z}^+ \quad (3.10)$$

Las restricciones 3.2, 3.3, 3.4 aseguran la conservación del flujo.

- La restricción 3.2 indica que todos los vuelos (de la misma ruta) que despegan de un nodo de origen i en un cierto intervalo de tiempo t ($\sum_{j:(i,j) \in \mathcal{A}} f_{i,j}^k(t)$) pueden ser; o vuelos programados para despegar de este nodo en este intervalo t ($P_i^k(t)$) o vuelos retrasados que están esperando en este nodo ($\sum_{j:(j,i) \in \mathcal{A}} f_{i,j}^k(t - a_{j,i})$).

- La restricción 3.3 indica que todos los vuelos (de la misma ruta) que salen de un nodo intermedio i en un intervalo de tiempo t , $(\sum_{j:(i,j) \in \mathcal{A}} f_{i,j}^k(t))$, es igual al total de vuelos entrantes en este mismo intervalo $(\sum_{j:(j,i) \in \mathcal{A}} f_{j,i}^k(t - a_{j,i}))$. Ciertamente, el total de vuelos entrantes al nodo i es equivalente al sumatorio de vuelos salientes de los nodos adjuntos, con dirección a i y que salieron durante el intervalo $t - a_{j,i}$ (ya que cada adjunto está a una distancia y tiempo diferente).
- La restricción 3.4 indica que todos los vuelos que despegan de los nodos origen para cada ruta $(\sum_{t,q \in \mathcal{N}_{orig}} P_i^k(t))$, deben aterrizar en sus nodos de destino $(\sum_{t,j:(j,i) \in \mathcal{A}} f_{j,i}^k(t))$.
- La restricción 3.5 limita el número total de vuelos para cada aerovía e intervalo de tiempo.
- La restricción 3.7 determina la relación entre la variable $x_{i,j}(t)$ y $z_{i,j}(t)$. Esta restricción, combinada con la función objetivo, fuerza a la variable $z_{i,j}(t)$ a valer 1 cuando $x_{i,j}(t)$ pasa de 0 a 1.
- Las restricciones 3.8 y 3.9 limitan el número de aerovías entrando y saliendo para cada nodo. De esta manera evitamos la sobrecarga de trabajo y complejidad para los controladores aéreos.

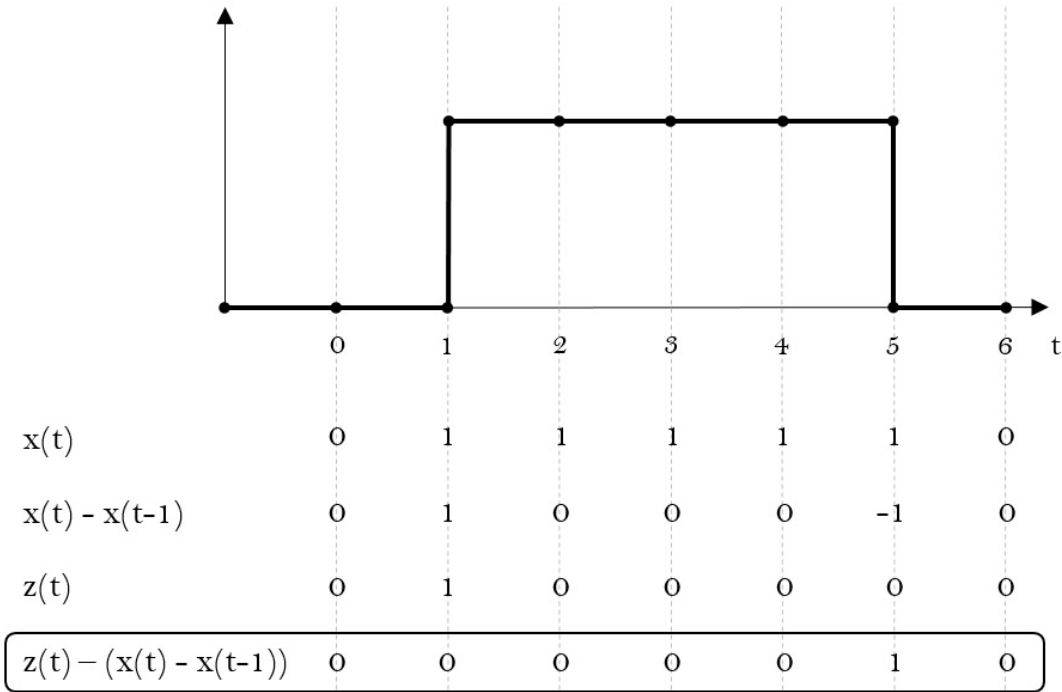


Figura 3.1: Típica relación entre $x_{i,k}(t)$ y $z_{i,k}(t)$

- La restricción 3.10 indica las propiedades para las variables; enteras y no negativas.
- Finalmente, la restricción 3.6 tiene una comprensión compleja. Su función es asegurar que se usa cada aerovía temporal durante un mínimo de tiempo en caso de ser abierta. Para entender su funcionamiento, partimos de la condición ideal suponiendo que nos encontramos justo en el intervalo de tiempo en que se cierra una aerovía:

$$U_{i,j}(t) \leq \sum_{\tau \in (t-U_{i,j}, t)} x_{i,j}(\tau)$$

Recorremos la ventana temporal desde $t-U_{i,j}$ hasta t sumando todos los instantes en que la aerovía ha permanecido abierta ($\sum_{\tau \in (t-U_{i,j}, t)} x_{i,j}(\tau)$). Ese valor debe ser mayor o igual al mínimo tiempo de ocupación ($U_{i,j}(t)$). Esta condición solo debe aplicarse en el preciso intervalo de tiempo en que se cierra la aerovía, para ello, recorreremos todos los intervalos de tiempo y añadimos el termino $[z_{i,j}(t) - (x_{i,j}(t) - x_{i,j}(t-1))]$ multiplicando la parte restrictiva de la condición ($U_{i,j}(t)$). Este término toma el valor 1 cuando la aerovía se cierra en ese intervalo de tiempo, y 0 si no es el caso. En la figura 3.1 podemos ver un ejemplo de su comportamiento.

4. Codificación del modelo

4.1. Programario utilizado

El programa utilizado para la codificación, implementación y solución del modelo matemático es el *IBM © ILOG © CPLEX © Optimization Studio*. Se caracteriza por ser una “suit” de herramientas de soporte analítico, diseñado para el rápido desarrollo e implementación de modelos de optimización utilizando programación matemática. Combina un Entorno de Desarrollo Integrado (IDE, Integrated Development Environment) con el potente *Optimization Programming Language (OPL)* y los motores *ILOG CPLEX*, de alto rendimiento, para resolver los modelos.

El lenguaje utilizado, *Optimization Programming Language (OPL)*, permite una descripción matemática natural de los modelos de optimización. Al utilizar una sintaxis de alto nivel para los modelos matemáticos, produce código considerablemente más simple y más corto que los lenguajes de programación de propósito general, lo que reduce el esfuerzo y la mejora de la fiabilidad en el desarrollo de aplicaciones, actualizaciones y mantenimiento. Su potente sintaxis es compatible con todas las expresiones necesarias para modelizar y resolver problemas matemáticos.

A modo de interfície, se usa el programa *Microsoft Excel ©*, añadiendo piezas de código mediante *Visual Basic ©*.

4.2. Estructura de la implementación

Cómo hemos visto en la sección 3.1, el modelo busca una solución (output) a partir de unas entradas (input), optimizando una función objetivo y cumpliendo las restricciones indicadas. Esta distribución del flujo de información, nos permite diferenciar tres bloques, input, modelo y output.

4.2.1. Input

Con el fin de facilitar la interacción del usuario con la herramienta, se centraliza toda la información de entrada en un mismo libro Excel®. Los datos se clasifican en distintas hojas de cálculo, coincidiendo con la estructura teórica del modelo:

1. Estructura del espacio aéreo estudiado:
 - Listado de nodos. Hoja *NODOS*.

- Total de aerovías (Hoja *AEROVIAS*) y aerovías temporales (Hoja *AEROVIAStmp*). Para cada aerovía se especifica: Origen, destino, distancia y tiempo de recorrido.

The screenshot displays three sheets from an Excel model:

- RESUMEN Sheet:** Contains simulation parameters and results.

START_SIM	END_SIM	TOTAL_VUELOS	TOTAL_INCIDENCIAS	START_SIM_INT	END_SIM_INT
9:01:00	11:00:00	15	1	1	120

 Below this, there are buttons for 'Ejecutar modelo', 'Extraer información', and 'Crear Escenario'. To the right, a table shows costs:

Cost Type	Value
FlightDistCost	208.255
FlightDelayCost	0
OpeningTempAirways	0
UsingTempAirways	0
Total	208.255
- PLANIFICACION Sheet:** Contains a list of flights.

COMPANÍA	TIPO AERONAVE	ORIGEN	DESTINO	VUELO	TAKEOFF	NEW TAKEOFF
IBERIA	A-320	BCN	JRZ52	IB1234	9:01:00	9:01:00
VUELING	A-320	BCN	ALT70	VU1222	9:02:00	9:02:00
RYANAIR	A-320	BRA01	IBZ45	RY8634	9:03:00	9:03:00
SWISS AIRLINES	A-320	ALT	MHN	SW8945	9:04:00	9:04:00
BRITISH AIRLINES	A-320	BRA01	MHN	BR5543	9:05:00	9:05:00
IBERIA	A-320	SSN	ALT70	IB3032	9:06:00	9:06:00
VUELING	A-320	AOG	MLG25	VU9999	9:07:00	9:07:00
- INCIDENCIAS Sheet:** Contains a list of incidents.

ORIGEN	DESTINO	TIEMPO INT	TIEMPO	MAX VUELOS
FINAM	LOMDA	1	9:01:00	0

Figura 4.1: Ejemplo de datos de entrada

2. Limitaciones del espacio aéreo:

- Listado de incidencias, hoja *INCIDENCIAS*. Para cada incidencia se especifica: Origen, destino, cuándo ocurre la incidencia, número máximo de aeronaves permitidas. Ver figura 4.1.

3. Planificación inicial:

- Listado de vuelos programados, hoja *PLANIFICACION*. Para cada vuelo programado se especifica: Origen, destino, hora programada del despegue. Ver figura 4.1.

4. Detalles simulación, hoja *RESUMEN*. En esta hoja se recompila automáticamente información necesaria para el modelo; número total de vuelos, incidencias y cuando empieza y termina la simulación. Ver figura 4.1.

4.2.2. Modelo

El modelo matemático está implementado íntegramente dentro del programa *IBM © ILOG © CPLEX © Optimization Studio*, usando el lenguaje de programación *OPL*. Pero, además del modelo, hay distintos módulos que lo rodean y que le sirven de interfaz con otros sistemas. Podemos diferenciar tres partes diferenciadas, coincidiendo con los distintos archivos que conforman el modelo:

1. Adaptación datos de entrada. Archivo “.dat”.

Este módulo hace de puente entre la hoja de cálculo dónde está recopilada la información de entrada y el modelo. En el archivo .dat se indica la localización de la información para cada parámetro del modelo. Para facilitar el mantenimiento de la herramienta, el archivo contiene referencias a variables de Excel, no al rango en concreto (ver código 24 en el Anexo A). Con este paso intermedio, ponemos en práctica el *Softcoding*⁵. Estas variables están definidas en las hojas de cálculo Excel, donde se les asigna un rango dinámico, más fácil de mantener y manipular (ver figura 4.2).

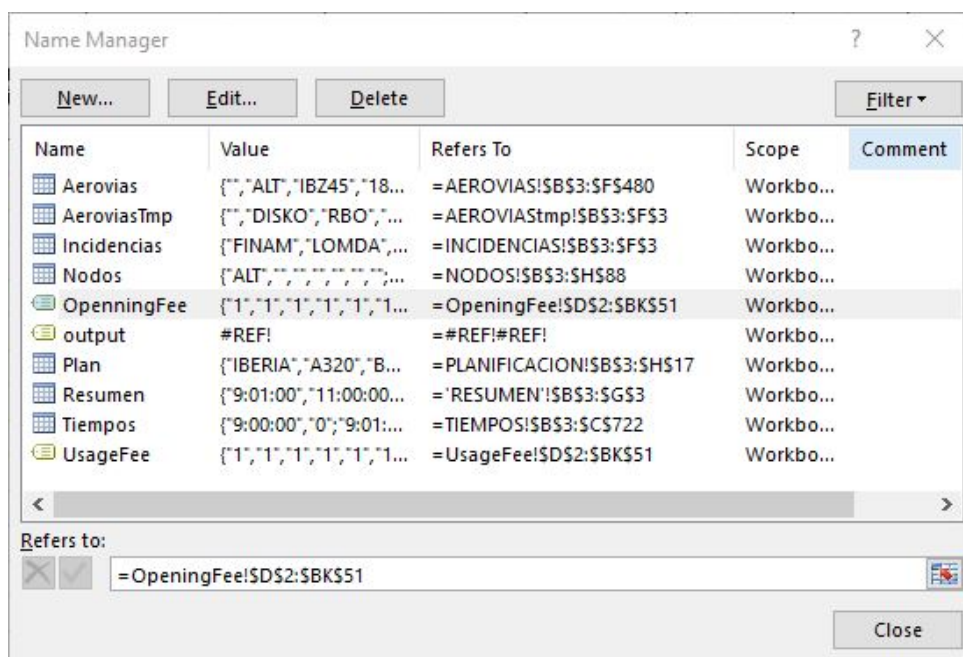


Figura 4.2: Gestor de variables de Excel

⁵*Softcoding* es un término usado en programación que hace referencia a la metodología usada para obtener el valor de una función a partir de un recurso externo. Es el opuesto al *Hardcoding*, codificar valores y funciones directamente en el código origen.

2. Generación del modelo matemático y búsqueda de la solución óptima. Archivos “.mod” y “.ops”.

El modelo matemático en sí, se encuentra codificado en el archivo “.mod”. Por otro lado, los detalles de la simulación se encuentran en el archivo “.ops”, donde podemos escoger la metodología usada para resolver el problema de optimización, de entre las siguientes opciones:

- Simplex primario
- Simplex dual
- Simplex de red
- Barrera
- Tamizado
- Concurrente dual, de barrera y primario

En nuestro caso dejaremos el valor por defecto *Automático* (ver figura 4.3).

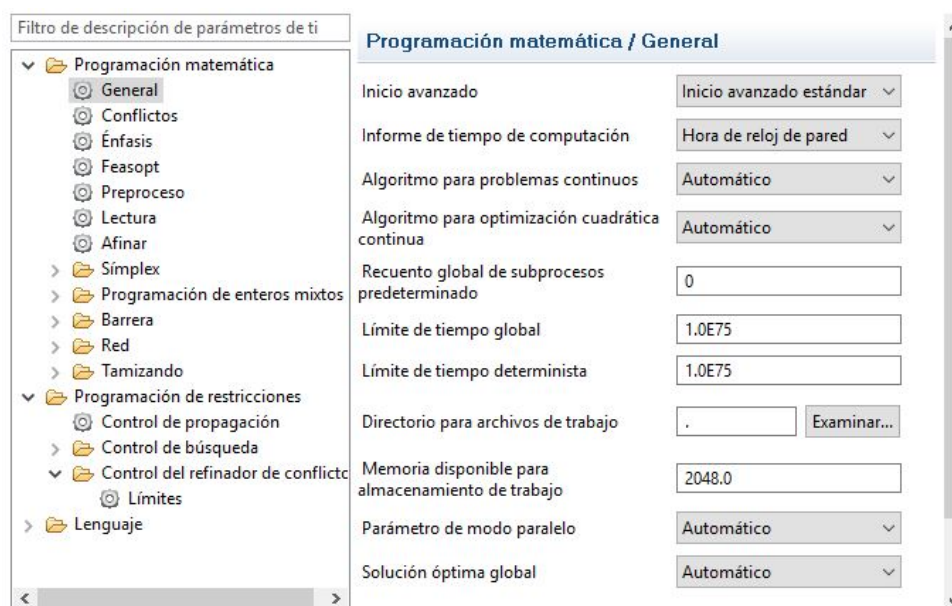


Figura 4.3: Archivo OPS. Parámetros de la simulación

De la misma forma podemos configurar el límite de tiempo y la memoria utilizada. Más adelante, en la sección 4.3 se explica con detalle la programación del modelo, aunque podemos ver el archivo completo en el Anexo A (Código 25).

4.2.3. Escritura de la solución

Existe una parte del código en el archivo “.mod”, dedicado a exportar la solución obtenida a unos ficheros de texto “sol.out” y “Coste.out”. La parte de código dedicado a la exportación de la información se encuentra en el Anexo A (Código 26).

4.2.4. Interfaz usuario

En el mismo archivo Excel dónde se encuentra la información de entrada descrita en la sección 4.2.1. En la hoja *RESUMEN* podemos ver tres botones que permiten al usuario interactuar con la herramienta. La funcionalidad de cada uno de estos botones está programada mediante *Visual Basic* ©.

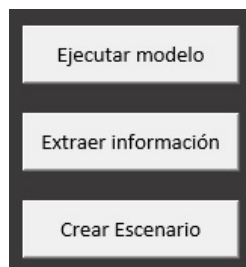


Figura 4.4: Opciones permitidas para el usuario

A continuación, la función de cada uno de estos botones:

- Ejecutar Modelo.

Permite al usuario lanzar un script que ejecuta automáticamente el modelo, junto con sus detalles de ejecución. El código VB de la macro se encuentra en el Anexo B (Código 27). Este código permite al usuario confirmar si realmente quiere lanzar la macro, y si es así, donde se encuentra la ruta del programa *IBM © ILOG © CPLEX © Optimization Studio*. Entonces genera variables con las rutas correspondientes y ejecuta el script: “oplrn -p” junto con los parámetros de ejecución. Este script permite lanzar modelos vía Script Shell.

Figura 4.5: Ejecución del modelo via línea de comandos

■ Extraer Información

Como podemos ver en la figura 4.6 extrae información de los archivos “sol.out” y “Coste.out” para introducirla en el mismo Excel. El código VB de la macro se encuentra en el Anexo B (Código 29). Esta macro lee los archivos sol.out” y “Coste.out”, para interpretarlos. De “Coste.out” extrae el coste total y actualiza su valor en la pestaña “RESUMEN”. De “sol.out” saca los nuevos horarios y rutas para actualizar su valor en la hoja “PLANIFICACION”.

TAKEOFF	NEW TAKEOFF
9:01:00	9:01:00
9:02:00	9:02:00
9:03:00	9:03:00
9:04:00	9:04:00
9:05:00	9:05:00
9:06:00	9:06:00
9:07:00	9:07:00

Ruta:
 ALT-VLC50-IBA-BASSO-MHN
 Hora aproximada:
 9:48:00

Figura 4.6: Escritura de la solución

■ Crear Escenario

Genera un archivo de texto adaptable a simuladores ATM como *BlueSky*. Parecida a la macro *ExtraerInfo*, lee el archivo “sol.out” y genera un nuevo archivo plano de texto, “inputBS.scn”. El código VB de la macro se encuentra en el Anexo B (Código 28).

```

1  0:00:00.00>CRE IB1234, B744, 41.3071, 2.1078, 0, 0, 300
2  0:00:00.00>ADDWPT IB1234, 41.3071, 2.1078, 15000, 300
3  0:00:00.00>ADDWPT IB1234, 41.1821, 1.7103, 15000, 300
4  0:00:00.00>ADDWPT IB1234, 40.5566, 0.2655, 15000, 300

```

```

5 0:00:00.00>ADDWPT IB1234, 40, -0.2837, 15000, 300
6 0:00:00.00>ADDWPT IB1234, 39.4797, -0.457, 15000, 300
7 0:00:00.00>ADDWPT IB1234, 39.0244, -1.263, 15000, 300
8 0:00:00.00>ADDWPT IB1234, 38.5211, -2.0928, 15000, 300
9 0:00:00.00>ADDWPT IB1234, 38.3608, -2.3528, 15000, 300
10 0:00:00.00>ADDWPT IB1234, 38.1525, -3.625, 15000, 300
11 0:00:00.00>ADDWPT IB1234, 37.5684, -4.332, 15000, 300
12 0:00:00.00>ADDWPT IB1234, 37.1901, -5.6091, 15000, 300
13 0:00:00.00>ADDWPT IB1234, 36.754, -6.0569, 15000, 300
14 0:00:00.00>LNAV IB1234 ON
15 0:00:00.00>VNAV IB1234 ON

```

Código 1: Ejemplo escenario para BlueSky

4.2.5. Diagrama flujo de información

A continuación, un diagrama global del flujo de información para el modelo.



Figura 4.7: Estructura básica de la implementación

4.3. Interpretación OPL del modelo matemático

En este apartado analizaremos la conversión a *OPL* del modelo matemático descrito en la sección 3. Para ello compararemos, por bloques, la formulación teórica con el código OPL y se explicarán los distintos elementos.

4.3.1. Parámetros

En el lenguaje *OPL* existen distintos tipos de información; *Integers*, *Floats*, *Strings* y *Functions*. Al mismo tiempo, se puede organizar esta información en distintas estructuras; *Ranges*, *Arrays*, *Tuples* y *Sets* [7].

En la programación del modelo se usan prácticamente todos los tipos de información y estructuras, pero las tuplas son especialmente importantes. Uno de los elementos más tratados a lo largo de todo el modelo, son las aerovías y rutas, es decir, información sobre recorridos. Para definir un recorrido, solo hace falta saber dónde empieza y dónde termina, en nuestro caso, el nodo de origen y destino.

Por ese motivo se definen dos nuevas clases de tuplas; aerovías y rutas. Las dos clases son tuplas contenedoras de dos elementos, “strings”, definiendo el nodo de origen y el nodo de destino, como podemos ver en el código 2.

```
1 //DEFINICIÓN DE CLASES
2 tuple aerovia //Tupla que define la aerovía (orig, dest)
3 {string orig; string dest;}
4 tuple ruta //Tupla que define la ruta (orig, dest)
5 {string orig; string dest;}
```

Código 2: Definición de clases aerovía y ruta

A continuación, se importa la información de entrada recopilada en el libro Excel. Para ello, se define una variable contenedora y se indica que la procedencia se especifica en el archivo “.dat” mediante tres puntos; “...” (como se puede observar en el código 24 del Anexo A). Primero importamos las características principales del modelo en variables del tipo entero:

```
1 //CARACTERÍSTICAS DEL MODELO
2 int totalVuelos = ...; //Número total de vuelos
3 int totalIncidencias = ...; //Número total de incidencias
4 int iniInt = ...; //Primer intervalo de tiempo
5 int finInt = ...; //Último intervalo de tiempo
```

Código 3: Características del modelo importado

A continuación, importamos la información relacionada con los nodos y los almacenamos en una variable con estructura tipo “set”, es decir, un conjunto.

Mientras que la variable `MidN` se genera a partir de sus conjuntos complementarios, destacar que los conjuntos `OrigN` y `DestN` se extraen directamente de la programación, eliminando duplicados. Es decir, el conjunto `OrigN` está formado por esos nodos origen que efectivamente aparecen en la programación, del mismo modo que los nodos destino aparecen en el conjunto `DestN`. Son conjuntos de elementos únicos, no contienen duplicados. Por otro lado, la variable `OrigNtotal` corresponde a un conjunto indexado y ordenado de los distintos nodos de origen para los vuelos programados (parecido a `OrigN`, pero pudiendo contener duplicados y siguiendo un orden).

Por ejemplo; `OrigN={"MAD", "BCN"}`, `OrigNtotal={"MAD", "MAD", "MAD", "BCN"}`, sabemos que hay programados cuatro vuelos, y, para saber el nodo origen del tercero debemos ejecutar `OrigNtotal[3]`.

Más adelante veremos que nos será de utilidad y que trataremos de indexar, por orden de despegue, toda la información relacionada con los vuelos programados.

```

1 //NODOS
2 {string} N = ...; //Conjunto total de nodos
3 {string} OrigN = ...; //Conjunto de nodos origen, {OrigN} in {N}
4 {string} DestN = ...; //Conjunto de nodos destino, {DestN} in {N}
5 {string} MidN = N diff (OrigN union DestN); //Conjunto de nodos intermedios, {MidN} in {N}
6 string OrigNtotal[1..totalVuelos] = ...; //Conjunto indexado y ordenado (para cada vuelo) de
   nodos de origen

```

Código 4: Nodos del modelo importado

Seguidamente importamos la estructura temporal para el modelo, al igual que el conjunto indexado y ordenado de tiempos de despegue.

```

1 //TIEMPO
2 range T = iniInt..finInt; //Conjunto total de intervalos de tiempo
3 range Tini = (iniInt-1)..finInt; //Conjunto con inicialización
4 int takeOff[1..totalVuelos] = ...; //Conjunto indexado y ordenado (para cada vuelo) de tiempo
   de despegue

```

Código 5: Importación de la estructura temporal del modelo

A continuación, importamos la estructura de las aerovías y sus características, como distancia, tiempo de recorrido y tiempo mínimo de ocupación. Además, importamos el conjunto total de rutas, al igual que un conjunto indexado y ordenado de las distintas rutas correspondientes a los vuelos programados. Vale la pena destacar que tanto el conjunto total de aerovías como de rutas son una estructura set, o conjunto, de la clase “aerovia”.

Para indicar que se trata de un conjunto y no de un solo elemento, se indica mediante los símbolos “{” y “}”.

```

1 //AEROVÍAS
2 {aerovia} A = ...; //Conjunto total de aerovías (nodo origen, nodo destino)
3 {aerovia} Atmp = ...; //Conjunto de aerovías temporales (nodo origen, nodo destino)
4 {ruta} K = ...; //Conjunto de rutas de vuelo (nodo origen, nodo destino)
5 ruta Ktotal[1..totalVuelos] = ...; //Conjunto indexado y ordenado de rutas
6 float dis[A] = ...; //Longitud de las aerovías
7 int a[A] = ...; //Tiempo de vuelo para aerovías
8 int U[Atmp] = ...; //Tiempo mínimo de ocupación para las aerovías

```

Código 6: Aerovías del modelo importado

El siguiente paso es definir la capacidad de la red. Mediante el código 7 importamos tres conjuntos; aerovías afectadas ($A_{\text{incidencia}}$), tiempos de afectación ($T_{\text{incidencia}}$) y número máximo permitido de aeronaves ($L_{\text{incidencia}}$). Los tres conjuntos están indexados y ordenados por orden de aparición en el listado de incidencias (homólogo a la programación). De esta forma podemos recorrer todo el rango de incidencias y saber, para cada una de ellas, qué aerovías están afectadas, en qué intervalo de tiempo y cuántas aeronaves pueden usarla. A continuación, inicializamos la variable “c”⁶ con un valor por defecto (en nuestro caso 1), indicando la bidimensionalidad y sus índices. Posteriormente, ejecutamos el bloque “ini_c”, que ataca la variable solo en esos puntos donde se han detectado incidencias y establece el valor indicado L. Veremos más adelante que este procedimiento es muy útil y eficaz para la inicialización de variables, ahorramos tiempo y recursos.

```

1 //INCIDENCIAS
2 aerovia A_incidencia[1..totalIncidencias] = ...; //Conjunto indexado y ordenado (para cada
   incidencia) de aerovías (origen, destino) afectadas por incidencias
3 int T_incidencia[1..totalIncidencias] = ...; //Conjunto indexado y ordenado (para cada
   incidencia) de tiempos en que ocurre cada incidencia
4 int L_incidencia[1..totalIncidencias] = ...; //Conjunto indexado y ordenado (para cada
   incidencia) de limitaciones máximas de capacidad para cada incidencia
5 int c[i in A][j in T] = 1; //Matriz representativa del número máx de vuelos que pueden
   entrar para cada aerovía (i, j) y cada intervalo t.
6 execute ini_c {
7   for(var i=1; i<=totalIncidencias; i++) {
8     c[A_incidencia[i]][T_incidencia[i]] = L_incidencia[i];
9   }

```

Código 7: Incidencias del modelo importado

⁶El parámetro “c” corresponde con el número máximo de vuelos que pueden entrar para cada aerovía (i, j) para cada intervalo t.

Finalmente, hay que proporcionar al modelo la información sobre la programación inicial. A partir de ella, el modelo buscará una programación definitiva, lo más parecida a la inicial, optimizando costes y retrasos. También nos proporcionará la ruta exacta de cada aeronave y el tiempo aproximado del trayecto.

La programación inicial se almacena en la variable tridimensional “P”. Está indexada por ruta, nodo origen y tiempo, por ese motivo, su dimensión es considerablemente mayor que las otras variables. Para generar esta variable de tal magnitud, empezamos inicializando el “contenedor” con las dimensiones adecuadas, y con valor por defecto nulo.

A continuación, ejecutamos el bloque “ini_P”. Este bloque recorre todos los vuelos programados y, para cada uno de ellos, extrae la ruta, el nodo de origen y el intervalo de tiempo en el que despegue. Como se ha indicado anteriormente, llegados a este punto ya disponemos de toda la información referente a la programación inicial, almacenada en distintas variables indexadas y ordenadas por orden de despegue; ruta (“Ktotal”), nodo origen (“OrigNtotal”) e intervalo de despegue (“takeOff”).

Seguidamente, usamos esta información (extraída para cada vuelo) como índice para atacar la variable “P” en esos puntos donde hay un vuelo programado.

Cabe mencionar que, con este procedimiento, ahorramos tiempo y recursos. Por naturaleza, la gran mayoría de puntos de la variable “P” serán nulos, por ejemplo, si consideramos el punto $k = \langle \text{"BCN"}, \text{"MAD"} \rangle$, $\text{OrigN} = \text{"GIR"}$, $t = 2$, es decir $P[\langle \text{"BCN"}, \text{"MAD"} \rangle][\text{"GIR"}][2]$, sabemos que será nulo ya que no es lógico que para la ruta $\text{BCN} \Rightarrow \text{MAD}$, despegue un avión de GIR.

El procedimiento es análogo al de esculpir una figura a partir de un cubo, en lugar de crear la figura de la nada.

```

1 int P[K][OrigN][T]; //Matriz tridimensional que representa los vuelos planeados para despegar
  para cada ruta, nodo origen e intervalo de tiempo.
2   execute ini_P {
3     for(var i=1; i<=totalVuelos; i++) {
4       P[Ktotal[i]][OrigNtotal[i]][takeOff[i]] += 1;
5     }
  }
```

Código 8: Programación inicial del modelo importado

Por último, importamos algunas constantes. Ciertamente, podríamos personalizar estas constantes para cada nodo de origen, aerovía, aerovía temporal o intervalo de tiempo (según el caso), pero, para no aumentar la complejidad del modelo, se definen igual para todos los elementos (inicialización genérica).

```

1 //CONSTANTES
```

```

2 int Gin[i in N] = 10; //Número máximo de arovías entrantes para cada nodo
3 int Gout[i in N] = 10; //Número máximo de arovías salientes para cada nodo
4 float r = 0.025; //Coste de volar 1 km
5 float d[i in OrigN][j in T] = 1; //Coste de retrasar un vuelo para cada nodo de origen y
   intervalo de tiempo
6 float o[i in Atmp][j in T] = 5.5; //Opening fee para cada aerovía temporal a cada intervalo de
   tiempo
7 float m[i in Atmp][j in T] = 0.3; //Coste de uso para cada aerovía temporal a cada intervalo
   de tiempo

```

Código 9: Constantes del modelo importado

4.3.2. Variables de decisión

Tal y como se ha descrito en la sección 3.3, las variables de decisión se definen teóricamente de la siguiente forma:

$$\begin{aligned}
 x_{i,j}(t) &= \begin{cases} 1, & \text{si la aerovía } (i, j) \text{ se usa en intervalo } t. \\ 0, & \text{en caso contrario} \end{cases} \\
 z_{i,j}(t) &= \begin{cases} 1, & \text{si la aerovía } (i, j) \text{ se abre en intervalo } t. \\ 0, & \text{en caso contrario} \end{cases} \\
 f_{i,j}^k(t) &\in \mathbb{Z}^+
 \end{aligned}$$

Las variables $x_{i,j}(t)$ y $z_{i,j}(t)$ se transforman respectivamente en “x” y “z”, mientras que la variable $f_{i,j}^k(t)$ pasa a ser “f”. Declaramos las dos primeras como booleanas (mediante el comando “dvar boolean”) y “f” como entera positiva (mediante el comando “dvar int+”).

Para indicar la dimensionalidad y magnitud de las variables de forma genérica, escribimos entre los símbolos “[” y “]” los índices que queremos. De esta forma podemos determinar la dimensión de la variable al mismo tiempo que el índice que usaremos para explotarla.

La variable “f” corresponde a la programación definitiva de todo el espacio aéreo, el objetivo final y principal del modelo. Está indexada por ruta, aerovía e intervalo de tiempo, por ejemplo; $f[<\text{“BCN”}, \text{“MAD”}>][<\text{“BCN”}, \text{“NOVAS”}>][1]=1$ nos indica que despega un avión de la ruta “BCN”-“MAD”, hacia el nodo “NOVAS” en el instante 1.

Es importante destacar que gran parte de la arquitectura del modelo está enfocada a hacer la variable “f” fácilmente accesible. Muestra de ello es el uso de tuplas para

la definición de aerovías y rutas, como hemos visto. El hecho de que las tuplas sean estructuras inamovibles e indexadas, nos permite acceder en ellas e imponer restricciones en todo momento, más adelante veremos ejemplos.

```

1 //VARIABLES DE DECISIÓN
2 dvar boolean x[A][Tini]; //Matriz para cada aerovía e intervalo de tiempo que representa si
    está en uso (1) o no (0)
3 dvar boolean z[Atmp][Tini]; //Matriz para cada aerovía e intervalo de tiempo que representa si
    se abre (1) o no (0)
4 dvar int+ f[K][A][Tini]; //Matriz tridimensional para cada ruta, aerovía e intervalo de tiempo
    que representa los vuelos entrantes

```

Código 10: Variables de decisión del modelo importado

4.3.3. Función objetivo

Tal y como se ha descrito en la sección 3.4, la función objetivo se define teóricamente de la siguiente forma:

$$\begin{aligned}
 \min Z = & \sum_{t,k,i \in N_{orig}} d_i(t) \cdot f_{i,i}^k(t) + \sum_{t,k,i \in A} r \cdot dis_{i,j} \cdot f_{i,j}^k(t) \\
 & + \sum_{t,(i,j) \in A_s} o_{i,j}(t) \cdot z_{i,j}(t) + \sum_{t,(i,j) \in A_s} m_{i,j}(t) \cdot x_{i,j}(t)
 \end{aligned} \tag{4.1}$$

Nos interesa segregar el sumatorio, de esta forma podemos tratar individualmente cada término. Para ello, definimos cuatro expresiones independientes; FlightDelayCost, FlightDistCost, OpeningTempAirways, UsingTempAirways.

1. FlightDelayCost:

```

1 //Coste de los retrasos
2 dexpr float FlightDelayCost =
3   sum(t in T, k in K, i in OrigN, <orig, dest> in A: orig == dest && dest == i) d[i][t]
    ]*f[k][<orig, dest>][t];

```

Código 11: Coste de los retrasos

Sumatorio de todos los retrasos, ponderado por su coste correspondiente. Debemos recorrer todos los intervalos de tiempo, rutas y nodo origen. Para ello, mediante la condición “<orig, dest>in A: orig == dest && dest == i”, el modelo solo tiene en cuenta las variables “f[k][<i, i>][t]”⁷ donde $i \in \{\text{OrigN}\}$.

⁷Una aerovia con mismo origen y destino, define un retraso, con distancia nula y tiempo de trayecto unitario.

2. FlightDistCost

```

1 //Coste de vuelo
2 dexpr float FlightDistCost =
3   sum(t in T, k in K, a in A) r*dis[a]*f[k][a][t];

```

Código 12: Coste de vuelo

Sumatorio total del coste de vuelo. Se recorre todo el rango de tiempo, todas las rutas⁸ y aerovías. Se pondera por la distancia y el coste por kilómetro.

3. OpeningTempAirways

```

1 //Coste de la apertura de aerovías temporales
2 dexpr float OpeningTempAirways =
3   sum(t in T) sum(a in Atmp) o[a][t]*z[a][t];

```

Código 13: Coste Opening Fee

Sumatorio total del coste de apertura para las aerovías temporales. Se recorre todo el rango de tiempos y aerovías. Se pondera por el coste de abrir cada aerovía y por la variable booleana z , que determina si se abre o no.

4. UsingTempAirways

```

1 //Coste de mantener abiertas las aerovías temporales
2 dexpr float UsingTempAirways =
3   sum(t in T) sum(a in Atmp) m[a][t]*x[a][t];

```

Código 14: Coste Usage Fee

Sumatorio total del coste de mantenimiento para las aerovías temporales. Se recorre todo el rango de tiempos y aerovías. Se pondera por el coste de mantener abierta cada aerovía y por la variable booleana x , que determina si se está en uso o no.

Finalmente se unifican todas las expresiones y se indica el objetivo del modelo; minimizar la función objetivo.

```

1 //Coste total
2 minimize FlightDelayCost + FlightDistCost + OpeningTempAirways + UsingTempAirways;

```

Código 15: Coste total

⁸Las aerovías con mismo origen y destino tienen distancia nula y no computan.

4.3.4. Restricciones

Sin duda, la interpretación de las restricciones del modelo, es uno de los puntos con más complejidad en el proceso de programación. Aunque el lenguaje de programación *OPL* esté diseñado para tal circunstancia, la riqueza del modelo reside en sus restricciones, por lo que se requiere un buen conocimiento del lenguaje.

En este apartado sólo se explicará la interpretación codificada, la funcionalidad matemática de cada una de las restricciones está descrita en la sección 3.5.

1. Primero, inicializamos algunas variables. Se fija el valor de aquellas variables que son necesarias para algún cálculo posterior, aunque no tienen sentido real.

```

1 //RESTRICCIONES
2 subject to {
3     //Inicialización de x, z, f para t = 0
4     forall (a in A)
5         x[a][0] == 0;
6     forall (a in Atmp)
7         z[a][0] == 0;
8     forall (a in A, k in K)
9         f[k][a][0] == 0;

```

Código 16: Inicialización de x, z, f para t = 0

A continuación, la definición teórica de las restricciones de conservación del flujo, junto con su interpretación OPL:

2. Conservación del flujo en nodos origen

$$\sum_{j:(i,j) \in \mathcal{A}} f_{i,j}^k(t) - \sum_{j:(j,i) \in \mathcal{A}} f_{i,j}^k(t - a_{j,i}) = P_i^k(t) \quad \forall i \in \mathcal{N}_{orig}, t \in \mathcal{T}, k \in \mathcal{K} \quad (4.2)$$

```

1 forall (i in OrigN, t in T, k in K)
2 sum(<i1, j1> in A: i1 == i) f[k][<i1, j1>][t] - sum(<j2, i2> in A: i2 == i) f[k][<j2,
   i2>][(t-a[<j2, i2>]>0)?(t-a[<j2, i2>]):0] == P[k][i][t];

```

Código 17: Restricción de conservación del flujo en nodos origen

Mediante la condición “<i1, j1> in A: i1 == i”, forzamos que recorra solo las aerovías salientes de nodos origen. La condición [(t-a[<j2, i2>]>0) ? (t-a[<j2, i2>]) : 0]⁹ nos asegura que el índice (t - a_{j,i}) no será negativo, y en caso de que lo sea (por ejemplo t=1 y a=2), la expresión toma valor = 0. Este es un claro ejemplo del uso de la inicialización de la variable f en el código 16.

⁹Expresamos de la siguiente forma las condiciones (condition)?thenExpr : elseExpr[7]

3. Conservación del flujo en nodos intermedios

$$\sum_{j:(i,j) \in \mathcal{A}} f_{i,j}^k(t) - \sum_{j:(j,i) \in \mathcal{A}} f_{i,j}^k(t - a_{j,i}) = 0 \quad \forall i \in \mathcal{N}_{mid}, t \in \mathcal{T}, k \in \mathcal{K} \quad (4.3)$$

```

1 forall (i in MidN, t in T, k in K)
2   sum(<i1, j1> in A: i1 == i && i1 != j1) f[k][<i1, j1>][t] - sum(<j2, i2> in A: i2 == i
   && i2 != j2) f[k][<j2, i2>][(t-a[<j2, i2>])>0?(t-a[<j2, i2>]):0] == 0;
3 forall (i in MidN, t in T, k in K)
4   sum(<i1, j1> in A: i1 == i && i1 == j1) f[k][<i1, j1>][t] == 0;

```

Código 18: Restricción de conservación del flujo en nodos intermedios

Mediante la condición “<i1, j1> in A: i1 == i && i1 != j1”, forzamos que recorra solo las aerovías salientes de nodos intermedios, además de indicarle que no sean nodos retraso (origen y destino coinciden). La condición “[(t-a[<j2, i2>])>0 ? (t-a[<j2, i2>]) : 0]” nos asegura la no negatividad del índice, al igual que en el código 17.

Aplicamos dos restricciones para interpretar la ecuación 4.3. La segunda restricción cubre el caso de las aerovías retraso (origen igual a destino).

4. Conservación del flujo en nodos destino

$$\sum_{t,j:(j,i) \in \mathcal{A}} f_{j,i}^k(t) - \sum_{t,q \in \mathcal{N}_{orig}} P_i^k(t) = 0 \quad \forall i \in \mathcal{N}_{dest}, k \in \mathcal{K} \quad (4.4)$$

```

1 forall (i in DestN, <k1, k2> in K)
2   sum(<j1, i1> in A: i1 == i, t in T) f[<k1, k2>][<j1, i1>][t] - ((i == k2)? (sum(t in T)
   P[<k1, k2>][k1][t]):0) == 0;

```

Código 19: Restricción de conservación del flujo en nodos destino

Mediante la condición “((i == k2)? (sum(t in T) P[<k1, k2>][k1][t]):0)” nos aseguramos de que la restricción se aplica correctamente para cada pareja de ruta y nodo destino. Sin esta condición, el sumatorio abarcaría a todo el conjunto “DestN”, sin tener en cuenta que cada ruta tiene un solo destino.

A continuación, otras restricciones del modelo:

5. La restricción de capacidad.

$$\sum_{k \in \mathcal{K}} f_{i,j}^k(t) \leq c_{i,j}(t) \cdot x_{i,j}(t) \quad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T} \quad (4.5)$$

```

1 //Restricción de capacidad
2 forall (<i1, j1> in A, t in T)
3   sum(k in K) f[k][<i1, j1>][t] <= c[<i1, j1>][t]*x[<i1, j1>][t];

```

Código 20: Restricción de capacidad

6. La restricción del mínimo tiempo de ocupación.

$$U_{i,j}(t)[z_{i,j}(t) - (x_{i,j}(t) - x_{i,j}(t-1))] \leq \sum_{\tau \in (t-U_{i,j}, t)} x_{i,j}(\tau) \quad \forall (i,j) \in \mathcal{A}_{tmp}, t \in \mathcal{T} \quad (4.6)$$

```

1 //Restricción de capacidad
2 forall (a in Atmp, t in T)
3   U[a]*(z[a][t]-(x[a][t]-x[a][t-1])) <= sum(soi in ((t-U[a]>0)?(t-U[a]): 0) .. (t))x[a][
   soi];

```

Código 21: Restricción de tiempo de ocupación

Mediante la expresión “soi in ((t-U[a]>0)?(t-U[a]): 0) .. (t)”, delimitamos el rango temporal $\tau \in (t - U_{i,j}, t)$. Este rango empieza en $(t - U_{i,j})$ y termina en t . Para asegurar la no negatividad del primer término, lo expresamos con la condición “(t-U[a]>0)?(t-U[a]): 0”, y, para crear el rango, lo expresamos mediante dos puntos; “.. (t)”.

Destacar que, en la expresión “x[a][t-1]”, el término “[t-1]” solo será nulo o positivo. En caso de que sea nulo ($t=1$, $[t-1]=0$), la variable está preparada, ya que la hemos inicializado a 0 (ver código 16).

7. La restricción que relaciona la variable x y z. $z = 1$ cuando x cambia de 0 a 1:

$$x_{i,j}(t) - x_{i,j}(t-1) \leq z_{i,j}(t) \quad \forall (i,j) \in \mathcal{A}_{tmp}, t \in \mathcal{T} \quad (4.7)$$

```

1 //Relación entre x & z. z = 1 cuando x cambia de 0 a 1
2 forall (a in Atmp, t in T)
3   x[a][t] - x[a][t-1] <= z[a][t];

```

Código 22: Restricción relación entre x y z

8. Por último, la restricción que limita el número de aerovías entrantes y salientes de un nodo:

$$\sum_j x_{j,i}(t) \leq w_{in}(i) \quad \forall i \in \mathcal{N}_{mid}, t \in \mathcal{T} \quad (4.8)$$

$$\sum_j x_{i,j}(t) \leq w_{out}(i) \quad \forall i \in \mathcal{N}_{mid}, t \in \mathcal{T} \quad (4.9)$$

```

1 //Limitación de aerovías entrantes y salientes
2 forall (i in MidN, t in T)
3   sum(<j1, i1> in A: i1 == i)x[<j1, i1>][t] <= Gin[i];
4 forall (i in MidN, t in T)
5   sum(<i1, j1> in A: i1 == i)x[<i1, j1>][t] <= Gout[i];

```

Código 23: Limitación de aerovías entrantes y salientes

5. Resultados

Para probar la aplicación y los resultados obtenidos con ella, se han planteado dos escenarios de distinta naturaleza. En este apartado se mostrará, a modo de guía de usuario, el procedimiento seguido para llegar a estos resultados. Los dos escenarios son los siguientes:

- *ATM_SCN*: Este escenario ha servido para el desarrollo del aplicativo. Se trata de un espacio aéreo simple, diseñado para la fácil comprobación de los resultados. Consta de 100 nodos y más de 400 aerovías, o conexiones entre nodos. Aunque no se trate de un ejemplo real, al tener una estructura simplificada, permite una mayor velocidad de ejecución para el modelo.
- *ATM_SPAIN*: En este escenario se ha introducido la estructura del espacio aéreo español real, aunque simplificada, con posibles rutas, aeropuertos, etc. Se han escogido 86 nodos reales, y más de 450 aerovías. El objetivo de este escenario es poder generar unas rutas que sirvan de “input” para el simulador ATC *BlueSky*.

5.1. Escenario ATM_SCN

5.1.1. Estructura

Se ha diseñado un espacio aéreo simplificado, una matriz 10 por 10, con cuatro aeropuertos, 3 rutas directas y 2 rutas opcionales.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

Figura 5.1: Espacio aéreo escenario ATM_SCN

Como se puede observar en la figura 5.1, los nodos 10, 11, 20 y 21 forman un aeropuerto,

donde la terminal de salida es el nodo 20 (verde) y la de llegada el nodo 10 (naranja). Lo mismo sucede con los otros tres aeropuertos.

En general, para cada nodo del espacio aéreo solo están permitidos cuatro movimientos; arriba, abajo, derecha, izquierda (color blanco). Solo se permite el movimiento diagonal entre los nodos de las rutas directas (color azul). Adicionalmente, existen las rutas opcionales, pudiendo ser abiertas o no, y que también permiten el movimiento diagonal entre ellas (color ocre).

5.1.2. Planificación

Se ha diseñado una programación que intenta cubrir todas las posibilidades. Esta programación, al igual que todos los detalles de la estructura se encuentran en el archivo Excel correspondiente al modelo. En la figura 5.2, podemos ver los detalles.

COMPañÍA	TIPO_AERONAVE	ORIGEN	DESTINO	VUELO	TAKEOFF_INT	TAKEOFF
IBERIA	A-320	18	10	IB1234	1	9:01:00
VUELING	A-320	20	98	VU1222	2	9:02:00
RYANAIR	A-320	89	62	RY8634	3	9:03:00
SWISS AIRLINES	A-320	89	19	SW8945	4	9:04:00
BRITISH AIRLINES	A-320	18	10	BR5543	5	9:05:00
IBERIA	A-320	18	62	IB3352	5	9:05:00

Figura 5.2: Planificación escenario ATM_SCN

5.1.3. Ejecución

En la hoja “*RESUMEN*” del archivo Excel, podemos encontrar los botones de ejecución para el modelo, como se puede observar en la figura 5.3.

START_SIM	END_SIM	TOTAL_VUELOS	TOTAL_INCIDENCIAS	START_SIM_INT	END_SIM_INT
9:01:00	11:00:00	6	1	1	120
<div> <div>Ejecutar modelo</div> <div>Extraer información</div> <div>Crear Escenario</div> </div> <div> <div>FlightDistCost</div> <div>FlightDelayCost</div> <div>OpeningTempAirways</div> <div>UsingTempAirways</div> <div>Total</div> </div> <div> <div>1.332</div> <div>0</div> <div>0</div> <div>0</div> <div>1.332</div> </div>					

Figura 5.3: Botones de ejecución

Para la ejecución del modelo, pulsamos el botón “*Ejecutar Modelo*”, seguidamente nos aparecerá una ventana de confirmación (para evitar ejecuciones involuntarias), como en la figura 5.4.

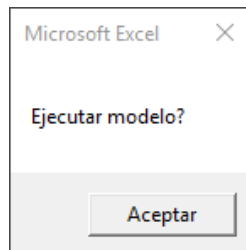


Figura 5.4: Confirmación ejecución modelo

Posteriormente, otro diálogo aparecerá, preguntando por la ruta del programa *IBM © ILOG © CPLEX © Optimization Studio*, como en la figura 5.5 ¹⁰.

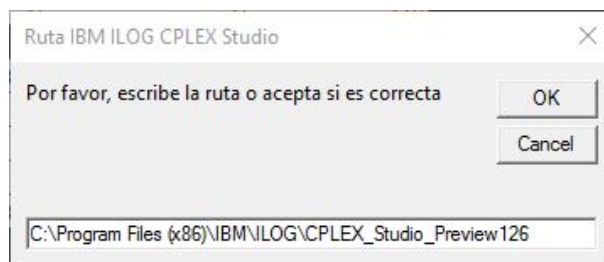


Figura 5.5: Confirmación ruta del programa

Al aceptar, se ejecutará el modelo vía script.

5.1.4. Resultados

Una vez se haya ejecutado el modelo, podemos extraer los resultados mediante el botón “*Extraer Información*”. De esta manera podremos ver el coste total de la simulación en la hoja “*RESUMEN*” (resultado del sumatorio de distintos costes, ver figura 5.7), al igual que la ruta para cada vuelo, la nueva hora de despegue y la hora prevista de aterrizaje en la hoja “*PLANIFICACION*” (ver figura 5.6).

¹⁰Por defecto aparece la ruta que se ha usado en este proyecto.

TAKEOFF	NEW TAKEOFF		0	1	2	3	4	5	6	7
9:01:00	9:01:00									
9:02:00	9:02:00									
9:03:00	9:03:00									
9:04:00	9:04:00									
9:05:00	9:05:00									
9:05:00	9:05:00									

Ruta:
18-17-16-15-14-13-12-11-10
Hora aproximada:
9:17:00

	40	41	42	43	44	45	46	47
	50	51	52	53	54	55	56	57
	60	61	62	63	64	65	66	67

Figura 5.6: Nueva programación para el escenario ATM_SCN

FlightDistCost	1.332
FlightDelayCost	0
OpeningTempAirways	0
UsingTempAirways	0
Total	1.332

Figura 5.7: Coste final del escenario ATM_SCN

5.1.5. Visualización de los resultados

Este ejemplo de prueba y desarrollo no está adaptado para el simulador ATM, pero podemos representar las rutas que seguirán los aviones, ver figura 5.8.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

Figura 5.8: Representación final para el escenario ATM_SCN

Se puede observar perfectamente como las nuevas rutas planificadas por el modelo son óptimas y efectivas.

5.2. Escenario ATM_SPAIN

5.2.1. Estructura

La idea de esta simulación es poder comprobar la compatibilidad de la aplicación en un caso real, aunque simplificado. Para ello, se ha generado un espacio aéreo, escogiendo estratégicamente puntos del espacio aéreo español, concretamente, sobre la península. Estos puntos se han obtenido de la página enaire.es [8]. En la figura 5.9 podemos observar los distintos nodos escogidos. La representación de estos mismos se ha hecho mediante la aplicación web mapcustomizer.com [9].



Figura 5.9: Nodos que conforman el espacio aéreo para el escenario ATM_SPAIN

Las conexiones entre los distintos nodos han sido escogidas convenientemente para generar un espacio aéreo homogéneo y lógico.

5.2.2. Planificación

Se ha diseñado una programación que intenta cubrir posibles vuelos reales. Esta programación, al igual que todos los detalles de la estructura se encuentran en el archivo Excel correspondiente al modelo. En la figura 5.10, podemos ver los detalles.

COMPañÍA	TIPO AERONAVE	ORIGEN	DESTINO	VUELO	TAKEOFF
IBERIA	A-320	BCN	JRZ52	IB1234	9:01:00
VUELING	A-320	BCN	ALT70	VU1222	9:02:00
RYANAIR	A-320	BRA01	IBZ45	RY8634	9:03:00
SWISS AIRLINES	A-320	ALT	MHN	SW8945	9:04:00
BRITISH AIRLINES	A-320	BRA01	MHN	BR5543	9:05:00
IBERIA	A-320	SSN	ALT70	IB3032	9:06:00
VUELING	A-320	AOG	MLG25	VU9999	9:07:00
VUELING	A-320	BCN	MLG25	VU8888	9:08:00
VUELING	A-320	BRA01	ALT70	VU7777	9:09:00
IBERIA	A-320	BRA01	AMR04	IB1919	9:10:00
RYANAIR	A-320	BRA01	IBZ45	RY6464	9:11:00
SWISS AIRLINES	A-320	BRA01	MLG25	SW7473	9:12:00
BRITISH AIRLINES	A-320	BRA01	MHN	BR9876	9:13:00
VUELING	A-320	BRA01	SSJ38	VU1112	9:14:00
IBERIA	A-320	BRA01	SLL	IB3947	9:15:00

Figura 5.10: Planificación escenario ATM_SPAIN

5.2.3. Ejecución

En la hoja “*RESUMEN*” del archivo Excel, podemos encontrar los botones de ejecución para el modelo, como se puede observar en la figura 5.11.

START_SIM	END_SIM	TOTAL_VUELOS	TOTAL_INCIDENCIAS	START_SIM_INT	END_SIM_INT
9:01:00	11:00:00	15	1	1	120
<div> <div>Ejecutar modelo</div> <div> <div>FlightDistCost</div> <div>0</div> </div> <div> <div>FlightDelayCost</div> <div>0</div> </div> <div> <div>OpeningTempAirways</div> <div>0</div> </div> <div> <div>UsingTempAirways</div> <div>0</div> </div> <div> <div>Total</div> <div>0</div> </div> </div>					
<div> <div>Extraer información</div> </div>					
<div> <div>Crear Escenario</div> </div>					

Figura 5.11: Botones de ejecución

Para la ejecución del modelo, pulsamos el botón “*Ejecutar Modelo*”, seguidamente nos aparecerá una ventana de confirmación (para evitar ejecuciones involuntarias), posteriormente, otro diálogo aparecerá, preguntando por la ruta del programa *IBM © ILOG © CPLEX © Optimization Studio*, siguiendo un procedimiento análogo a la sección 5.1.3. Finalmente, aparecerá una ventana, donde podremos ver como el programa resuelve el modelo vía script (ver figura 5.12).

```

C:\Program Files (x86)\IBM\ILOG\CPLEX_Studio_Preview126\opl\bin\x86_win32\oplrun.exe
Reduced MIP has 143935 rows, 621759 columns, and 1697308 nonzeros.
Reduced MIP has 621758 binaries, 1 generals, 0 SOSs, and 0 indicators.
Presolve time = 2.50 sec. (733.31 ticks)
Probing time = 1.37 sec. (141.55 ticks)
Clique table members: 800862.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 4 threads.
Root relaxation solution time = 13.08 sec. (3588.39 ticks)

      Nodes
      Node Left      Objective  IInf  Best Integer    Cuts/
                                     Best Bound    ItCnt    Gap
*      0      0      integral      0      208.2547      208.2548      17037      0.00%
Elapsed time = 39.08 sec. (11157.43 ticks, tree = 0.00 MB, solutions = 1)

Root node processing (before b&c):
  Real time                = 39.20 sec. (11181.77 ticks)
Parallel b&c, 4 threads:
  Real time                = 0.00 sec. (0.00 ticks)
  Sync time (average)      = 0.00 sec.
  Wait time (average)      = 0.00 sec.
  -----
Total (root+branch&cut) = 39.20 sec. (11181.77 ticks)

<<< solve

OBJECTIVE: 208.2547

```

Figura 5.12: Ejecución vía script del escenario ATM_SPAIN

5.2.4. Resultados

Al igual que en la sección 5.1.4, una vez se haya ejecutado el modelo, podemos extraer los resultados mediante el botón “*Extraer Información*”. De esta manera podremos ver el coste total de la simulación en la hoja “*RESUMEN*” (resultado del sumatorio de distintos costes, al igual que la ruta para cada vuelo, la nueva hora de despegue y la hora prevista de aterrizaje en la hoja “*PLANIFICACION*” (ver figura 5.13).

COMPañÍA	TIPO AERONAVE	ORIGEN	DESTINO	VUELO	TAKEOFF	NEW TAKEOFF	
IBERIA	A-320	BCN	JRZ52	IB1234	9:01:00	9:01:00	Ruta: BCN-NOVAS-ARLES-TATOS-VLC50-ASTRO-XEBAR-YES-BLUI-BAENA-R2247-JRZ52 Hora aproximada: 10:12:00
VUELING	A-320	BCN	ALT70	VU1222	9:02:00	9:02:00	
RYANAIR	A-320	BRA01	IBZ45	RY8634	9:03:00	9:03:00	
SWISS AIRLINES	A-320	ALT	MHN	SW8945	9:04:00	9:04:00	
BRITISH AIRLINES	A-320	BRA01	MHN	BR5543	9:05:00	9:05:00	
IBERIA	A-320	SSN	ALT70	IB3032	9:06:00	9:06:00	
VUELING	A-320	AOG	MLG25	VU9999	9:07:00	9:07:00	
VUELING	A-320	BCN	MLG25	VU1888	9:08:00	9:08:00	

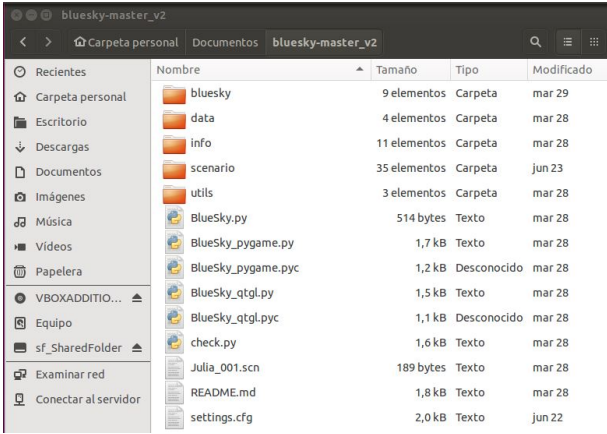
Figura 5.13: Nueva programación para el escenario ATM_SPAIN

5.2.5. Generación datos para *BlueSky*

En este escenario se usará la herramienta *BlueSky*[10].

BlueSky es una herramienta escrita íntegramente en *Python*, de código abierto y con constante evolución. Funciona bajo un Sistema Operativo Unix y existe un manual de descarga e instalación [10] para poder representar la solución obtenida. Entre sus distintas características, podemos destacar las siguientes:

- Código abierto, ultra-sencilla, y, por tanto, fácil de aprender, multi-plataforma escrita en Python 2.x (usando numpy y pygame para la visualización).
- Contiene datos sobre ayudas a la navegación, datos de rendimiento de las aeronaves y la estructura del espacio aéreo.
- Contiene simulaciones de rendimiento para distintas aeronaves, un sistema de gestión de vuelo (LNAV y VNAV en construcción), piloto automático, detección y resolución de conflictos y sistemas de garantía de la separación en el aire.
- Compatible con los datos 3.x BADA [11].
- Compatible con el gestor de tráfico TMX que se usan por NLR y la NASA LaRC.
- El tráfico se controla a través de la interacción con el usuario mediante una ventana de comandos o mediante la reproducción de archivos de escenarios (.scn) que contienen los mismos comandos con una marca de tiempo antes de la orden ("HH:MM:SS.HH>").



Nombre	Tamaño	Tipo	Modificado
bluesky	9 elementos	Carpeta	mar 29
data	4 elementos	Carpeta	mar 28
info	11 elementos	Carpeta	mar 28
scenari	35 elementos	Carpeta	jun 23
utils	3 elementos	Carpeta	mar 28
BlueSky.py	514 bytes	Texto	mar 28
BlueSky_pygame.py	1,7 kB	Texto	mar 28
BlueSky_pygame.pyc	1,2 kB	Desconocido	mar 28
BlueSky_qtgl.py	1,5 kB	Texto	mar 28
BlueSky_qtgl.pyc	1,1 kB	Desconocido	mar 28
check.py	1,6 kB	Texto	mar 28
Julia_001.scn	189 bytes	Texto	mar 28
README.md	1,8 kB	Texto	mar 28
settings.cfg	2,0 kB	Texto	jun 22

Figura 5.14: Estructura de ficheros del simulador *BlueSky*

Para generar este archivo “.scn”, el usuario debe pulsar el botón “*Generar Escenario*” de la hoja “*RESUMEN*”. Se creará un archivo .scn en la misma ruta que el Excel, con las indicaciones necesarias para simular los distintos vuelos.

5.2.6. Visualización de los resultados

Para visualizar los resultados, debemos ejecutar el programa *BlueSky*. Para ello, necesitamos disponer de un sistema operativo Unix, en nuestro caso se ha creado una máquina virtual mediante *ORACLE © VM VirtualBox* [12]. Se ha escogido el Sistema Operativo *Ubuntu 15.10* [13].

Después de descargar e instalar *BlueSky*, ejecutamos el programa por comandos, ver figura 5.15.

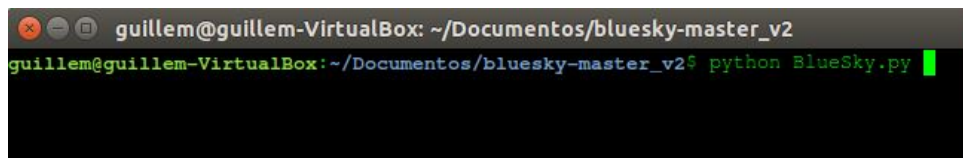


Figura 5.15: Ejecución por comandos de *BlueSky*

Después de cargar nuestro escenario, veremos la simulación completa (ver figura 5.16)



Figura 5.16: Simulación del escenario ATM_SPAIN mediante *BlueSky*

6. Presupuesto

Para la realización del proyecto se necesitarían tres personas, una con la cualificación de ingeniero informático, otra persona con la cualificación de programador y una tercera persona para la obtención de los datos y su posterior introducción en la aplicación.

Se estima un coste de 15 € por hora de trabajo del ingeniero informático, 10 € por hora de trabajo del programador y 8 € por hora de trabajo de la persona encargada de los datos.

Respecto al Software necesario para la realización del proyecto, se necesitan los siguientes programas:

- Programa base para obtener las rutas aéreas, “waypoints”, aeropuertos, etc. Por ejemplo *Microsoft Flight Simulator*.
- El módulo para el cálculo de las rutas que se ejecutará dentro del programa base para obtener las rutas; *IBM © ILOG © CPLEX © Optimization Studio*.
- Un paquete con el entorno completo para el desarrollo de la aplicación en Microsoft Visual Basic 6.0.
- Un módulo de simulación ATC para comprobar resultados, en nuestro caso se ha escogido el simulador *BlueSky*, OpenSource y codificado en *Python*.

Finalmente, respecto al Hardware necesario, se necesita, al menos, un ordenador de sobremesa con el sistema operativo Windows 10 y Unix (para usar *BlueSky*).

Los costes del proyecto están descritos en la siguiente tabla:

Concepto	Coste unitario	Cantidad	Coste [€]
Ingeniero Informático	15.00 €/h	250 h	3.750,00
Programador	10.00 €/h	170 h	1.700,00
Obtención de datos	8.00 €/h	600 h	4.800,00
Subtotal			10.250,00
Programa base para obtener las rutas	70,00 €/u	1 u	70,00
Microsoft Visual Basic 6.0	370,00 €/u	1 u	370,00
IBM ILOG CPLEX Optimization Studio	8.900 €/u	1 u	8.900,00
BlueSky ATM Simulator	0,00 €/u	1 u	0,00
Subtotal			9.340,00
Ordenador sobremesa			900,00
Consumo Electricidad			70.00
Subtotal			970,00
Total			20.560,00

Cuadro 6.1: Presupuesto

Conclusiones

En este trabajo se ha analizado un problema real y muy presente en toda la comunidad, la gestión efectiva del tráfico aéreo. Para optimizar la regulación de este, se ha formulado un modelo de optimización y se ha propuesto implementación. Para comprobar la eficacia, se han estudiado dos escenarios y simulado mediante la herramienta *BlueSky*.

Se ha realizado durante cuatro meses y los principales inconvenientes encontrados han sido;

- La restricción de acceso a determinados datos del espacio aéreo.
- La complicidad de poner todos los subsistemas a punto para el desarrollo del proyecto (máquina virtual, instalación Python 2.x, instalación de *IBM © ILOG © CPLEX © Optimization Studio*).
- La codificación del modelo desde la formulación matemática.
- El tratamiento de los resultados y la generación de archivos que los contienen.

Cabe decir que esta herramienta ha sido diseñada para espacios aéreos de complejidad baja, a causa de los recursos usados; capacidad de computación, capacidad de almacenaje, etc. Aun así, el modelo es totalmente válido para su aplicación en un entorno global.

El precio estimado para la ejecución del proyecto es de alrededor de 20.560,00 €, incluyendo el material necesario y la mano de obra.

En conclusión, se ha comprobado que el modelo planteado es efectivo y funciona en un entorno simulado. Su aplicación en un entorno real permitiría minimizar los costes mediante una organización óptima y reestructuración del espacio aéreo, así como la reprogramación de los vuelos permitiría minimizar los constantes retrasos que padecen hoy en día la mayoría de aerolíneas.

Referencias

- [1] Airbus Multi Media Support, “Global Market Forecast 2015.”, Airbus S.A.S (France) (2015).
- [2] Vossen, T. and Ball, M. , “Optimization and mediated bartering models for ground delay programs.”, Naval Research Logistics, vol 53 (2006)
- [3] O. Richetta, A.R. Odoni, “Solving optimally the static ground holding policy problem in air traffic control.”, Transport Science 24 (1993).
- [4] D. Bertsimas, S. Stock, “The air traffic flow management problem with enroute capacities.”, Oper Res 46 (1998).
- [5] Lulli, G., Odoni, A.R., “The European Air Traffic Flow Management Problem.”, Transportation Science 41 (2007).
- [6] Eswar Sivaraman, “The Multi-Commodity Network Flow Problem”, CIENA Corporation Internal Report – Core Switching Division, Cupertino, CA (2000).
- [7] “IBM ILOG CPLEX Optimization Studio OPL Language Reference Manual”, IBM, USA (2015).
- [8] <http://www.enaire.es/csee/Satellite/navegacion-aerea/es/Page/1078418725153//ENR-En-ruta.html>, Gobierno de España - Ministerio de Fomento, ENAIRE, 2014.
- [9] <https://www.mapcustomizer.com/map/DEFINITIVO1>, Patrick Kaeding, 2014.
- [10] <https://github.com/ProfHoekstra/bluesky>, Jacco M. Hoekstra & Joost Ellerbroek, Control & Simulation section, Faculty of Aerospace Engineering, TU Delft, Delft, Netherlands, 2014.
- [11] <http://www.eurocontrol.int/ddr>, Eurocontrol Demand Data Repository, 2016.
- [12] <https://www.virtualbox.org>, Oracle, 2016.
- [13] <http://www.ubuntu.com/>, Ubuntu, 2016.
- [14] S. Even and A. Itai and A. Shamir, “On the Complexity of Timetable and Multi-commodity Flow Problems”, SIAM Journal on Computing (SIAM), USA, 1976.

- [15] Max Mulder, Peng Cheng & Rui GengAir "Traffic Control", Sciyo, Croatia, 2010.

A. Código OPL

```

1 SheetConnection ExcelSheet ("DATA3.xlsx");
2 totalVuelos from SheetRead(ExcelSheet, "Resumen[TOTAL_VUELOS]");
3 totalIncidencias from SheetRead(ExcelSheet, "Resumen[TOTAL_INCIDENCIAS]");
4 iniInt      from SheetRead(ExcelSheet, "Resumen[START_SIM_INT]");
5 finInt      from SheetRead(ExcelSheet, "Resumen[END_SIM_INT]");
6 N          from SheetRead(ExcelSheet, "Nodos[CODE]");
7 A          from SheetRead(ExcelSheet, "Aerovias[[ORIGEN]:[DESTINO]]");
8 Atmp       from SheetRead(ExcelSheet, "AeroviasTmp[[ORIGEN]:[DESTINO]]");
9 K          from SheetRead(ExcelSheet, "Plan[[ORIGEN]:[DESTINO]]");
10 Ktotal     from SheetRead(ExcelSheet, "Plan[[ORIGEN]:[DESTINO]]");
11 OrigN      from SheetRead(ExcelSheet, "Plan[ORIGEN]");
12 OrigNtotal from SheetRead(ExcelSheet, "Plan[ORIGEN]");
13 DestN      from SheetRead(ExcelSheet, "Plan[DESTINO]");
14 takeOff    from SheetRead(ExcelSheet, "Plan[TAKEOFF_INT]");
15 dis        from SheetRead(ExcelSheet, "Aerovias[DISTANCIA]");
16 a          from SheetRead(ExcelSheet, "Aerovias[TIEMPO]");
17 A_incidencia from SheetRead(ExcelSheet, "Incidencias[[ORIGEN]:[DESTINO]]");
18 T_incidencia from SheetRead(ExcelSheet, "Incidencias[TIEMPO_INT]");
19 L_incidencia from SheetRead(ExcelSheet, "Incidencias[MAX_VUELOS]");
20 U          from SheetRead(ExcelSheet, "AeroviasTmp[MIN_TIEMPO]");
21 o          from SheetRead(ExcelSheet, "OpenningFee");
22 m          from SheetRead(ExcelSheet, "UsageFee");

```

Código 24: Archivo .dat

```

1
2 //PARÁMETROS
3
4 //DEFINICIÓN DE CLASES
5 tuple aerovia      //Tupla que define la aerovía (orig, dest)
6 {string orig; string dest;}
7 tuple ruta         //Tupla que define la ruta (orig, dest)
8 {string orig; string dest;}
9
10 //CARACTERÍSTICAS DEL MODELO
11 int totalVuelos = ...; //Número total de vuelos
12 int totalIncidencias = ...; //Número total de incidencias
13 int iniInt = ...; //Primer intervalo de tiempo
14 int finInt = ...; //Último intervalo de tiempo
15
16 //NODOS
17 {string} N = ...; //Conjunto total de nodos
18 {string} OrigN = ...; //Conjunto de nodos origen, {OrigN} in {N}
19 {string} DestN = ...; //Conjunto de nodos destino, {DestN} in {N}
20 {string} MidN = N diff (OrigN union DestN); //Conjunto de nodos intermedios, {MidN} in {N}
21 string OrigNtotal[1..totalVuelos] = ...; //Conjunto indexado y ordenado (para cada
    vuelo) de nodos de origen
22
23 //TIEMPO
24 range T = iniInt..finInt; //Conjunto total de intervalos de tiempo
25 range Tini = (iniInt-1)..finInt; //Conjunto con inicialización
26 int takeOff[1..totalVuelos] = ...; //Conjunto indexado y ordenado (para cada vuelo) de
    tiempo de despegue
27
28 //AEROVÍAS
29 {aerovia} A = ...; //Conjunto total de aerovías (nodo origen, nodo destino)
30 {aerovia} Atmp = ...; //Conjunto de aerovías temporales (nodo origen, nodo destino), {Atmp
    } in {A}
31 {ruta} K = ...; //Conjunto de rutas de vuelo (nodo origen, nodo destino)
32 ruta Ktotal[1..totalVuelos] = ...; //Conjunto indexado y ordenado (para cada vuelo) de
    rutas
33 float dis[A] = ...; //Longitud de las aerovías
34 int a[A] = ...; //Tiempo de vuelo para aerovías
35 int U[Atmp] = ...; //Tiempo mínimo de ocupación para las aerovías
36
37 //INCIDENCIAS
38 aerovia A_incidencia[1..totalIncidencias] = ...; //Conjunto indexado y ordenado (para cada
    incidencia) de aerovías (origen, destino) afectadas por incidencias
39 int T_incidencia[1..totalIncidencias] = ...; //Conjunto indexado y ordenado (para cada
    incidencia) de tiempos en que ocurre cada incidencia
40 int L_incidencia[1..totalIncidencias] = ...; //Conjunto indexado y ordenado (para cada
    incidencia) de limitaciones máximas de capacidad para cada incidencia
41 int c[i in A][j in T] = 1; //Matriz representativa del número máx de vuelos que pueden
    entrar para cada aerovía (i, j) y cada intervalo t.
42 execute ini_c {
43     for(var i=1; i<=totalIncidencias; i++) {
44         c[A_incidencia[i]][T_incidencia[i]] = L_incidencia[i];
45     }}

```

```

46
47 //PLANIFICACIÓN INICIAL
48 int P[K][OrigN][T]; //Matriz tridimensional que representa los vuelos planeados para
despegar para cada ruta, nodo origen e intervalo de tiempo
49 execute ini_P {
50     for(var i=1; i<=totalVuelos; i++) {
51         P[Ktotal[i]][OrigNtotal[i]][takeOff[i]] += 1;
52     }
53 //CONSTANTES
54
55 int Gin[i in N] = 10; //Número máximo de aroviás entrantes para cada nodo
56 int Gout[i in N] = 10; //Número máximo de aroviás salientes para cada nodo
57 float r = 0.025; //Coste de volar 1 km
58 float d[i in OrigN][j in T] = 1; //Coste de retrasar un vuelo para cada nodo de origen y
intervalo de tiempo
59 float o[i in Atmp][j in T] = 5.5; //Opening fee para cada aerovía temporal a cada
intervalo de tiempo
60 float m[i in Atmp][j in T] = 0.3; //Coste de uso para cada aerovía temporal a cada
intervalo de tiempo
61
62 //VARIABLES DE DECISIÓN
63 dvar boolean x[A][Tini]; //Matriz para cada aerovía e intervalo de tiempo que
representa si está en uso (1) o no (0)
64 dvar boolean z[Atmp][Tini]; //Matriz para cada aerovía e intervalo de tiempo que
representa si se abre (1) o no (0)
65 dvar int+ f[K][A][Tini]; //Matriz tridimensional para cada ruta, aerovía e intervalo de
tiempo que representa los vuelos entrantes
66
67 //FUNCIÓN OBJETIVO
68 //Coste de los retrasos
69 dexpr float FlightDelayCost =
70     sum(t in T, k in K, i in OrigN, <orig, dest> in A: orig == dest && dest == i) d[i
][t]*f[k][<orig, dest>][t];
71
72 //Coste de vuelo
73 dexpr float FlightDistCost =
74     sum(t in T, k in K, a in A) r*dis[a]*f[k][a][t];
75
76 //Coste de la apertura de aerovías temporales
77 dexpr float OpeningTempAirways =
78     sum(t in T) sum(a in Atmp) o[a][t]*z[a][t];
79
80 //Coste de mantener abiertas las aerovías temporales
81 dexpr float UsingTempAirways =
82     sum(t in T) sum(a in Atmp) m[a][t]*x[a][t];
83
84 //Coste total
85 minimize FlightDelayCost + FlightDistCost + OpeningTempAirways + UsingTempAirways;
86
87 //RESTRICCIONES
88 subject to {
89
90     //Inicialización de x, z, f para t = 0
91     forall (a in A)

```

```

92     x[a][0] == 0;
93     forall (a in Atmp)
94         z[a][0] == 0;
95     forall (a in A, k in K)
96         f[k][a][0] == 0;
97
98     //Restricciones de conservación del flujo
99     forall (i in OrigN, t in T, k in K)
100
101         sum(<i1, j1> in A: i1 == i) f[k][<i1, j1>][t] - sum(<j2, i2> in A: i2 == i) f[k][<j2, i2>][t-a[<j2, i2>]>0?(t-a[<j2, i2>]):0] == P[k][i][t];
102
103     forall (i in MidN, t in T, k in K)
104
105         sum(<i1, j1> in A: i1 == i && i1 != j1) f[k][<i1, j1>][t] - sum(<j2, i2> in A: i2 == i && i2 != j2) f[k][<j2, i2>][t-a[<j2, i2>]>0?(t-a[<j2, i2>]):0] == 0;
106
107     forall (i in MidN, t in T, k in K)
108
109         sum(<i1, j1> in A: i1 == i && i1 == j1) f[k][<i1, j1>][t] == 0;
110
111     forall (i in DestN, <k1, k2> in K)
112
113         sum(<j1, i1> in A: i1 == i, t in T) f[<k1, k2>][<j1, i1>][t] - ((i == k2)? (sum(t in T) P[<k1, k2>][k1][t]):0) == 0;
114
115
116     //Restricción de capacidad
117     forall (<i1, j1> in A, t in T)
118
119         sum(k in K) f[k][<i1, j1>][t] <= c[<i1, j1>][t]*x[<i1, j1>][t];
120
121     //Restricción del mínimo tiempo de ocupación
122     forall (a in Atmp, t in T)
123
124         U[a]*(z[a][t]-(x[a][t]-x[a][t-1])) <= sum(soi in ((t-U[a]>0)?(t-U[a]): 0) .. (t))x[a][soi];
125
126     //Relación entre x & z. z = 1 cuando x cambia de 0 a 1
127     forall (a in Atmp, t in T)
128
129         x[a][t] - x[a][t-1] <= z[a][t];
130
131     //Limitación de aerovías entrantes y salientes
132     forall (i in MidN, t in T)
133
134         sum(<j1, i1> in A: i1 == i)x[<j1, i1>][t] <= Gin[i];
135
136     forall (i in MidN, t in T)
137
138         sum(<i1, j1> in A: i1 == i)x[<i1, j1>][t] <= Gout[i];
139 }

```

Código 25: Archivo .mod

```

1 //OUTPUT
2
3 execute{
4   var fp = new IloOplOutputFile("C:\\Users\\guill\\opl\\TFG\\Coste.out");
5   var iLoop
6   var newt
7   fp.writeln("FlightDistCost:", FlightDistCost)
8   fp.writeln("FlightDelayCost:", FlightDelayCost)
9   fp.writeln("OpeningTempAirways:", OpeningTempAirways)
10  fp.writeln("UsingTempAirways:", UsingTempAirways)
11
12 };
13
14 execute{
15   var fp = new IloOplOutputFile("C:\\Users\\guill\\opl\\TFG\\sol.out");
16   var iLoop
17   var newt
18   for (k in K) for (i in A) for (t in T){
19     if ( i.orig == k.orig && i.dest != k.orig && f[k][i][t] > 0)
20     {
21       fp.write(t, ";", i.orig);
22       iLoop = i
23       newt = t
24       while (iLoop.dest != k.dest)
25       {
26         fp.write("-", iLoop.dest);
27         newt = newt + a[iLoop]
28         for (j in A)
29         {
30           if (j.orig == iLoop.dest && f[k][j][newt] > 0)
31           {
32             iLoop = j
33           }
34         }
35       }
36       newt = newt + a[iLoop]
37       fp.writeln("-", k.dest, ";", newt)
38     }
39   }
40
41   fp.close();
42 };
43
44
45 tuple exportData{
46   ruta ke;
47   aerovia ae;
48   int te;
49   int value;
50 }
51 {exportData} export = {<ke,ae,te,f[ke][ae][te]> | ke in K, ae in A, te in T};
52
53 execute{

```

```
54 var fp = new IloOplOutputFile("C:\\Users\\guill\\opl\\TFG\\sol.csv");
55 fp.writeln("RUTA;AEROVIA;TIEMPO")
56 for (k in K) for (i in A) for (t in T)
57 {
58     if (f[k][i][t]>0)
59     {
60         fp.writeln(k, ";", i, ";", t )
61     }
62 }
63 };
```

Código 26: Código encargado de la exportación de la solución

B. Código VisualBasic

```

1 Sub EjecutarModelo_Click()
2 MsgBox "Ejecutar modelo?"
3
4 Dim strPath          As String
5 Dim strCommandModel As String
6
7 Dim strPathCplex     As Variant
8 Dim oShell
9 Set oShell = CreateObject("WScript.Shell")
10
11 strPathModel = Application.ActiveWorkbook.Path
12 strCommandModel = "oplrn -p "" & strPathModel & "" "" & Left(ThisWorkbook.Name, (InStrRev(
    ThisWorkbook.Name, ".", -1, vbTextCompare) - 1)) & """"
13
14 strPathCplex = InputBox("Por favor, escribe la ruta o acepta si es correcta", "Ruta IBM ILOG
    CPLEX Studio", "C:\Program Files (x86)\IBM\ILOG\CPLEX_Studio_Preview126")
15 strPathCplex = strPathCplex & "\opl\bin\x86_win32"
16
17 oShell.CurrentDirectory = strPathCplex
18 result = oShell.Run(strCommandModel, 3, True)
19
20 Set oShell = Nothing
21
22 Worksheets("RESUMEN").Select
23
24 End Sub

```

Código 27: Macro EjecutarModelo

```

1 Sub CrearEscenario_Click()
2
3 Dim scn          As String
4 Dim FileNum      As Integer
5 Dim DataLine     As String
6 Dim tbl          As ListObject
7 Dim x            As Long
8
9 Set tbl = Sheets("PLANIFICACION").ListObjects("Plan") 'Especificamos la tabla a tratar
10 scn = ""
11 FileNum = FreeFile()
12 Open Application.ActiveWorkbook.Path & "\sol.out" For Input As #FileNum 'Abrimos el archivo .
    out
13
14 'Reseteamos la columna NEW_TAKE_OFF_INT
15 For x = 1 To tbl.Range.Rows.Count
16     tbl.DataBodyRange.Cells(x, 8) = Null
17     tbl.DataBodyRange.Cells(x, 9).ClearComments
18 Next x
19
20
21 While Not EOF(FileNum)

```

```

22 Line Input #FileNum, DataLine
23 Info = Split(DataLine, ";")
24
25 'Separamos la información de cada línea
26 TakeOff = Info(0)
27 OrigN = Split(Info(1), "-")(0)
28 DestN = Split(Info(1), "-")(UBound(Split(Info(1), "-")))
29 Arrival = Info(2)
30 ArrivalHHMM = Application.VLookup(CInt(Arrival), Sheets("TIEMPOS_INV").Range("TiemposInv"), 2, False)
31
32 'Ponemos un trigger para cuando ya hayamos encontrado la ruta correspondiente
33 written = False
34
35 'Buscamos cada línea en la tabla y determinamos el nuevo tiempo de despegue
36 For x = 1 To tbl.Range.Rows.Count
37
38     If (tbl.DataBodyRange.Cells(x, 3) = OrigN And tbl.DataBodyRange.Cells(x, 4) = DestN
39     And tbl.DataBodyRange.Cells(x, 8) = "" And Not written = True) Then
40         tbl.DataBodyRange.Cells(x, 8).Value = TakeOff
41
42         'Añadimos comentarios
43         tbl.DataBodyRange.Cells(x, 9).AddComment "Ruta:" & vbCrLf & Info(1) & vbCrLf & "
44         Hora aproximada:" & vbCrLf & CDate(ArrivalHHMM)
45         tbl.DataBodyRange.Cells(x, 9).Comment.Shape.ScaleWidth 3, msoFalse,
46         msoScaleFromTopLeft 'Formateo de los comentarios
47         tbl.DataBodyRange.Cells(x, 9).Comment.Shape.ScaleHeight 1, msoFalse,
48         msoScaleFromTopLeft
49
50         acid = tbl.DataBodyRange.Cells(x, 5)
51         atype = "B744"
52         lat = Replace(Application.VLookup(OrigN, Sheets("NODOS").Range("Nodos"), 3, False)
53         , ",", ".")
54         lon = Replace(Application.VLookup(OrigN, Sheets("NODOS").Range("Nodos"), 4, False)
55         , ",", ".")
56
57         scn = scn & " 0:00:00.00>CRE " & acid & ", " & atype & ", " & lat & ", " & lon & "
58         , 0, 0, 300" & vbCrLf
59
60         For i = LBound(Split(Info(1), "-")) To UBound(Split(Info(1), "-"))
61
62             lat = Replace(Application.VLookup(Split(Info(1), "-")(i), Sheets("NODOS").
63             Range("Nodos"), 3, False), ",", ".")
64             lon = Replace(Application.VLookup(Split(Info(1), "-")(i), Sheets("NODOS").
65             Range("Nodos"), 4, False), ",", ".")
66             scn = scn & " 0:00:00.00>ADDWPT " & acid & ", " & lat & ", " & lon & ", 15000,
67             300" & vbCrLf
68
69         Next
70
71         scn = scn & " 0:00:00.00>LNAV " & acid & " ON" & vbCrLf
72         scn = scn & " 0:00:00.00>VNAV " & acid & " ON" & vbCrLf
73
74     End For
75 End For

```



```

65         written = True
66
67         End If
68
69     Next x
70
71 Wend
72
73 lngFree = FreeFile
74 NewSCN = ThisWorkbook.Path & "\prueba.scn"
75 Open NewSCN For Output As #lngFree
76 Print #lngFree, scn
77 Close #lngFree
78
79 Worksheets("PLANIFICACION").Select
80
81 End Sub

```

Código 28: Macro CrearEscenario

```

1 Sub ExtraerInfo_Click()
2
3 Dim FileNum      As Integer
4 Dim FileNum2     As Integer
5 Dim DataLine     As String
6 Dim tbl         As ListObject
7 Dim x           As Long
8 Set tbl = Sheets("PLANIFICACION").ListObjects("Plan") 'Especificamos la tabla a tratar
9
10
11 FileNum = FreeFile()
12 Open Application.ActiveWorkbook.Path & "\sol.out" For Input As #FileNum 'Abrimos el archivo
    sol.out
13
14 FileNum2 = FreeFile()
15 Open Application.ActiveWorkbook.Path & "\Coste.out" For Input As #FileNum2 'Abrimos el archivo
    Coste.out
16
17 i = 5
18 While Not EOF(FileNum2)
19     Line Input #FileNum2, DataLine
20     Info = Split(DataLine, ":")
21     Sheets("RESUMEN").Cells(i, 5).Value = Info(1)
22     i = i + 1
23 Wend
24
25 'Reseteamos la columna NEW_TAKE_OFF_INT
26 For x = 1 To tbl.Range.Rows.Count
27     tbl.DataBodyRange.Cells(x, 8) = Null
28     tbl.DataBodyRange.Cells(x, 9).ClearComments
29 Next x
30
31
32 While Not EOF(FileNum)

```

```

33 Line Input #FileNum, DataLine
34 Info = Split(DataLine, ";")
35
36 'Separamos la información de cada linea
37 TakeOff = Info(0)
38 OrigN = Split(Info(1), "-")(0)
39 DestN = Split(Info(1), "-")(UBound(Split(Info(1), "-")))
40 Arrival = Info(2)
41 ArrivalHHMM = Application.VLookup(CInt(Arrival), Sheets("TIEMPOS_INV").Range("TiemposInv"), 2, False)
42
43 'Ponemos un trigger para cuando ya hayamos encontrado la ruta correspondiente
44 written = False
45
46
47 'Buscamos cada linea en la tabla y determinamos el nuevo tiempo de despegue
48 For x = 1 To tbl.Range.Rows.Count
49
50     If (tbl.DataBodyRange.Cells(x, 3) = OrigN And tbl.DataBodyRange.Cells(x, 4) = DestN
51     And tbl.DataBodyRange.Cells(x, 8) = "" And Not written = True) Then
52         tbl.DataBodyRange.Cells(x, 8).Value = TakeOff
53
54         'Añadimos comentarios
55         tbl.DataBodyRange.Cells(x, 9).AddComment "Ruta:" & vbCrLf & Info(1) & vbCrLf & "
56         Hora aproximada:" & vbCrLf & CDate(ArrivalHHMM)
57         tbl.DataBodyRange.Cells(x, 9).Comment.Shape.ScaleWidth 3, msoFalse,
58         msoScaleFromTopLeft 'Formateo de los comentarios
59         tbl.DataBodyRange.Cells(x, 9).Comment.Shape.ScaleHeight 1, msoFalse,
60         msoScaleFromTopLeft
61
62         written = True
63
64     End If
65
66 Next x
67
68 Wend
69
70 Worksheets("PLANIFICACION").Select
71
72 End Sub

```

Código 29: Macro ExtraerInfo