PRÁCTICA 1

Heurística y optimización - El problema de la compañía aérea

Mª Isabel Hernández Barrio

NIA: 100472315

31/10/2024

Universidad Carlos III de Madrid

Ingeniería Informática

Índice

Introducción

En este documento se presenta la primera práctica de la asignatura de Heurística y Optimización, que consiste en el modelado y resolución de un problema de programación lineal utilizando LibreOffice Calc y GLPK.

Este consta de un primer apartado explicando y justificando cómo se han planteado los modelos, seguido de un análisis de resultados, y termina con algunas conclusiones sobre el trabajo realizado.

Descripción de los modelos

Parte 1 - Venta de billetes con LibreOffice Calc

La función objetivo es la siguiente, teniendo AV = {AV1, AV2, AV3, AV4, AV5}. La letra E se usa para los billetes estándar, la L para leisure plus y la B para business plus. Las variables de decisión representan el número de billetes de cada tipo que se deben vender para cada avión.

$$\max\ z = \sum_{a \in AV} (19x_{E_a} + 49x_{L_a} + 69x_{B_a})$$

Y estas son las restricciones, que se cumplen para $\forall a \in AV$:

$$x_{E_a} + x_{L_a} + x_{B_a} \leq asientos_a$$
$$x_{E_a} + 20x_{L_a} + 40x_{B_a} \leq capacidad_a$$
$$x_{L_a} \leq 20$$
$$x_{B_a} \leq 10$$
$$\sum_{a \in AV} x_{E_a} \geq 0,6 \cdot \sum_{a \in AV} (x_{E_a} + x_{L_a} + x_{B_a})$$
$$x_{E_a},\ x_{L_a},\ x_{B_a} \geq 0$$

He decidido incluir tres variables por cada uno de los cinco aviones, una representando el número de billetes estándar a vender, la segunda con el número de billetes leisure plus y la tercera para los billetes business plus. Las he colocado en una especie de matriz 5x3 aviones - tipo de billete. Estas son las celdas que el solver debe cambiar para llegar a la solución óptima.

Seguidamente, se encuentra una tabla con los precios de cada tipo de billete y la función objetivo, que es una suma-producto entre las variables y sus precios correspondientes. He decidido dejar los precios duplicados pese a que todos los aviones tengan el mismo precio para cada tipo de billete, por cambios futuros que se pudieran hacer a estos precios, ya que en la vida real lo común es que cada avión (vuelo) tenga unos precios diferentes.

Después encontramos dos tablas con datos del enunciado: una con el equipaje permitido en kilogramos para cada tipo de billete; y otra con el número de asientos y capacidad que tiene cada avión.

Por último tenemos las restricciones:

- La primera, “no vender más billetes que el número de asientos disponible”, suma para cada avión las variables de los 3 tipos de billetes e indica que esta suma debe ser menor o igual al número de asientos de ese avión.
- La segunda, “no superar la capacidad máxima”, utiliza una suma-producto para coger las variables de los 3 tipos de billete de cada avión y multiplicar cada tipo con su respectivo equipaje permitido, e indica que esta suma-producto debe ser menor o igual a la capacidad máxima del avión correspondiente.
- La tercera restricción, “el mínimo de billetes leisure plus por avión debe ser de 20”, se modela simplemente cogiendo las variables de la segunda columna de la matriz (que corresponde a los billetes leisure plus de cada avión) e indicar que deben ser mayores o iguales que 20.
- La cuarta es similar, “el mínimo de billetes business plus por avión debe ser de 10”. Se seleccionan las variables de la tercera columna de la matriz (que corresponde a los billetes business plus de cada avión) y se indica que deben ser mayores o iguales que 10.
- Finalmente, la quinta restricción dicta que el número total de billetes estándar (suma de los billetes estándar de los cinco aviones) debe ser mayor o igual al 60% del número de billetes totales (0.6 multiplicado por la suma de todos los elementos de la matriz, todos los tipos de billetes de todos los aviones)
- Adicionalmente, se indica que todas las variables deben ser no negativas, condición que se ajusta en el apartado “Opciones…” del Solver.

Un apunte importante es que al resolver el problema con el Solver de LibreOffice Calc en opciones también he ajustado que las variables deben ser enteros, dado que no se puede vender 20.6 billetes, el número de billetes vendidos de cada tipo y avión debe ser un número no negativo y entero.

Parte 2 - Asignación de pistas más venta de billetes con GLPK

Estos son los conjuntos de nuestro problema, que se podrían modificar si quisiéramos. Los slots están definidos en el código de la forma en que se ve abajo, tomando como referencia temporal inicial la hora 8:55, es decir, que esa hora es el minuto 0 y el slot 9:00-9:15 se define como 5 (minuto 5, tomando la hora de inicio del slot), 9:15-9:30 es 5 + 15 = 20, etc. Lo hago de esta forma para poder sacar los minutos de retraso, ya que restar horas en el formato 9.55 no daría un resultado correcto.

$$AV = \{AV_1,\ AV_2,\ \ AV_3,\ \ AV_4,\ \ AV_5\}$$

$$PISTAS = \{P1,\ P2,\ P3,\ P4,\ P5\}$$

$$SLOTS = \{5,\ 20,\ 35,\ 50,\ 65,\ 80\}$$

Esta es la función objetivo, que se cumple para $\forall$ a$\in$AV, y las restricciones. r representa el costo por minuto de retraso del avión, s el slot de tiempo, hp la hora programada del avión, hl la hora límite y x_{nps} es una variable binaria que indica si el avión n es asignado a la pista p en el slot s o no. x_E, x_L y x_B son las variables de decisión, el número de billetes Estándar, Leisure y Business para cada avión a. Disponibilidad es una variable binaria que indica si la pista p está libre o no en el slot s; y consecutivos es otra variable binaria que indica si cada par de slots son consecutivos o no. Las restricciones están en el orden en que se explican más adelante (Restricción 1, Restricción 2…). **A estas restricciones se les añade las especificadas arriba, en la sección de la Parte 1**, ya que este problema también incluye el problema de la venta de billetes.
Nótese que para sacar el retraso de un avión se resta el slot de tiempo que se le asigna menos su hora programada, dado que la hora programada siempre debe ser anterior (menor) o igual al inicio del slot asignado, nunca mayor, de forma que no podría obtenerse un retraso negativo que invierta el signo de la sustracción del coste por retrasos.

$$\max\ z\ = \sum_{a \in AV} \left(19x_{E_a} + 49x_{L_a} + 69x_{B_a} - \left(x_{a_{P_S}} \cdot r_a \cdot (S - hp_a)\right)\right)$$

$$\sum_{p \in PISTAS} \sum_{s \in SLOTS} x_{a_{ps}} = 1 \quad \forall\, a \in AVIONES$$

$$\sum_{a \in AVIONES} x_{a_{ps}} \leq 1 \quad \forall\, p \in PISTAS,\ \forall\, s \in SLOTS$$

$$x_{a_{p_S}} \leq disponibilidad_{p_s} \qquad \forall a\ \in AVIONES,\ \forall\, p \in PISTAS\ ,\ \forall\, s \in SLOTS$$

$$s \cdot x_{a_{p_S}} \leq hp_a \cdot x_{a_{p_S}} \qquad \forall a\ \in AVIONES,\ \forall\, p \in PISTAS\ ,\ \forall\, s \in SLOTS$$

$$s \cdot x_{a_{p_S}} > hl_a \cdot x_{a_{p_S}} \qquad \forall a\ \in AVIONES,\ \forall\, p \in PISTAS\ ,\ \forall\, s \in SLOTS$$

$$\sum_{a \in AV} \left(x_{a_{ps_1}} + x_{a_{ps_2}}\right) \cdot consecutivos_{s_1 s_2}\ \leq\ 1 \quad \forall\, p \in PISTAS\ ,\ \forall\, s_1,\ s_2 \in SLOTS$$

Con el objetivo de pasarle al programa todos los datos fijos que necesita, aquellos que da el enunciado, he declarado los siguientes parámetros, cuyos valores se atribuyen en el .dat:

- hora_programada de cada avión
- hora_limite de cada avión
- coste_retraso, el coste adicional en € por minuto de retraso de cada avión
- disponibilidad, una variable binaria que indica si cada pista en cada slot de tiempo se puede utilizar o no
- asientos, el número de asientos con los que cuenta cada avión
- capacidad de cada avión
- precio_estandar (precio un billete estándar)
- precio_leisure_plus (precio un billete leisure plus)
- precio_business_plus (precio un billete business plus)
- consecutivos, una variable binaria que indica para cada par de slots de tiempo si estos son consecutivos o no

Respecto a las restricciones, se traducen a MathProg de la siguiente manera:

- Restricción 1. Cada avión debe tener asignado un slot de tiempo, ni más ni menos: para cada elemento de AVIONES, la suma de todas sus combinaciones con los diferentes elementos de PISTAS y SLOTS debe ser igual a 1.
- Restricción 2. Un slot de tiempo puede ser asignado a un solo avión o puede no estar asignado a ninguno, es decir, para cada combinación de pista + slot, la suma de los aviones atribuidos a esta debe ser 1 o 0.
- Restricción 3. Un slot debe estar libre para ser asignado, es decir, en la matriz “disponibilidad” el par pista-slot asignado al avión debe tener el valor 1 (disponible).
- Restricción 4. El slot de aterrizaje debe ser igual o posterior a la hora de llegada, lo que se traduce como que todas las combinaciones de avión + pista + slot, si la combinación está asignada (asignado[a, p, s]=1), el slot será igual o mayor (posterior) a la hora programada.
- Restricción 5. El slot de aterrizaje debe ser igual o anterior a la hora límite, por lo que para todos los valores de las combinaciones de estos tres set, si la combinación se asigna la restricción asegurará que s es menor o igual a hora_limite.
- Restricción 6. No asignar dos slots de tiempo consecutivos en la misma pista. Esto se consigue especificando que para cada pista y combinación de dos slots seguidos (el par de slots en consecutivos es 1), la suma de estas dos combinaciones avión-pista-slot debe ser 1 o menor, ya que si ambos estuvieran asignados así (ambos fuera 1), la suma sería 2, y esto no puede ocurrir.
- Restricción 7. No vender más billetes que el número de asientos disponibles en cada avión, es decir, para cada avión, su valor x_estandar sumado al de x_leisure y x_business debe ser menor o igual al valor de asientos para ese avión.

- Restricción 8. No se puede superar la capacidad máxima de cada avión (1kg billetes estándar, 20kg leisure, 40kg business). Para cada avión, el valor de la variable de decisión x_estandar más el de x_leisure multiplicado por 20 y el de x_business multiplicado por 40, debe ser menor o igual a el parámetro capacidad para ese avión.
- Restricción 9. Mínimo 20 billetes leisure plus por avión: el valor de la variable x_leisure de cada avión debe ser mayor o igual a 20.
- Restricción 10. Mínimo 10 billetes business plus por avión: el valor de la variable x_business de cada avión debe ser mayor o igual a 10.
- Restricción 11. El número total de billetes estándar debe ser al menos el 60% del total de billetes ofertados: se suman todos los valores de x_estandar de cada avión y esto debe ser mayor o igual a 0.6 multiplicado por la suma del valor de x_estandar + x_leisure + x_business de todos los aviones.

La función objetivo en el código MathProg es la maximización de: la suma-producto de los precios de los billetes (precio_estandar, precio_leisure, precio_business) y las variables de decisión (x_estandar[a], x_leisure[a], x_business[a]) para cada avión a, menos la suma del coste de retraso por minuto (coste_retraso[a]) multiplicado por la diferencia de tiempo entre el slot en cuestión(s) y la hora programada del avión (hora_programada[a]) y esto multiplicado por la variable binaria de cada combinación avión-pista-slot (asignado[a, p, s], de manera que solo resten aquellas combinaciones que hayan sido asignadas), para cada combinación de estas.

De nuevo, para que el cálculo del retraso de los aviones sea correcto, en vez de operar con horas (9.00, 9.15, etc), en el .dat las horas se expresan en minutos tomando de referencia las 8.55 como 0 minutos, de manera que las horas tienen esta correspondencia con los minutos:

Slot en hora	Slot en minutos
9.00 - 9:15	5
9.15 - 9:30	20
9.30 - 9:45	35
9.45 - 10:00	50
10.00 - 10.15	65
10.15 - 10:30	80

Análisis de los resultados

Resultados obtenidos

La **solución óptima** que he obtenido produce **21 690 €** de beneficios (26 190 € de ingresos por la venta de billetes, menos 4 500 € de gastos por retrasos en las pistas):

Avión	Time Slot	Pista	Billetes Estándar	Billetes Leisure Plus	Billetes Business Plus
AV1	9:15 - 9:30	P3	19	58	13
AV2	9:00 - 9:15	P4	38	31	51
AV3	9:45 - 10:00	P2	160	23	17
AV4	10:00 - 10:15	P1	100	20	30
AV5	10:15 - 10:30	P4	133	21	36

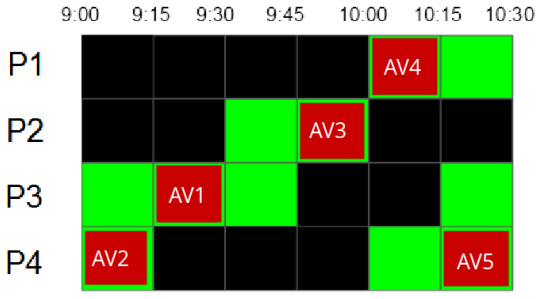
Como vemos se cumplen todas las **restricciones**, dado que:

- Restricción 1. Cada avión es asignado a un slot, ni más ni menos, como vemos en la tabla

- Restricción 2. Un slot de tiempo puede ser asignado a un solo avión, puede no estar asignado a ninguno. No vemos ningún slot de tiempo que se repita o se use dos veces.

- Restricción 3 y 6. Todos estos slots que hemos obtenido para asignar a los 5 aviones están disponibles (marcados en verde en la tabla dada). Y no se asignan slots consecutivos en la misma pista:

Slot	Pista
9:15 - 9:30	P3
9:00 - 9:15	P4
9:45 - 10:00	P2
10:00 - 10:15	P1
10:15 - 10:30	P4



- Restricción 4. Los slots de aterrizaje son iguales o posteriores a la hora de llegada

Hora de llegada	Slot	Pista
9:10	9:15 - 9:30	P3
8:55	9:00 - 9:15	P4
9:40	9:45 - 10:00	P2
9:55	10:00 - 10:15	P1
10:10	10:15 - 10:30	P4

- Restricción 5. Los slots de aterrizaje son iguales o anteriores a la hora límite

Slot	Hora Límite
9:15 - 9:30	10:15
9:00 - 9:15	9:30
9:45 - 10:00	10:00
10:00 - 10:15	10:15
10:15 - 10:30	10:30

- Restricción 7. No se venden más billetes que el número de asientos disponibles en cada avión:

$$19 + 58 + 13 = 90 <= 90$$
$$38 + 31 + 51 = 120 <= 120$$
$$160 + 23 + 17 = 200 <= 200$$
$$100 + 20 + 30 = 150 <= 150$$
$$133 + 21 + 36 = 190 <= 190$$

- Restricción 8. No se supera la capacidad máxima de cada avión (sabemos que se permite a cada pasajero portar 1 kg para billetes estándar, 20 kg para leisure y 40 kg para business):

$$19*1 + 58*20 + 13*40 = 1699 <= 1700$$
$$38*1 + 31*20 + 51*40 = 2698 <= 2700$$
$$160*1 + 23*20 + 17*40 = 1300 <= 1300$$

100*1 + 20*20 + 30*40 = 1700 <= 1700
133*1 + 21*20 + 36*40 = 1993 <= 2000

- Restricción 9. Se ofertan mínimo 20 billetes leisure plus por avión: 58, 31, 23, 20, 21 todas mayores o iguales que 20

- Restricción 10. Se ofertan mínimo 10 billetes business plus por avión: 12, 51, 17, 30, 36 todas mayores o iguales que 10

- Restricción 11. El número total de billetes estándar es al menos el 60% del total de billetes ofertados:

  19 + 38 + 169 + 100 + 133 = 459 billetes estándar en total
  459 + 58 + 31 + 23 + 20 + 21 + 90 + 120 + 200 + 150 + 190 = 1362 billetes en total
  0.6 * 1362 = 817.2 >= 459

La **restricción que limita la solución es la primera**, “no se pueden vender más billetes que la capacidad máxima de cada avión”, ya que los valores óptimos corresponden con vender exactamente el máximo de billetes posibles. Si se pudieran vender más billetes, la función objetivo podría crecer.

Esto casi también sucede con la segunda restricción, “no se puede superar la capacidad máxima de cada avión”, ya que los valores que toman las variables de decisión son muy cercanos al máximo y exactamente el máximo para varias de estas.

Análisis de complejidad

He utilizado **15 variables de decisión** (3 por cada avión de los 5 que tenemos) y un total de **11 restricciones**, una para cada condición especificada en el enunciado.

Poniendo **30 asientos más** en cada avión, se venden más billetes y los beneficios aumentan. Sin embargo, observamos que se venden menos billetes business, probablemente porque no se ha aumentado la capacidad de los aviones y este último tipo de billete permite demasiados kilogramos de equipaje:

ORIGINAL - Estándar: 450, Leisure: 153, Business: 147,
beneficio_total.val = 21 690
MÁS 30 ASIENTO -: Estándar: 540, Leisure: 278, Business: 82,
beneficio_total.val = 25040

Ahora bien, si **aumentamos la capacidad de los aviones por 1000 k**g, aumentarán todavía más los beneficios, puesto que la compañía podrá cobrar más billetes business plus, que son mucho más caros. Para haber aumentado tanto la capacidad, los billetes estándar se mantienen en el mismo valor e incluso los de leisure plus disminuyen significativamente:

ORIGINAL - Estándar: 450, Leisure: 153, Business:
147, beneficio_total.val = 21 690
MÁS 30 ASIENTOS Y 1000 KG DE CAPACIDAD - 540, Leisure: 100, Business: 260, beneficio_total.val = 28600

Si **duplicamos el costo de retraso** por minuto de los aviones, obtenemos los mismos billetes a vender de cada tipo y avión, las mismas asignaciones de pistas y slots, pero se ve una disminución de los beneficios a 17 190 € puesto que los gastos ascienden a 9 000 €, exactamente el doble.

Si **quitamos 2 pistas**, la P3 y la P4 del problema, obtenemos que este no tiene una solución primal factible, ya que no es posible agendar una pista en un slot correcto a cada uno de los aviones. Así mismo, si con las pistas originales **añadimos más aviones y hacemos las horas de todos los aviones coincidir**, obtenemos que el problema no tiene una solución factible tampoco, lo que nos hace pensar que todo funciona correctamente y simula la realidad como se espera.

Retrasos de los aviones

La implementación de retrasos para los aviones en mi modelo supondría un cambio en los valores de hora_programada[a]: habría que sumarle el tiempo de retraso a estos valores. Por lo tanto, añadiría un **parámetro nuevo retraso{AVIONES}**, y asignaría valores de 0 para todos los aviones en el .dat. En el momento que se prevea un retraso para algún avión, simplemente habría que cambiar un número en ese archivo, añadiendo el número de minutos de retraso. En el archivo .mod, cada vez que se haga un cálculo o una restricción con los parámetros nombrados anteriormente (hora_programada[a]), habría que sumarles retraso[a].

Al sumarle un retraso de 20 minutos al Avión 1 (AV1), el reparto de pistas y slots no cambia, solamente cambia para AV1, que se mueve al slot de las 9.30 - 9:45 en la misma pista. Ese retraso supone que el avión llegue a las 9:30 en vez de a las 9:10, es decir que llega justo cuando empieza su nuevo slot asignado, de forma que su tiempo de retraso se reduce a 0 minutos en vez de 5 y se produce un ahorro de 500 €.

LibreOffice Calc y GLPK, ventajas y desventajas

La principal ventaja de utilizar LibreOffice para resolver este tipo de problemas es que se trata de una herramienta **visual**, podemos encontrar y seleccionar un parámetro o variable o restricción rápidamente en el caso de que queramos modificarlo o simplemente consultarlo. Está muy claro qué elemento corresponde a qué y su significado, además de poder añadirle estilos y formatos a la hoja de cálculo

para hacerlo incluso más sencillo. Mientras tanto, GLPK no cuenta con una interfaz gráfica de usuario.

Además, al **no tener que escribir código**, es algo mucho más fácil de usar y accesible para cualquiera, sepa o no programar, dado el uso de herramientas de Hojas de cálculo (Excel, LibreOffice…) es más común para estudiantes y trabajadores de cualquier área.

También, para resolver un problema de este tipo, es suficiente con un solver que viene incluido en las herramientas de LibreOffice, que **ya viene instalad**o en tu sistema operativo, mientras que GLPK es necesario instalarlo manualmente, aunque tampoco es complicado, lo cual está muy bien.

El problema de LibreOffice que GLPK viene a solucionar, es la **dimensión de los problemas que puede resolver**, ya que al meter algo más complejo como la parte 2 de este problema, sería más complicado y confuso el implementarlo en una hoja de cálculo, mientras que utilizando MathProg se puede conseguir con una lógica de programación sencilla.

Otra ventaja que veo en GLPK, es el hecho de que requiera un archivo con el modulado (.mod) y otro con los datos (.dat), ya que de esta manera se hace mucho **más fácil la modificación de datos** al estar todos en un mismo archivo bastante sencillo de hacer y entender.

Conclusiones sobre la práctica

Me ha parecido muy interesante trabajar con estas herramientas para la resolución de problemas de programación lineal, además de ser una forma relativamente rápida de hacerlo y tener una curva de aprendizaje bastante asumible.

Lo que me ha parecido más desafiante ha sido aprender a implementar restricciones que implican comparación de tiempo y especialmente la de asegurar que no se asignen dos slots consecutivos de tiempo. El trabajar con conjuntos avión-pista-slot también es un concepto que me pareció un poco complicado de entender y aplicar al principio, pero muy útil cuando te acostumbras a ello.

Este proyecto también me ha enseñado que en programación lineal no se usan condicionales, que es algo que no tenía del todo claro, y me ha hecho trabajar en la obtención de soluciones que cumplan la función de estos respetando la definición y barreras de la programación lineal.

En conclusión, esta práctica me ha ayudado enormemente a afianzar mis conocimientos respecto al planteamiento y resolución de problemas de optimización de este tipo, además de enseñarme herramientas muy convenientes para esto.