Heurística y Optimización

PRÁCTICA DE PROGRAMACIÓN LINEAL

Curso 2024/2025

Nombre	NIA	Correo Electrónico	Grupo
Javier Rosales Lozano	100495802	100495802@alumnos.uc3m.es	81
Alonso Rios Guerra	100495821	100495821@alumnos.uc3m.es	81

Fecha de entrega: 31/10/2024

Índice de contenidos

1. Introducción.

La primera práctica del curso de Heurística y Optimización propone un problema de programación lineal sobre el problema de una importante compañía aérea. El objetivo de la práctica se centra en el planteamiento, desarrollo y resolución del problema mediante las técnicas aprendidas en clase puestas en práctica en dos entornos de resolución: LibreOffice Calc y el lenguaje MathProg.

El contenido de la memoria se dividirá en diversas secciones, atendiendo a la estructura de la práctica, donde se explicará el planteamiento del problema y la resolución de éste. Finalmente, analizaremos los resultados obtenidos y daremos nuestras conclusiones sobre ellos.

Primeramente, analizaremos el primer enunciado del problema, y estableceremos un modelo que represente el problema de programación lineal presentado. Además, responderemos a la solución obtenida de este mediante el resultado hecho por una hoja de cálculo de LibreOffice Calc. Después, interpretaremos los resultados obtenidos.

Seguidamente, utilizaremos el mismo método para el segundo enunciado del problema, el cual será un modelo más avanzado. Este modelado vendrá seguido por la implementación del problema anterior en un script de MathProg, el cual nos servirá para compararlo con las soluciones obtenidas en la primera parte de la práctica, y para modelar más fácilmente el segundo problema.

Finalmente, concluiremos con la valoración de los resultados obtenidos en ambas partes y consideraremos nuestras respectivas opiniones acerca del trabajo realizado.

2. Parte 1: Modelo Básico en Calc

2.1. Modelado del problema

Una vez interpretado el enunciado, podemos concluir que tendremos una **matriz de variables de decisión**, en la que cada columna representará el número de billetes vendidos para cada avión, y cada fila representará número de billetes vendidos de cada tipo. Asignaremos esta matriz como M de dimensión (3×5); para diferenciar el tipo de billetes en la matriz, asignaremos las letras $\boldsymbol{x_i, y_i, z_i}$ (donde i representa el avión) para los billetes Estándar, Leisure+ y Business+, respectivamente.

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{x_1} & \boldsymbol{x_2} & \boldsymbol{x_3} & \boldsymbol{x_4} & \boldsymbol{x_5} \\ \boldsymbol{y_1} & \boldsymbol{y_2} & \boldsymbol{y_3} & \boldsymbol{y_4} & \boldsymbol{y_5} \\ \boldsymbol{z_1} & \boldsymbol{z_2} & \boldsymbol{z_3} & \boldsymbol{z_4} & \boldsymbol{z_5} \end{pmatrix}$$

Podemos representar el **número de billetes vendidos en cada avión** como un vector columna A (5×1); cada fila se compone de la suma de los elementos x_i, y_i, z_i de la matriz M; por otro lado, podemos representar el **total de billetes vendidos de cada tipo entre todos los aviones** como un vector columna B (3×1); cada fila se compone de la suma de los elementos de cada fila de la matriz M;

$$\boldsymbol{A} = \begin{pmatrix} \sum_{i=1}^{3} \boldsymbol{M_{i1}} \\ \sum_{i=1}^{3} \boldsymbol{M_{i2}} \\ \sum_{i=1}^{3} \boldsymbol{M_{i3}} \\ \sum_{i=1}^{3} \boldsymbol{M_{i4}} \\ \sum_{i=1}^{3} \boldsymbol{M_{i5}} \end{pmatrix}, \qquad \boldsymbol{B} = \begin{pmatrix} \sum_{i=1}^{5} \boldsymbol{x_i} \\ \sum_{i=1}^{5} \boldsymbol{y_i} \\ \sum_{i=1}^{5} \boldsymbol{z_i} \end{pmatrix}$$

Donde M_{ij} representa el elemento en la fila i y la columna $j \in [1,5]$ de la matriz M.

Una vez interpretadas las variables de decisión, debemos representar la función objetivo. Ya hemos dicho que el objetivo será maximizar los beneficios de la empresa, por lo que es obvio que la función objetivo sea de tipo maximización. También sabemos los precios de cada tipo de billete, por lo que podemos interpretar la función objetivo como la suma del número de billetes vendidos de cada tipo en cada avión por su respectivo precio. Como los tipos de billete los hemos dividido por filas, podemos sumar los elementos de cada fila entre sí y multiplicarlos por dichos precios. Entonces, la **función objetivo**, quedaría de la siguiente manera:

$$\boldsymbol{\max Z = 19 \sum_{i=1}^{5} x_i + 49 \sum_{i=1}^{5} y_i + 69 \sum_{i=1}^{5} z_i}$$

En este caso, cada sumatorio representa la suma total de billetes de cada tipo vendidos. Para entenderlo de una mejor forma, la función objetivo podría interpretarse como la suma de todos los elementos de cada fila multiplicada por su respectivo precio. Podemos representar los **precios de cada billete (en €)** como un vector fila (1×3) al cual le asignaremos la letra C:

$$\boldsymbol{C = (19 \quad 49 \quad 69)}$$

También representaremos la **capacidad permitida por cada billete (en kg)** como un vector columna D (1×3) de la siguiente manera:

$$\boldsymbol{D} = \begin{pmatrix} \mathbf{1} \\ \mathbf{20} \\ \mathbf{40} \end{pmatrix}$$

Para terminar el modelado del problema, nos queda representar las **restricciones**. Estas restricciones vienen dadas por:

- **Número de asientos de cada avión (N)**: cada avión presenta un número de asientos limitado, por lo que la suma de los elementos de cada columna de la matriz M debería ser menor o igual a los asientos totales del avión que representa dicha columna.

Podemos representar el número de asientos totales de cada avión como un vector columna (5×1) al cual denominaremos N; debemos considerar que este dato debe componerse de números enteros.

$$\boldsymbol{N} = \begin{pmatrix} \mathbf{90} \\ \mathbf{120} \\ \mathbf{200} \\ \mathbf{150} \\ \mathbf{190} \end{pmatrix}$$

- **Capacidad máxima de cada avión (P)**: podríamos interpretarlo como la multiplicación de cada elemento de una columna por el respectivo equipaje máximo dado por el tipo de billete.

Podemos representar la **capacidad máxima de cada avión (en kg)** como un vector columna (5×1) que denominaremos P, donde cada fila representa la capacidad de un avión.

$$\boldsymbol{P} = \begin{pmatrix} \mathbf{1700} \\ \mathbf{2700} \\ \mathbf{1300} \\ \mathbf{1700} \\ \mathbf{2000} \end{pmatrix}$$

- **Mínimo de billetes vendidos del tipo Leisure+ para cada avión**: el hecho de que sea para cada avión cobra importancia en el modelado; como hemos dividido los tipos de billetes por filas, todos los elementos pertenecientes a la segunda fila de la matriz deben ser individualmente mayor o igual a este parámetro.
- **El mínimo de billetes vendidos del tipo Business+**: del mismo modo que la anterior restricción, solo que considerando los elementos de la tercera fila de la matriz M.
- **Porcentaje de billetes Estándar vendidos respecto al total:** finalmente, debido a que la compañía es de bajo coste, el número de billetes Estándar vendidos debe conformar mínimo el 60% del total de billetes vendidos entre todos los aviones. Esto sería fácil de representar considerando la suma de los elementos de la primera fila como el total de los billetes Estándar vendidos y la suma de todos los elementos de la matriz como el número total de billetes vendidos.

Una vez hemos establecido los vectores y las matrices, problema de programación lineal que modela el problema especificado se representaría de la siguiente manera:

$$\max Z_1 = CB$$
$$sujeto\ a:$$
$$a_i \leq n_i$$
$$M^t D \leq P$$
$$y_i \geq 20$$
$$z_i \geq 10$$
$$B_1 \geq 0,6 \cdot (1 \quad 1 \quad 1)B$$

Donde recordemos que:

- C es el vector fila de los costes de cada billete
- B es el vector columna de la suma de los billetes de cada tipo entre todos los aviones (B_i será la suma de los billetes vendidos del tipo i)
- $a_i \in A$ es la suma de billetes vendidos en el avión i
- $n_i \in N$ es el número máximo de asientos en el avión i
- M es la matriz de variables de decisión, que especifica el número de billetes de cada tipo en cada avión
- D es el vector fila de la capacidad máxima permitida para cada tipo de billete
- $y_i \in M$ es el número de billetes Leisure+ vendidos en el avión i
- $z_i \in M$ es el número de billetes Business+ vendidos en el avión i

2.2. Implementación del modelo en LibreOffice Calc

La resolución de éste modelo se hace en una hoja de cálculo de LibreOffice, y la resolución del problema mediante la función **Solver** (usando el algoritmo lineal de LibreOffice).

En la hoja de cálculo podemos encontrar diferentes matrices y vectores, representando los datos ya especificados en el apartado anterior. Podemos diferenciar también los datos mediante el color de su celda, **estableciendo en rojo los datos constantes del problema y en amarillo las variables que muestran la solución óptima del problema**.

Como decisiones de diseño podemos declarar la creación de una matriz de costes en vez de un vector de costes, ya que supondría una facilidad a la hora de calcular la función objetivo; por otro lado, hemos asignado para cada tipo de billete una letra de forma que podamos distinguir los tipos de billetes en cada parte del proyecto; de igual manera, el modelado en la hoja de cálculo se ha diseñado de tal manera que se puedan modificar los datos otorgados del problema, pudiendo cambiar el problema de programación lineal. De esta forma cualquier restricción nueva que queramos añadir, o cualquier dato que queramos modificar, es posible. El resultado del problema modelado en la hoja de cálculo de LibreOffice es el siguiente:

$$M = \begin{pmatrix} 38 & 40 & 160 & 79 & 133 \\ 21 & 27 & 23 & 61 & 21 \\ 31 & 53 & 17 & 10 & 36 \end{pmatrix}, \qquad Z_1 = 26190\ €$$

Según lo observado en la hoja de cálculo, podemos establecer que la solución es correcta, ya que **cumple todas las restricciones impuestas y todas las variables son enteras y positivas**. Por otro lado, interpretando las restricciones individualmente, podemos observar que la solución que nos otorga el Solver de LibreOffice es una solución donde, para cada avión, se venden todos los asientos que dispone; la capacidad de cada avión no es superada nunca; la restricción del número mínimo de billetes vendidos Leisure+ y Business+ en cada avión se cumple; y el total de billetes Estándar vendidos representa exactamente el 60% del total de billetes vendidos entre todos los aviones.

3. Parte 2: Modelo Avanzado en GLPK

3.1. Modelado del problema

Para el modelado del segundo problema, podemos basarnos en la tabla de slots y pistas de aterrizaje que nos enseñan en la memoria para la creación de una matriz de variables objetivo. En este caso, interpretaremos una **matriz tridimensional**, donde cada dimensión se centrará en un parámetro de las variables objetivo: la pista de aterrizaje, el slot de tiempo, y el avión asignado. Las variables objetivo no serán enteras, sino que serán binarias, y establecerán si el slot de dicha pista de aterrizaje estará asignado a ese avión. Para ello, diseñaremos una **variable binaria para determinar si el avión $\boldsymbol{i}$ tiene asignado para la pista $\boldsymbol{j}$ el slot $\boldsymbol{k}$**:

$$\boldsymbol{asignar\_avion\_slot\_pista}[\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}] = \{\mathbf{0},\mathbf{1}\}$$

La matriz de variables objetivo será una matriz tridimensional en el que cada elemento a_{ijk} será una variable binaria.

La función objetivo de este problema se centra en la minimización de los costes en combustible producidos por los aviones que tienen que esperar a que ese slot que tienen asignado esté disponible. Es por ello por lo que tendremos un **vector columna de costes** en el que cada avión precisará de un determinado coste en función de los minutos de espera. El vector de costes la asignaremos como C:

$$\boldsymbol{C} = \begin{pmatrix} \mathbf{100} \\ \mathbf{200} \\ \mathbf{150} \\ \mathbf{250} \\ \mathbf{200} \end{pmatrix}$$

Finalmente, las restricciones las basaremos en función de las variables objetivo binarias, ya que simplemente mediremos si un avión i tiene establecido para la pista de aterrizaje j el slot k. Estas restricciones establecen lo siguiente:

- **Cada avión tiene un slot de aterrizaje**: no se pueden reservar más de un slot de aterrizaje para un único avión; de igual forma, todos los aviones deben de tener un slot asignado (no puede haber ningún avión sin un slot asignado). Es por ello que esta restricción será lineal.
- **Cada slot de aterrizaje debe estar asignado a un único avión**: ningún slot debe coincidir como reserva de dos aviones distintos.
- **Disponibilidad del slot reservado**: hemos visto que la tabla de slots del enunciado no dispone de todos los slots de tiempo para todas las pistas de aterrizaje, por lo que deberemos comprobar para la pista de aterrizaje j si slot k está libre o no se encuentra disponible.

Para esta tercera restricción representaremos con una matriz binaria D un conjunto de filas como pistas de aterrizaje y columnas como slots de tiempo **para determinar si dicho slot está disponible o no**:

$$\boldsymbol{D} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

Deberemos representar de alguna manera los tiempos de aterrizaje de cada slot. La matriz T representará **el tiempo en minutos que transcurre desde la hora inicial (9:00) hasta que se inicia el slot**, para cada pista de aterrizaje.

$$\boldsymbol{T} = \begin{pmatrix} \mathbf{0} & \mathbf{15} & \mathbf{30} & \mathbf{45} & \mathbf{60} & \mathbf{75} \\ \mathbf{0} & \mathbf{15} & \mathbf{30} & \mathbf{45} & \mathbf{60} & \mathbf{75} \\ \mathbf{0} & \mathbf{15} & \mathbf{30} & \mathbf{45} & \mathbf{60} & \mathbf{75} \\ \mathbf{0} & \mathbf{15} & \mathbf{30} & \mathbf{45} & \mathbf{60} & \mathbf{75} \end{pmatrix}$$

- **Inicio de slot coincidente con la llegada programada del avión**: los slots de tiempo se miden por intervalos. El inicio de este intervalo debe comenzar posteriormente de la llegada programada para el avión al cual se le ha reservado ese slot.
- **Finalización de slot coincidente con llegada límite del avión**: de igual manera que la restricción anterior, se acota el número de slots posibles para cada avión dependiendo de si el final del intervalo de tiempo asignado para ese slot es anterior o posterior a la hora límite de llegada de un avión.

Para estas dos restricciones estableceremos también **dos vectores columna que determinarán el tiempo de llegada programada de los aviones y el tiempo límite de estos**, respectivamente. Estos vectores los llamaremos P y L. Debido a que no podemos representar intervalos de tiempo, debemos declarar estos datos de una manera que podamos realizar cálculos con ellos. Es por eso por lo que cada fila de los vectores representará **el tiempo en minutos que ha transcurrido desde la hora a la que empiezan los slots**, es decir, a las 9:00:

$$\boldsymbol{P} = \begin{pmatrix} \mathbf{10} \\ \mathbf{-5} \\ \mathbf{40} \\ \mathbf{55} \\ \mathbf{70} \end{pmatrix}, \qquad \boldsymbol{L} = \begin{pmatrix} \mathbf{75} \\ \mathbf{30} \\ \mathbf{60} \\ \mathbf{75} \\ \mathbf{90} \end{pmatrix}$$

- **Slots consecutivos para la misma pista**: si ya de por sí el problema tiene varias restricciones, el enunciado le añade la condición para cada pista de aterrizaje de no juntar dos slots de tiempo asignados para distintos aviones consecutivamente.

Para controlar esto, podemos representar una **matriz que represente los pares de slots consecutivos para todas las pistas**, lo que resultará como seis posibles combinaciones de consecutivos para los seis slots de aterrizaje existentes. Esta matriz S será una matriz de valores binarios:

$$\boldsymbol{S} = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

Una vez establecido todo, el modelado del segundo problema sería:

$$\min Z = \boldsymbol{asignar\_avion\_slot\_pista}[i,j,k] \cdot (\boldsymbol{T}[j,k] - \boldsymbol{P}[i]) * \boldsymbol{C}[i]$$

$$\boldsymbol{sujeto\ a}:$$

$$\sum_{j=1}^{4}\sum_{k=1}^{6} \boldsymbol{asignar\_avion\_slot\_pista}[i,j,k] = 1\,, \qquad \forall i$$

$$\sum_{i=1}^{5} \boldsymbol{asignar\_avion\_pista}[i,j,k] \geq 1, \ \ \forall j,k$$

$$\boldsymbol{asignar\_avion\_pista}[i,j,k] - \boldsymbol{D_{jk}} \leq 0, \ \ \forall i,j,k$$

$$\boldsymbol{asignar\_avion\_pista}[i,j,k] \cdot (\boldsymbol{T_{jk}} - \boldsymbol{P_i}) \geq 0, \ \ \forall i,j,k$$

$$\boldsymbol{asignar\_avion\_pista}[i,j,k] \cdot (\boldsymbol{T_{jk}} - \boldsymbol{L_i}) \leq 0, \ \ \forall i,j,k$$

$$\sum_{i=1}^{5}\sum_{k=1}^{6} \boldsymbol{asignar\_avion\_slot\_pista}[i,j,k] \cdot \boldsymbol{S_{hk}} \leq 1\,, \qquad \forall j,h$$

$$\boldsymbol{asignar\_avion\_pista}[i,j,k] \in \{0,1\}, \qquad \forall i,j,k$$

3.2. Implementación de la Parte 1 en MathProg

La implementación del primer problema de la práctica es similar al modelo implementado en la hoja de cálculo de LibreOffice, con la única diferencia de que dividimos el modelado en tres módulos.

En el módulo `.dat` establecemos las variables constantes del problema, como el precio de cada billete, la capacidad de cada avión, el número de asientos, el peso permitido para cada billete, el número mínimo de billetes Leisure+ y Business+ vendidos para cada avión, y el porcentaje mínimo de billetes estándar vendidos en total. Todos estos datos son modificables, y su cambio permitiría el modelado de diferentes problemas de programación lineal distintos.

En el módulo `.mod` declaramos todo lo necesario para la resolución del problema: la matriz de las variables de decisión, la función objetivo y las restricciones del problema. Este módulo es intocable, ya que representa el problema especificado en el enunciado.

Finalmente, si ejecutamos el comando `glpsol` obtenemos un fichero de texto `.txt` donde podremos observar los resultados obtenidos. En el fichero primeramente vemos que **el resultado de la función objetivo es el mismo que al ejecutar el Solver de LibreOffice**, lo cual ya nos da indicios de que el resultado de la parte 1 es correcto. Se muestra a su vez una tabla donde vemos que cada restricción del problema se cumple. Finalmente, podemos observar una última tabla, donde podemos observar el valor de todas las variables de decisión:

$$\boldsymbol{M} = \begin{pmatrix} \mathbf{19} & \mathbf{38} & \mathbf{160} & \mathbf{100} & \mathbf{133} \\ \mathbf{58} & \mathbf{31} & \mathbf{23} & \mathbf{20} & \mathbf{21} \\ \mathbf{13} & \mathbf{51} & \mathbf{17} & \mathbf{30} & \mathbf{36} \end{pmatrix}, \qquad \boldsymbol{Z_1} = \mathbf{26190}\ €$$

Sin embargo, podemos observar que estos valores no coinciden con los marcados en la hoja de cálculo anterior; esto podría dar indicios de que **no hay una única combinación de valores fijos para las variables de decisión que otorguen la solución óptima del problema**.

3.3. Implementación de la Parte 2 en MathProg

El segundo problema se modela de forma similar al primero; tendremos el mismo formato de cada módulo (`.dat, .mod, .txt`) donde estableceremos los datos, variables, función objetivo y

restricciones explicadas en el primer apartado de esta sección. Una vez ejecutamos el comando de ejecución, obtenemos que la función objetivo de minimización toma el siguiente valor:

$$\min Z_2 = 4500\ €$$

Debido a que las variables de decisión se representan mediante una matriz "tridimensional" con valores binarios, estableceremos los resultados en la siguiente tabla, donde las filas representan las pistas de aterrizaje y las columnas los slots de tiempo posible. **El color de la celda representa el estado de cada slot de tiempo en cada pista de aterrizaje**: las celdas negras son los slots que no están permitidos para asignar, según el enunciado del problema; las celdas verdes son los slots que han quedado libres; y las celdas azules son los slots asignados a cada avión en la resolución del problema.

Pista/Slot	9:00-9:14	9:15-9:29	9:30-9:44	9:45-9:59	10:00-10:14	10:15-10:30
1					AV4	
2				AV3		
3		AV1				
4	AV2					AV5

Solamente con la interpretación de la tabla, podemos confirmar que **todas las restricciones establecidas en el problema se cumplen**: cada slot tiene asignado como máximo un avión, cada avión tiene asignado un slot que anteriormente estaba disponible, los tiempos de cada slot se cumplen atendiendo al tiempo de llegada y tiempo límite de cada uno de los aviones, y ningún slot reservado por un avión es consecutivo a un slot reservado por otro avión.

3.4. Implementación de ambos modelos (Parte 1 y Parte 2) en MathProg

Finalmente, debemos juntar ambos modelados del problema para maximizar el beneficio de la aerolínea. Debemos tener en cuenta que el problema de programación lineal de la primera parte considera una función de maximización, mientras que el segundo considera una de minimización. Debido a que se trata de una empresa que busca siempre maximizar su beneficio, la función objetivo comparará la resta entre beneficios por venta de billetes y costes de cada avión. Por lo tanto, la función objetivo de este problema será una función objetivo de maximización.

$$\max Z = Z_1 - Z_2$$

Con respecto a las restricciones y a los datos no debemos cambiar nada; por la forma en la que se ha modelado el proyecto, simplemente con añadir a cualquiera de los dos modelos las restricciones, variables objetivo y constantes bastaría para aplicar el Solver GLPK. Lo único a tener en cuenta en el archivo `.mod` sería la modificación de la función objetivo según hemos establecido arriba. De esta forma, el resultado de la función objetivo para el problema global sería:

$$\max Z = 26190 - 4500 = 21690\ €$$

Una vez ejecutamos el comando, el fichero de salida nos muestra los resultados que ya sabíamos. Ninguna variable de decisión tanto del primer como del segundo problema ha variado en su valor, ya

que estamos hablando de dos problemas que no son complementarios, y realizan cálculos independientes con respecto al otro. Es por ello que la variación del resultado se muestra solamente en la función objetivo.

Podemos observar que, como hemos establecido en la declaración de la función objetivo global, el valor de la función objetivo es la resta entre los valores obtenidos en las respectivas funciones objetivo de cada problema por separado. Esto es debido a que, como acabamos de explicar, las variables de decisión no se entremezclan, y hace posible que ambos problemas se consideren de forma independiente.

4. Conclusiones y Análisis de Resultados

4.1. Preguntas de la práctica

En el enunciado de la práctica se propone como ejercicio final a realizar una serie de preguntas que nos ayudarían a entender más aún el problema, relacionadas con la interpretación de posibles soluciones que obtendríamos al modificar el problema. Gracias a que el modelado realizado tanto en MathProg como en LibreOffice Calc, podemos modificar estos para obtener la respuesta a este apartado:

1. **¿Qué restricciones están limitando la solución del problema?**

Las restricciones que limitan nuestra solución son las siguientes:

- *Slot aterrizaje igual o posterior al aterrizaje*
- *Un solo slot por avión*
- *El mínimo de billetes estándar*
- *La capacidad máxima de equipaje por avión*
- *El máximo de asientos por avión*

Hemos comprobado volviendo a ejecutar el programa y eliminando únicamente dichas restricciones, observando que varía el resultado de la función objetivo.

2. **¿Cuántas variables y restricciones se han definido?**

Podemos comprobar el número de restricciones en el contenido del archivo de salida que especificamos en el comando para que albergue la solución al problema. Podemos observar que el modelo general incluye 431 restricciones y 135 variables de decisión; se trata de números muy grandes ya que hay prevalecen en número las restricciones de variables binarias que las enteras.

3. **¿Qué ocurriría si variamos las restricciones y los datos del problema?**

Atendiendo a aquellas restricciones que si las eliminamos no varían el resultado de la función objetivo (como hemos indicado en la primera pregunta), podemos empezar a endurecer dichas restricciones para hacer más restrictivo el problema. Por ejemplo, si añadimos que un cierto porcentaje menor al 60% del total debe significar el total de billetes Leisure+ y/o Business+ vendidos, entonces el problema se hace más restrictivo, y por tanto los beneficios podrían disminuir. De igual manera, si asumimos que la compañía es de alto standing y el número de billetes estándar vendidos debe conformar menos del 60% del total, el problema sería menos restrictivo y, por tanto, el valor de la función objetivo aumentaría. Algo parecido podría ocurrir con las demás restricciones que hemos mencionado (si no medimos la capacidad ni el número de asientos de los aviones, si podemos reservar slots de aterrizaje antes de la hora programada para el aterrizaje del avión, etc.).

Por otro lado, con respecto a la adición de datos y variables de decisión (como, por ejemplo, añadir más aviones, más pistas de aterrizaje y/o más variedad de billetes)

También deberíamos considerar el resultado de añadir nuevas restricciones que no se mencionan en el problema. Por ejemplo, podríamos cambiar el precio de cada billete dependiendo de para qué avión se vendan (para el primer avión, el billete estándar se vende a 19€; para el segundo, a 15€; para

el tercero, 30€); y así con todas las restricciones. Esto, dependiendo del valor del dato modificado, beneficiaría o restringiría al problema, y aumentaría o disminuiría el valor de la función objetivo.

4. ¿Qué pasaría si ocurriera un retraso en los vuelos?

Se crearía un parámetro denominado `tiempo_retraso` por cada avión, tomando como valor por defecto 0 (no hay retraso) o m (donde m son minutos de retraso). Luego, se sumaría este tiempo de retraso al tiempo de llegada y al tiempo límite.

En el caso de que haya 20 minutos de retraso en el vuelo del primer avión, sí se encuentra una solución. De hecho, esta nueva solución aumenta el beneficio en comparación con la solución del problema de la práctica. La nueva asignación de vuelos sería:

Pista/Slot	9:00-9:14	9:15-9:29	9:30-9:44	9:45-9:59	10:00-10:14	10:15-10:30
1					AV4	
2				AV3		
3			AV1			
4	AV2					AV5

4.2. Interpretación de los resultados y decisiones de diseño generales

Podemos asumir con los resultados obtenidos que la empresa obtendrá unos beneficios considerables a pesar del coste generado por cada avión. Como observaciones podemos mencionar la **posibilidad de distintas combinaciones de valores para la solución óptima del primer problema**, lo cual es lógico pensar debido a que el precio de cada tipo de billete es el mismo independientemente del avión al que está asignado. Probablemente, si el precio del billete dependiese también del avión al que pertenece, estableceríamos más restricciones y, por tanto, se darían menos combinaciones para llegar a la solución óptima. Por otro lado, **en el desarrollo de este proyecto no hay forma de comprobar posibles soluciones que otorguen el resultado óptimo para el segundo problema**. Sería posible si se implementase éste en un modelo de LibreOffice Calc. Sin embargo, el modelado de éste cambiaría, ya que la matriz de variables objetivo sería tridimensional, lo cual no es precisamente sencillo de representar.

En cuanto a las decisiones de diseño, hemos optado por definir las restricciones como $expr$ ($\geq$ o $\leq$) n, siendo n un número y $expr$ la expresión de la restricción comúnmente como una resta. Esta decisión ha sido tomada tras investigar un poco el funcionamiento del Solver. Con esto entregamos al Solver una tarea algo más sencilla ya que no tiene que traducir las restricciones de $expr \geq expr$ a $expr \geq n$, a parte nos ayuda a hacer más compatible y escalable nuestro código.

Finalmente, diferenciamos las dos herramientas utilizadas para el desarrollo de este proyecto. Por un lado, **LibreOffice Calc otorga un Solver sorprendentemente amplio, capaz de otorgar soluciones bastante precisas para problemas de programación lineal "sencillos"**. Con "sencillos" no nos referimos, obviamente, a problemas de pocas restricciones, o pocas variables objetivo. Sin embargo, **en el caso de que se presente un problema más complejo**, como es el caso de la segunda parte de la práctica donde podemos interpretar una matriz de variables objetivo de tres dimensiones, **es mucho más práctico usar el Solver GLPK**.

Desafortunadamente, el uso de MathProg puede llegar a ser muy tedioso, sobre todo si es la primera vez que uno lo usa, y probablemente resulte una complicación más para problemas de menor complejidad, los cuales se pueden interpretar perfectamente y de forma más visual en una hoja de cálculo.

4.3. Conclusiones de la práctica

El desarrollo de la práctica nos ha resultado muy útil. Hemos aprendido el manejo de herramientas como LibreOffice Calc y MathProg (GLPK) para la resolución de problemas de programación lineal. Además, consideramos que, a pesar de estar resolviendo un problema relacionado con la maximización del beneficio de una compañía aérea, nos sentimos capaces de resolver una amplia variedad de problemas relacionados del mismo modo.

En cuanto al desarrollo de la práctica, nos hemos apoyado en un repositorio GitHub para ir subiendo las distintas versiones de esta, donde también hemos mostrado cada modelado por separado y sus respectivas soluciones (https://github.com/AlonsoR04/p1_hyo.git).

Por último, consideramos que hemos realizado un buen trabajo y estamos satisfechos con el mismo.