Heurística y Optimización

Práctica 1. Programación lineal.

Desarrollado por:

Marcelino Tena Blanco; NIA: 100383266

Alejandro Díaz García; NIA: 100383181

Índice:

Introducción general de la práctica

En la siguiente práctica se va a desarrollar un problema propuesto de programación lineal utilizando dos herramientas, cada una de ellas con diferentes opciones. La práctica por lo tanto se va a dividir en dos partes diferenciadas, una de ellas se hará uso de la herramienta de hoja de cálculo (*Calc*) para la resolución de la tarea de programación lineal. Por otro lado, en la segunda parte se va a hacer uso del lenguaje de modelado *GLPK* (GNU Linear Programming Kit) para resolver un modelo más avanzado que el de la primera parte.
El problema propuesto se basa en un problema de una compañía aérea la cual busca maximizar sus beneficios, para lo que desea obtener la configuración óptima de billetes que debe vender para ello, y por otro lado un problema más avanzado de un aeropuerto en el cual se deben asignar pistas a cada uno de los aviones del aeropuerto.

Las dos primeras partes del documento estarán estructuradas de la siguiente forma:

- Introducción: se basa en la explicación del problema y de los datos que tenemos.
    - En la parte 1 exponemos los datos y explicamos el problema a solucionar de maximizar los beneficios a partir de la venta de billetes.
    - En la parte 2 exponemos los datos y explicamos el problema a solucionar para asignar aviones en puertos y en distintos slots.
- Modelación: se basa en explicar el modelo generado por cada parte
    - En la parte 1 explicamos el modelo mediante programación lineal entera para resolver el problema de maximizar los beneficios a partir de la venta de billetes.
    - En la parte 2 explicamos el modelo mediante programación lineal entera para resolver el problema de asignar aviones en puertos y en distintos slots. En esta parte se añade un párrafo donde se explica cómo hemos unido los modelos de la parte 1 y de la parte 2 para formar un modelo final donde se busca la maximización de beneficio a partir de los ingresos de billetes menos los gastos de tiempo de vuelo de más.
- Implementación: se basa en explicar cómo hemos implementado cada uno de los modelos en sus programas:
    - En la parte 1 se explica cómo se ha implementado en *Calc*, además de agregar una tabla con los valores que deben agregarse al solver/solucionador.
    - En la parte 2 se explica cómo se ha implementado la parte modelada final en lenguaje Mathprog utilizando GLPK.

La parte 3 es la de análisis de resultados, donde:

- Tanto para el problema 1 como para el problema final se han explicado cada una de las soluciones óptimas y se ha comprobado si cumplen con las restricciones.
- Para el problema final se ha agregado la parte de explicar un modelado final con un parámetro nuevo con retrasos de tiempo por avión.
- Al final de esta parte se compara las herramientas utilizadas (*Calc* y *GLPK*)

La última parte es la de conclusiones, donde hablamos sobre nuestras observaciones sobre la práctica.

Parte 1. Modelo básico en Calc

Introducción

Una compañía aérea quiere conocer cómo maximizar los beneficios mediante la venta de billetes. Para ello, nos da la siguiente información:

Tarifa	Peso de equipaje máximo	Precio
Ticket Estándar	1 Kg	19€
Ticket Leisure plus	20 Kg	49€
Ticket Business plus	40 Kg	69€

En esta tabla podemos ver las diferentes tarifas, con sus distintos precios y pesos permitidos del equipaje.

Avión	Asientos	Capacidad
AV1	90	1700 Kg
AV2	120	2700 Kg
AV3	200	1300 Kg
AV4	150	1700 Kg
AV5	190	2000 Kg

En esta tabla podemos ver el total de asientos que tiene cada avión y la capacidad de carga máxima de equipaje que puede llevar.

Una vez teniendo los diferentes datos, nos da las siguientes restricciones:

- Un avión **no puede superar** su capacidad máxima de asientos.
- El peso del equipaje **nunca puede superar** a la carga máxima del avión.
- Cada avión **mínimo** debe vender 20 billetes Leisure plus
- Cada avión **mínimo** debe vender 10 billetes de business plus.
- El total de billetes estándar **mínimo** debe ser 60% del total.

Modelización

En el enunciado podemos sacar en claro que el objetivo del problema se trata de maximizar los ingresos de la compañía vendiendo la configuración óptima de billetes en función de los aviones de los que se dispone. Por lo tanto, la función objetivo será de maximizar el número de billetes para obtener el máximo de ingresos. Disponemos de tres tipos de billetes a vender para cada avión. Cada billete tiene un precio diferente y cada avión dispone de unas características diferentes en cuanto a número de plazas y capacidad de carga máxima. Podemos crear en base a esto las variables de decisión que serán el número de billetes de cada tipo que debemos vender. Los billetes del tipo *Estándar* tienen un coste de 19€, los

billetes del tipo *Leisure Plus* tienen un coste de 49€ y por último los billetes del tipo Business Plus tienen un coste de 69€.

Los conjuntos seleccionados para este problema son:
AVIONES = {a1, a2, a3, a4, a5}.
BILLETES = {estandar, leisure_plus, business_plus}
BILLETES_MIN = {leisure_plus, business_plus}, donde $BILLETES\_MIN \subset BILLETES$

Para tener un problema general, debemos tener en cuenta los siguientes parámetros.

Parámetros	Significado
$asientos_i$	Número de asientos máximo que puede tener el avión i, donde i pertenece al conjunto AVIONES.
$carga\_aviones_i$	Capacidad máxima de carga de equipaje del avión i, donde i pertenece al conjunto AVIONES.
$precio\_billetes_i$	Precio de cada uno de los billetes, donde i pertenece al conjunto BILLETES.
$carga\_billetes_i$	Número máximo de carga permitida de equipaje por cada tipo de billete (Kg), donde i pertenece al conjunto BILLETES.
$min\_billetes_i$	Es el número mínimo de billetes, donde **i** pertenece a BILLETES_MIN
porcentaje_estandar	Porcentaje mínimo de billetes estándar que debe vender la aerolínea (Porcentaje/100).

El número de billetes que debemos vender para cada avión es lo que nos interesa al final, pero como la función objetivo trata de maximizar los ingresos vamos a considerar el total de billetes vendidos de cada tipo para la función objetivo. Para modelizar de la forma más eficiente hemos decidido crear la siguiente matriz de trazabilidad.

	AV1	AV2	AV3	AV4	AV5
Estándar	$units_{11}$	$units_{12}$	$units_{13}$	$units_{14}$	$units_{15}$
Leisure	$units_{21}$	$units_{22}$	$units_{23}$	$units_{24}$	$units_{25}$
Business	$units_{31}$	$units_{32}$	$units_{33}$	$units_{34}$	$units_{35}$

Como se puede apreciar cada $units_{ij}$ se corresponde al número de billetes de cada tipo a vender en cada avión, donde **i** se corresponde al tipo de billete de avión (filas) y **j** se corresponde al número de avión (columnas).

Para obtener el ingreso total, debemos multiplicar el precio del billete por nuestra variable $units_{ij}$, es decir, esa es nuestra función objetivo:

$$max\ beneficio\ = \sum_{i=1}^{BILLETES} \sum_{j=1}^{AVIONES} precio\_billetes_i\ units_{ij}$$

Tras obtener la función objetivo y la matriz de trazabilidad correspondiente ahora vamos a definir las restricciones propuestas en el problema:

- El total de billetes estándar **mínimo** debe ser $porcentaje\_estandar$ del total:

$$\sum_{j=1}^{AVIONES} units_{1j} \geq porcentaje\_estandar * \sum_{i=1}^{BILLETES} \sum_{j=1}^{AVIONES} units_{ij}$$

- Un avión **no puede superar** su capacidad máxima de asientos:

$$\sum_{i=1}^{BILLETES} units_{ij} \leq asiento_j\ \forall j \in AVIONES$$

- El peso del equipaje **nunca puede superar** a la carga máxima del avión:

$$\sum_{i=1}^{BILLETES} carga\_billetes_i\ units_{ij} \leq carga\_aviones_j\ \forall j \in AVIONES$$

- Cada avión **mínimo** debe vender $min\_billetes_1$ Leisure plus:

$$units_{2j} \geq min\_billetes_1\ \forall j \in AVIONES$$

- Cada avión **mínimo** debe vender $min\_billetes_2$ de business plus:

$$units_{3j} \geq min\_billete_2\ \forall j \in AVIONES$$

- Para las variables de decisión:

$units_{ij}$ >= 0 donde $units_{ij} \in Z\ \ \forall i \in BILLETES\ y\ \forall j \in AVIONES$

Implementación

En cuanto a la implementación hemos distribuido los valores del problema en las secciones típicas que se crean en el modelado de un problema de programación lineal. Para ello hemos distribuido la hoja de cálculo de *Calc* de la siguiente manera:

- Variables: Aquí tenemos las variables de decisión en forma de matriz de trazabilidad, de esta manera, como hemos planteado anteriormente, relacionamos los tipos de billetes con los aviones. Estas variables de la matriz indican la cantidad de billetes de cada tipo para cada avión.
- Datos: En esta sección se dispone de los datos esenciales del problema como el precio de cada billete, la carga máxima permitida por billete, número de asientos por cada avión, capacidad máxima del avión, mínimo de billetes a vender de Leisure plus, mínimo de billetes de Business plus y el porcentaje de billetes estándar a vender del total.
- Función objetivo: Maximizar los ingresos de la aerolínea.
- Restricciones: Divididas en dos secciones, las del tipo >= y las del tipo <=. En estas restricciones se encuentran las capacidades máximas de cada avión y el número de asientos, así como la restricción del número de billetes de tipo estándar del total que tienen que venderse.

Datos a introducir en el Solucionador/Solver de la herramienta Calc:

Celda objetivo	$B$29	
Optimizar resaltados a	Máximo	
Cambiando las celdas	$C$6:$G$8	
Condiciones limitantes		
Referencia de celda	Operador	Valor
$B$37:$B$47	=>	$C$37:$C$47
$B$53:$B$62	<=	$C$53:$C$62
Opciones...		
Algoritmo del solucionador	Solucionador lineal de LibreOffice	
☑ Asumir variables como enteros		
☑ Asumir variables como no negativas		
☑ Limitar profundidad del algoritmo y límite		

Parte 2. Modelo avanzado en *GLPK*

Introducción

La misma compañía de aviación nos dice que tiene que asignar pistas para el aterrizaje de aviones. Para ello, tienen 4 pistas y cada una de las pistas tiene algunos horarios ya asignados para otros aviones y nos dan los horarios disponibles para aterrizar. Para ello, vamos a usar la siguiente tabla donde el 1 significa que ese horario está cogido y el 0 significa que está libre:

	0 a 15	15 a 30	30 a 45	45 a 60	60 a 75	75 a 90
Pista 1	1	1	1	1	0	0
Pista 2	1	1	0	0	1	1
Pista 3	0	0	0	1	1	0
Pista 4	0	1	1	1	0	0

1 implica que ese slot de tiempo está cogido y 0 indica que está libre. Todo el tiempo está expresado en minutos, lo que significa que el minuto 0 son las 9:00 y el minuto 90 las 10:30.

El problema de asignar un slot de cada pista viene dado por las siguientes condiciones:

- Cada avión tiene una hora programada de llegada al aeropuerto. Los slots no se pueden particionar por lo que se tiene que asignar a cada avión una hora de llegada entre minuto 0, 15, 30, 45, 60 o 75. Esto implica que cada minuto que un avión esté en el aire de más generará un gasto extra.
- Cada avión tiene una hora límite de llegada. Esto implica que, si un avión no llega antes de su hora límite, se quedará sin combustible con sus posibles consecuencias.

	Avión 1	Avión 2	Avión 3	Avión 4	Avión 5
Hora programada	10	-5	40	55	70
Hora límite	75	30	60	75	90
Coste/minuto	100€	200€	150€	250€	200€

Expresa las horas programadas y límite de cada avión y el coste/minuto por cada minuto en el aire.

- Cada avión debe tener su propio slot de aterrizaje que no puede ser compartido. Un slot por avión, y un avión por slot.
- Por temas de seguridad, no puede haber slots contiguos asignados.

Estos problemas generan las siguientes restricciones:

- La hora de aterrizaje de cada avión **debe estar entre** 0, 15, 30, 45, 60 o 75.
- Esta hora de aterrizaje debe ser **superior o igual** a la hora programada.
- Esta hora de aterrizaje debe ser **inferior o igual** a la hora límite.
- La suma de slots de un avión **debe ser igual** a 1.

- La suma de aviones de un slot **debe ser igual** a 1.
- No puede haber dos slots escogidos **contiguos** en una misma pista.
- Todos los slots elegidos **deben estar** libres.

Modelización

Como podemos comprobar, el objetivo de este ejercicio es conseguir el menor gasto posible. Este gasto sucede debido a que cada minuto que pasa un avión de más en el aire, este genera un coste por minuto. Para solucionarlo, vamos a plantearlo como un problema de asignación, donde tendremos una matriz de coste por slot de tiempo y una matriz binaria tridimensional donde el primer índice serán los aviones, el segundo las pistas y el tercero los slots.

En el problema, tenemos los siguientes conjuntos:
AVIONES = {a1, a2, a3, a4, a5}
PISTAS = {p1, p2, p3, p4}
SLOTS = {s1, s2, s3, s4, s5, s6}

Tenemos los siguientes parámetros:

Parámetros	Significado
$inicioSlot_i$	Es el momento en minutos cuando el slot inicia, donde i pertenece SLOTS.
$horaEsperada_i$	Es la hora esperada por avión, donde i pertenece a AVIONES.
$horaLimite_i$	Es la hora límite que puede estar esperando un avión en el aire, donde i pertenece a AVIONES.
$costeMinuto_i$	Es el coste por minuto que tiene cada avión por esperar, donde i pertenece a AVIONES.
$slotsDisponibles_{ij}$	Es la matriz que especifica qué slots al inicio del problema están libres, donde i pertenece a PISTAS y j pertenece a SLOTS.

Todo problema de asignación necesita dos matrices: la matriz de coste y la matriz de asignación.

- La matriz de coste se consigue de la siguiente forma:

$$costesSlot_{ij} = costeMinuto_i * (inicioSlot_j - horaEsperada_i) \ \forall i \in AVIONES \ y \ \forall j \in SLOTS$$

Con esta matriz conseguimos el gasto por aterrizar en un slot según la hora programada.

- La matriz de asignación o matriz con variables de decisión es:

$asignacionSlot_{ijk}$ = *matriz de asignación binaria* $\forall i \in AVIONES, \ \forall j \in PISTAS \ y \ \forall k \in SLOTS$

En el caso de que una celda tenga el valor de 1 significa que el avión **k** va a aterrizar en esa pista **j**, en ese slot de tiempo **k** y si es un 0, significa que esa celda no es elegida para aterrizar.

A ser un problema de asignación, el objetivo final es minimizar el gasto asignando aviones en los slots que menos dinero requieren para aterrizar en ellos. Nuestra función objetivo es:

$$MIN\ Gastos = \sum_{i=1}^{AVIONES} \sum_{j=1}^{PISTAS} \sum_{k=1}^{SLOTS} asignaciónSlot_{ijk} * costesSlot_{ik}$$

Restricciones:

- La suma de slots de un avión **debe ser igual** a 1.

$$\sum_{j=1}^{PISTAS} \sum_{k=1}^{SLOTS} asignacionSlot_{ijk} = 1\ \forall i \in AVIONES$$

- La siguiente restricción comprende varias restricciones a la vez:
  - La hora de aterrizaje de cada avión **debe estar entre** 0, 15, 30, 45, 60 o 75.
  - Esta hora de aterrizaje debe ser **superior o igual** a la hora programada.
  - *NUEVA*: El coste debe **ser mayor que** 0.

$$asignacionSlot_{ijk} * costesSlot_{ik} \geq 0\ \forall i \in AVIONES,\ \forall j \in PISTAS\ y\ \forall k \in SLOTS$$

- La siguiente restricción comprende dos a la vez:
  - La suma de aviones de un slot **debe ser igual** a 1.
  - Todos los slots elegidos **deben estar** libres.

$$asignacionSlot_{ijk} + slotsDisponibles_{jk} \leq 1\ \forall i \in AVIONES,\ \forall j \in PISTAS\ y\ \forall k \in SLOTS$$

- La hora de aterrizaje debe ser **inferior o igual** a la hora límite.

$$asignacionSlot_{ijk} * inicioSlot_k \leq horaLimite_i\ \forall i \in AVIONES,\ \forall j \in PISTAS\ y\ \forall k \in SLOTS$$

- Todos los slots elegidos **deben estar** libres.

$$\sum_{i=1}^{AVIONES} asignacionSlot_{ij(k-1)} + asignacionSlot_{ijk} \leq 1\ \forall j \in PISTAS\ y\ \forall k \in SLOTS / k \geq 2$$

- La variable de decisión:

$$asignacionSlot_{ijk} \in \{0,1\}\ donde\ \forall i \in AVIONES,\ \forall j \in PISTAS\ y\ \forall k \in SLOTS$$

En cuanto a la modelización del problema completo, el cual se basa en obtener el máximo beneficio, hemos unido ambos modelos (parte 1 y parte 2) , obteniendo que AVIONES solo es un conjunto único; los parámetros entre sí no tienen relación por lo que se añaden ambos; en las variables de decisión tenemos dos matrices, una para los billetes llamada $units_{ij}$ y otra para $asignacionSlot_{ikl}$ donde **i** son AVIONES, **j** son BILLETES, **k** son PISTAS y **l** son SLOTS; las restricciones tampoco tiene relación entre ellos por lo que se añaden todas; y, por último, la función objetivo resultante es:

$$MAX\ beneficioNeto = \sum_{i=1}^{AVIONES} \left(\sum_{j=1}^{BILLETES} (units_{ji} * precio\_billetes_j) - \sum_{k=1}^{PISTAS} \sum_{l=1}^{SLOTS} (asignacionSlot_{ikl} * costesSlot_{il}) \right)$$

Esta función objetivo funciona de la siguiente forma: Para un avión **i,** por una parte, se calcula sus ingresos totales de la suma de billetes **j** vendidos de ese avión por el precio de cada billete y, por otra parte, se obtiene el coste de vuelo mediante la pista **k** y el slot **l** seleccionado y se multiplica por el coste del tiempo en el aire hasta el horario de inicio del slot **l.** El resultado obtenido es el beneficio neto de la compañía.

Implementación

En cuanto a la implementación del modelo avanzado en *MathProg GLPK* hemos implementado el modelo de la parte 1 y después el modelo de la parte 2 por separado, tal y como se aconsejaba en el enunciado para el desarrollo de la práctica. Posteriormente tras verificar el correcto funcionamiento de ambas implementaciones en *MathProg* hemos fusionado ambos modelos creando uno que contemple la optimización para obtener el mayor beneficio posible. Para el desarrollo de esta parte hemos utilizado *glpk-utils* en *Linux* y para la ejecución de los modelos hemos ejecutado el siguiente comando dentro de la carpeta con los archivos: *glpsol -m part-2.mod -d part-2.dat -o output.txt*.

En la sección de datos almacenada dentro del fichero .dat hemos definido una serie de conjuntos (sets) en los cuales se encuentran los siguientes: BILLETES, AVIONES, BILLETES_MIN, PISTAS, SLOTS. Para la implementación de algunas reglas hemos decidido crear BILLETES_MIN como subconjunto de BILLETES, para aquellos billetes que necesitan un mínimo. En cuanto a slots hemos decidido identificarlos como un índice numérico por cuestiones de implementación en las restricciones.

En cuanto a los parámetros (param) que hemos creado, han sido vectores de datos, parámetros numéricos y matriz de slots disponibles, son los siguientes: asientos, carga_aviones, precio_billetes, carga_billetes, min_billetes, porcentaje_estandar, inicioSlot, horaEsperada, horaLimite, costeMinuto y slotsDisponibles.

Pasando al archivo .mod nos encontramos con las variables de decisión de ambas partes, la variable de decisión de unidades que vender de cada tipo de billete (*units*) y la variable de decisión binaria de asignación de slots a los aviones (*asignacionSlot*). Entrando en la función objetivo (*beneficioNeto*) hemos optado por hacer una optimización que mejora el rendimiento considerablemente al sacar el sumatorio de aviones fuera. Esta función objetivo se compone de ambas funciones objetivo de las dos partes de la práctica, la del cálculo de beneficios con los billetes y el cálculo de los gastos con las pistas de aterrizaje.
Respecto a las restricciones aplicadas tenemos las siguiente: *pbilletes_estandar, min_leisure, min_business, max_asientos, max_carga, llegadaAvion, costePositivo, unAvionSlot, limiteHorario y slotsContiguos*.

Parte 3. Análisis de los resultados

● Análisis del problema 1: Modelo básico.

Una vez completado el modelo en *Calc* y *GLPK*, hemos obtenido los siguientes resultados:

	AV1	AV2	AV3	AV4	AV5	**TOTAL**
Estándar	38	40	160	79	132	**449**
Leisure	21	27	23	61	23	**155**
Business	31	53	17	10	32	**143**

Con el valor de ***Z*** igual a:

Dinero obtenido con la venta de los billetes	**26190**

Como podemos comprobar, los resultados cumplen todos los requisitos propuestos:

- Un avión **no puede superar** su capacidad máxima de asientos.

TOTAL	**90**	**120**	**200**	**150**	**187**	**747**
Restricción	90	120	200	150	190	750

- El peso del equipaje **nunca puede superar** a la carga máxima del avión.

	AV1	AV2	AV3	AV4	AV5
Total	**1698**	**2700**	**1300**	**1699**	**1993**
Restricción	1700	2700	1300	1700	2000

- Cada avión **mínimo** debe vender 20 billetes Leisure plus
  Como se puede apreciar en la tabla de billetes se cumple.
- Cada avión **mínimo** debe vender 10 billetes de business plus.
  Como se puede apreciar en la tabla de billetes se cumple.
- El total de billetes estándar **mínimo** debe ser 60% del total.
  - Lo que significa que el valor otorgado, 449 es mayor que 448,2 (747*0,6 = 448,2) y por lo tanto cumple con la restricción.

Son restricciones limitantes: número máximo de asientos por avión, carga máxima por avión, porcentaje de billetes de tipo estándar del total.
No son restricciones limitantes: número mínimo de billetes business plus en cada avión, número mínimo de billetes de leisure plus.

Una vez obtenido los datos, podemos decir con certeza que se ha encontrado una región factible, donde solo existe una única solución óptima, debido a que dentro de esta región se ha encontrado un valor máximo para la función objetivo. Este valor es único y nos proporciona la cantidad máxima de beneficios aplicando las restricciones pedidas.

Análisis de la complejidad del problema:

Número de variables definidas: 15 variables de decisión

Número de parámetros: 11 parámetros

Número de restricciones definidas: 21 restricciones en total, más las dos que se especifican en el solucionador de valores enteros y variables positivas, 23 restricciones.

Modificación de parámetros del problema:

En el caso de que agregamos un avión más (Dos aviones más):

-**Caso avión B** (350 asientos, 3000 Kg de capacidad): ingreso de 37040 euros. Es generalizable a nuevos aviones, es decir la función objetivo encuentra un valor máximo dentro de la nueva región factible al añadir un avión que cumple con las restricciones.

-**Caso avión C** (35 asientos, 30 Kg de capacidad): beneficio de 0, dado que no encuentra solución, esto se debe a que no se pueden cumplir las restricciones a la vez de vender al menos 20 asientos de leisure plus, 10 de business y la de tener al menos el 60 por ciento de tipo estándar, dado que solo tenemos 35 asientos. Por otro lado, la limitación del equipaje. El peso total del equipaje es superior al permitido por el avión, por lo que no hay solución.

En el caso de que eliminemos un avión:

-**Eliminamos el avión a3**: En caso de eliminar el avión 3 los resultados obtenidos a partir del resto de aviones serían de unos beneficios de 19850 respecto a los 26190 que se obtenían cuando estaban los cinco aviones.

-**Eliminamos todos los aviones menos el avión a1**: En este caso encuentra solución con unos ingresos de 3110 euros.

● Análisis del problema 2: Modelo avanzado.

Tras la ejecución del modelo avanzado con todos los parámetros, obtenemos las siguientes asignaciones de pistas y slots para cada uno de los cinco aviones:

	9:00	9:15	9:30	9:45	10:00	10:15
P1					**A4**	
P2				**A3**		
P3		**A1**				
P4	**A2**					**A5**

Tabla de asignación simple

Con el valor de ***Z*** igual a:

Beneficio neto tras quitar gastos	**21190**

Gastos por las asignaciones de los aviones de **4.500** euros.

Como podemos comprobar, el problema cumple con todos los requisitos, añadiendo :

- La hora de aterrizaje de cada avión **debe estar entre** 0, 15, 30, 45, 60 o 75.

Cada avión tiene su slot de tiempo asignado, como se puede ver en la tabla de asignación simple.

- Esta hora de aterrizaje debe ser **superior o igual** a la hora programada y esta hora de aterrizaje debe ser **inferior o igua**l a la hora límite.

Como se puede comprobar a continuación, estas restricciones se cumplen.

	AV1	AV2	AV3	AV4	AV5
Hora programada	**9:10**	**8:55**	**9:40**	**9:55**	**10:10**
Hora Asignada	9:15	9:00	9:45	10:00	10:15
Hora límite	**10:15**	**9:30**	**10:00**	**10:15**	**10:30**

- La suma de slots de un avión **debe ser igual** a 1 y la suma de aviones de un slot **debe ser igual** a 1;
- Todos los slots elegidos **deben estar** libres y no puede haber dos slots escogidos **contiguos** en una misma pista.

Como se puede comprobar en la tabla de asignación simple, solo existe un avión por slot y un slot por avión. Después, los slots libres son del color verde claro y los no libres son del color gris oscuro. Como se puede comprobar en la tabla de asignación simple, ningún avión ha escogido un slot no libre.

Dados los resultados y conociendo que el problema cumple con todas las restricciones, podemos decir que es un problema factible de una única solución. Sí, es cierto que existen slots libres los cuales pueden ocupar ciertos aviones, como AV5 que puede ocupar el slot con hora de inicio igual a 10:15 en dos pistas distintas, pero esto no cambiaría en absoluto la optimización de la función objetivo. Por lo tanto, es factible con solución única acotada.

Entrando en el análisis del modelo avanzado vamos a considerar que se tiene que realizar una modificación en la modelización para incluir retrasos de todos los aviones. Este nuevo problema sería posible de implementar creando dos parámetros nuevos, uno de ellos en el que se incluya los minutos de retraso de cada uno de los aviones y luego otro que calcule el coste que tendría en total, para finalmente restarlo en la función objetivo. También habría que modificar la hora esperada para que no influya en la implementación y así elegir el slot óptimo dentro del retraso.
Si el avión 1 se retrasara 20 minutos sí que seguiría teniendo solución dado que todavía le queda tiempo hasta su hora límite. El resultado de la optimización nos quedaría de 20190 euros netos de beneficio, tras incluir los retrasos en el modelo. La disposición de los aviones en las pistas y slots de tiempo sería la siguiente:

	9:10	9:15	9:30	9:45	10:00	10:15
P1					**A4**	
P2				**A3**		
P3			**A1**			
P4	**A2**					**A5**

Como podemos apreciar A1 es el avión que sufre un retraso y la optimización permite colocarlo en el slot siguiente sin afectar a los demás aviones.

Comentando las ventajas y desventajas del uso de *Calc* frente a *MathProg-GLPK* son diversas. Primero hay que destacar la curva de aprendizaje de cada una de las herramientas, *Calc* es mucho más intuitivo y simple, en cambio MathProg al tratarse de un lenguaje de modelado matemático tiene una mayor curva de aprendizaje por lo que al inicio puede ser más costoso. Una vez comentada la dificultad y siendo totalmente objetivos con las herramientas que son, cabe destacar la fácil interpretación de los resultados que nos muestra *Calc*, siempre y cuando se haya realizado correctamente el problema. *GLPK* puede llegar a ser más tedioso y confuso a la hora de analizar los resultados, sobre todo si no se nombran correctamente los parámetros, variables y restricciones. En cuanto al apartado del rendimiento para problemas grandes y complejos va a ser siempre más rápido y eficiente el uso de *GLPK* dado que ha sido creado en el lenguaje *C* y gracias a su ligera interfaz el rendimiento es superior al de *Calc*. A nivel personal nos ha parecido mejor herramienta *MathProg-GLPK* por el hecho de ser más rápido y flexible a la hora de realizar modificaciones o añadir nuevos requisitos al problema modelado.
Para concluir esta parte, queremos añadir que *GLPK* es mucho mejor para agregar modelos y añadir o quitar datos en parámetros (como añadir o quitar columnas o filas en matrices), ya que no es necesario modificar la parte *.mod* (modelado). Es decir, un mismo modelo puede funcionar para distintos datos modificando los valores de los parámetros. En cambio, en *Calc* solo puedes modificar los datos de los parámetros, pero no puedes añadir ni quitar datos debido a que cambiaría toda la hoja.

Conclusiones de la práctica

En cuanto a las conclusiones que hemos sacado sobre la práctica, nos ha parecido que la carga de trabajo ha sido correcta y que las herramientas utilizadas nos han servido para dar un mejor enfoque a todo este bloque de la investigación operativa. El tema nos ha parecido muy interesante, dada la semejanza que tiene con un problema real dado que nos ha permitido ver la infinidad de casos a los que se le puede aplicar la optimización mediante programación lineal. Respecto a las competencias teóricas de la asignatura nos ha ayudado mucho en cuanto a la modelización de problemas de programación lineal, sobre todo en cuanto a problemas de asignación. Hemos logrado mejorar nuestra destreza y conocimiento sobre el uso de la hoja de cálculo, en este caso *Calc*, lo que nos parece algo fundamental. En cuanto al uso de *GLPK* nos ha parecido muy interesante el lenguaje de modelado matemático que presenta y la flexibilidad que este ofrece, ya que nos parece muy interesante que para distintos ficheros *.dat* se pueda utilizar un único *.mod*.
Los inconvenientes y dificultades que hemos tenido durante la realización de la práctica han venido sobre todo a la hora de realizar el modelaje de ambas partes y la curva de aprendizaje comentada anteriormente con *MathProg-GLPK*.