uc3m | Universidad Carlos III de Madrid

# Heurística y Optimización

Inés Fuai Guillén Peña | 100495752@alumnos.uc3m.es | g85

Nicola Stefania Cindea | 100495713@alumnos.uc3m.es | g85

# ÍNDICE

1. Introducción

2. Parte 1: Modelización del problema

3. Parte 2: Modelización del problema

    - Implementación en Mathprog

4. Análisis de resultados

5. Conclusión

## 1. Introducción

Este documento presenta los resultados de la primera práctica de la asignatura de Heurística y Optimización. En esta práctica, se nos solicita modelar dos problemas utilizando programación lineal, así como llevar a cabo sus implementaciones en Calc y MathProg. Además, debemos obtener los resultados correspondientes y analizarlos exhaustivamente..

## 2. Parte 1: Modelización del problema.

En primer lugar, leemos y analizamos el enunciado de la práctica. Una vez comprendido el problema, lo planteamos como un problema de Programación Lineal entera. Antes de empezar, debemos definir la notación que vamos a utilizar:

- N = {AV1, AV2, AV3, AV4, AV5} → Conjunto de aviones disponibles donde cada AVi representa un avión.
- T = {E, LP, BP} → Conjunto de tarifas donde E es Estándar, LP es Leisure Plus, y BP es Business Plus.
- n = |N| = 5 = número de aviones.
- m = |T| = 3 = número de tarifas disponibles.
- xij representa el número de billetes de la tarifa j ofertados en el avión i
- Pj = {19, 49, 69} → Vector de precios correspondientes a las tarifas (Estándar, Leisure Plus, Business Plus).
- S = {90, 120, 200, 150, 190} → Vector que representa el número de asientos disponibles en cada avión i
- C = {1700, 2700, 1300, 1700, 2000} → Vector que representa la capacidad máxima de equipaje (en kg) para cada avión
- W hace referencia al peso de equipaje permitido por cada tarifa: wE=1kg, wLP=20kg, wBP=40kg
- k es la capacidad mínima de billetes de Leisure Plus que se deben ofrecer: kLP = 20
- p es la capacidad mínima de billetes de Business Plus que se deben ofrecer: pBP = 10
- Z será la función objetivo. Representará el precio total.

Primero, identificamos las variables de decisión.
<u>Variables de Decisión</u>

- xij: Número de billetes de la tarifa j que se ofertan en el avión i, donde:
  - i = 1, 2, 3, 4, 5 (correspondiente a los aviones AV1, AV2, AV3, AV4, AV5)
  - j = 1, 2, 3 (donde j = 1 para Estándar, j = 2 para Leisure Plus y j = 3 para Business Plus)

A continuación, definimos las restricciones del problema. Cabe mencionar que la mayoría de restricciones se encuentran explícitas en el enunciado del problema.

Restricciones

- No se pueden vender más billetes que el número de asientos disponibles en el avión.

$$xi1 + xi2 + xi3 \leq Si \; \forall i$$

(Donde S1=90,S2=120,S3=200,S4=150,S5=190)

- No se puede superar en ningún caso la capacidad máxima de cada avión

$$w1 * xi1 + w2 * xi2 + w3 * xi3 \leq Ci \; \forall i$$

(Donde w1=1 kg (Estándar), w2=20 kg (Leisure Plus), w3=40 kg (Business Plus), C1=1700 kg,C2=2700 kg,C3=1300 kg,C4=1700 kg,C5=2000 kg)

- Para cada avión, se deben ofertar como mínimo 20 billetes leisure plus y 10 billetes business plus.

$$xi2 \geq 20 \; \forall i, \; xi3 \geq 10 \; \forall i$$

- Además, al tratarse de una compañía de bajo coste, el número de billetes estándar total debe ser al menos un 60%de todos los billetes que se ofertan.

$$xi1 \geq 0,6 * (xi1 + xi2 + xi3) \; \forall i$$

- Las variables de decisión no pueden ser negativas

$$xij \geq 0 \; \forall i, j$$

Para terminar con el planteamiento del problema, definimos la función objetivo. Va a ser de maximización ya que se nos pide maximizar el beneficio total.

Función Objetivo

$$maxZ = 19\sum(i = 1 \ldots 5)\, xi1 + 49\sum(i = 1 \ldots 5)\, xi2 + 69\sum(i = 1 \ldots 5)\, xi3$$

Una vez definido el modelo, lo trasladamos a un fichero Calc, tal y como se nos pide en el documento. Para llevar esto a cabo, representamos en el fichero las variables de decisión, la función objetivo, los parámetros y las restricciones, sustituyendo el valor de los parámetros por aquellos indicados en el enunciado del problema. Nuestros parámetros serán C, S, a, w, P.

Por último, utilizamos la herramienta Solver de Calc para maximizar la función objetivo, obteniendo la solución óptima al problema de acuerdo con nuestro planteamiento: z = 26190

## 3. Parte 2: Modelización del problema

En esta parte, nos encontramos ante un problema similar al anterior, pero con ligeros matices y nuevas restricciones. El problema es la asignación de slots de aterrizaje a los aviones minimizando los costes por retraso.

Por esta razón, debemos definir nuevos datos. Esta será la notación que usaremos:

- A = {AV1, AV2, AV3, AV4, AV5} → Conjunto de aviones que deben aterrizar.
- S = {S1, S2, ..., S6} → Conjunto de slots disponibles en las pistas
- |A| = 5 → Número de aviones.
- |S| = 6 → Número de slots disponibles
- Oxj → Matriz binaria que indica si el slot j está disponible para el avión x (1 si está disponible, 0 si no).
- txj → Matriz de tiempos que indica el retraso (en minutos) si el avión x aterriza en el slot j. El valor es 0 si el slot es igual a la hora programada de aterrizaje y positivo si es posterior.
- pxj → Variable binaria que indica si el avión x aterriza en el slot j (1 si lo hace, 0 si no).
- Cx→ Matriz que representa el coste por retraso para cada avión x, calculado como el número de minutos de retraso multiplicado por el coste por minuto de retraso especificado para cada avión.

Al igual que en el problema anterior, definimos las variables de decisión que serán fundamentales para la solución del problema.
Variables de Decisión

- pxj: Esta es la variable de decisión principal. Representa si el avión x aterriza en el slot j. Es una variable binaria, donde:
    - pxj = 1 si el avión x aterriza en el slot j
    - pxj = 0 si el avión x no aterriza en el slot j

En cuanto a las restricciones, mantendremos las de la parte 1, añadiendo aquellas que controlan todo lo relacionado con las nuevas variables de decisión.
Restricciones

- Todos los aviones tienen que tener asignado un slot de tiempo para efectuar el aterrizaje.

$$\sum(j \in S)\ pxj \ = \ 1, \ \ \forall x \in A$$

- Un slot de tiempo puede estar asignado como máximo a un avión.

$$\sum(j \in S)\ pxj \leq 1, \ \ \forall j \in S$$

- El slot que se asigna a un avión debe ser un slot libre

$$pxj \leq Oxj \; \forall x \in A, j \in S$$

- El inicio del slot de aterrizaje debe ser igual o posterior a la hora de llegada del avión.

$$pxj = 0 \; si \; tj < Tprog, x \, , \; \forall x \in A, \; j \in S$$

- El inicio del slot de aterrizaje debe ser igual o anterior a la hora límite de aterrizaje del avión.

$$pxj = 0 \; si \; tj > Tlim, x \, , \; \forall x \in A, \; j \in S$$

- Por cuestiones de seguridad, no se pueden asignar dos slots consecutivos en la misma pista.

$$pxj + px(j + 1) \leq 1, \forall j \; consecutivos \; en \; la \; misma \; pista$$

Función Objetivo

El objetivo es minimizar el coste adicional por retrasos para todos los aviones. Para ello se debe calcular el costo de retraso acumulado para todos los aviones en función del tiempo de aterrizaje en cada slot.

$$MinZ = \sum(x \in A)\sum(j \in S) \; txj \cdot cx \cdot pxj$$

Donde:

- txj es el retraso en minutos para el avión x aterrizando en el slot j.
- Cx es el costo por minuto de retraso para el avión x.
- pxj es la variable binaria que indica si el avión x aterriza en el slot j

### Implementación en mathprog

Para ambas partes, creamos un fichero de datos .dat con los correspondientes valores de los conjuntos y los parámetros. Creamos otro fichero .mod donde se encontrarán las restricciones, variables de decisión y declaración de parámetros.

## 4. Análisis de resultados

En cuanto a la primera parte, el resultado muestra un valor óptimo de Z = 26190 euros, lo que implica que hemos logrado el máximo beneficio posible para la compañía, teniendo en cuenta las restricciones impuestas. El cumplimiento de estas restricciones indica que la solución obtenida es viable y razonable dentro del contexto del problema analizado.

Al ejecutar el mismo modelo en Mathprog (GLPK), hemos obtenido nuevamente un resultado de z = 26190. El hecho de que los resultados entre las distintas herramientas de optimización sean iguales muestra que el planteamiento del problema es correcto y que el modelo ha encontrado una solución óptima de manera eficiente.

Al analizar las restricciones, podemos afirmar que la capacidad de los aviones es un factor clave: si los aviones operan cerca de su capacidad máxima, esto podría limitar nuestro potencial de beneficio. Asimismo, los mínimos establecidos para las tarifas Leisure y Business Plus son también significativos: un cambio en la cantidad de billetes vendidos en estas categorías podría impactar el cumplimiento de las restricciones y, por ende, el beneficio total.

En cuanto a la complejidad del problema, definimos 15 variables y 6 restricciones. Además, al resolver el modelo, se agregan variables de holgura y artificiales. Cambiar el problema, tanto añadir como eliminar aviones, puede alterar significativamente la complejidad. Por ejemplo, sumar un nuevo avión aumentaría tanto el número de variables como el de restricciones, complicando la resolución y alargando el tiempo necesario para encontrar la solución óptima. Por otro lado, si modificamos las tarifas o ajustamos los precios, esto podría cambiar la dinámica de ventas y afectar la rentabilidad, haciendo que el problema sea más o menos complicado según cómo realicemos esos ajustes.

Refiriéndonos ahora a la segunda parte, obtenemos un resultado de Z = 21690. A modo de comparación, el valor de la primera parte del modelo (sin tener en cuenta la asignación de slots) alcanza Z = 26190. La diferencia de 4500 euros muestra que la introducción de restricciones adicionales en la gestión de las máquinas ha tenido un impacto negativo en los beneficios globales.

Obtener este resultado más bajo se puede deber a varias razones. En primer lugar, el número limitado de slots y las restricciones de tiempo obligan a algunos aviones aceptar horarios que no son los mejores para maximizar las ganancias, esto impide que aterricen en los horarios más convenientes, reduciendo el ingreso potencial por venta de billetes. Además, las restricciones que prohíben el uso de asientos consecutivos en la misma pista y que requieren que cada avión ocupe un horario disponible también limitan la flexibilidad de asignación. En la primera parte del modelo, estas restricciones no existen, por lo que las ganancias se pueden maximizar de manera más eficiente, lo que explica el mayor valor Z.

El modelo actualizado que incluye la asignación de slots introduce una complejidad adicional. Tras un análisis más detallado de la solución, hemos determinado que la capacidad máxima del avión es una de las restricciones más importantes. Si tenemos aviones más grandes, podremos transportar más pasajeros y equipaje, lo que aumentará los ingresos. Además, los estrictos calendarios de despegue y aterrizaje pueden limitar la flexibilidad de la misión.

Agregar más aviones o asientos puede complicar mucho la resolución del problema. Cada nuevo avión o slot crea más combinaciones posibles, lo que hace que el tiempo para encontrar la mejor solución sea mayor. Esta mayor complejidad puede resultar en soluciones menos efectivas, ya que el modelo podría no encontrar la opción ideal en un tiempo razonable. Por otro lado, si añadimos más pistas, esto ofrecerá más flexibilidad en la planificación de vuelos, lo que ayudará a utilizar mejor los recursos y, en consecuencia, aumentar los ingresos.

El modelo actualizado define un total de 135 variables y 310 restricciones. Esta complejidad surge de la necesidad de gestionar las decisiones y restricciones de asignación de franjas horarias para garantizar que la asignación de franjas horarias sea eficaz y eficiente.

En la segunda parte de la práctica, es importante tener en cuenta los retrasos en las llegadas de los vuelos, ya que pueden afectar mucho la asignación de aviones a las pistas de despegue. Para solucionar este problema, necesitamos agregar nuevas variables y restricciones al modelo actual que simulan cómo los retrasos impactan las operaciones.

Primero, hay que incluir variables que representan el tiempo de retraso de cada avión. Por ejemplo, si el avión AV1 tiene un retraso de 20 minutos, se debe definir un parámetro adicional *RETRASO* para reflejar ese retraso. Además de estas variables, es necesario ajustar los horarios asignados a cada avión. Un retraso en la llegada significa que un avión no estará disponible para su próxima asignación a tiempo, lo que puede afectar la programación de otros vuelos. Por lo tanto, debemos modificar la asignación de horarios para asegurarnos de que un avión no sea asignado a una pista hasta que haya terminado su tiempo de retraso.

Al ejecutar los nuevos archivos .mod y .dat con GLPK modificados para este caso, obtenemos un nuevo resultado de Z = 20190. En base a los resultados que nos da, la nueva asignación de aviones a pistas y slots de tiempo sería la siguiente:

- **Avión A1** asignado a **Pista P3**, **Slot S3** (1)
- **Avión A2** asignado a **Pista P3**, **Slot S1** (1)
- **Avión A3** asignado a **Pista P2**, **Slot S4** (1)
- **Avión A4** asignado a **Pista P1**, **Slot S5** (1)
- **Avión A5** asignado a **Pista P4**, **Slot S6** (1)

Para terminar con el análisis de resultados, vamos a realizar una breve comparación de las 2 herramientas utilizadas para resolver el problema de Programación Lineal. Por una parte, el programa Calc es, en un principio, más sencillo de utilizar debido a que es más intuitivo. Por esta razón es recomendable para personas que no saben o son novatos programando. Aunque es adecuada para problemas sencillos, puede volverse limitada ante modelos más complejos que requieren un modelado detallado.

GLPK (GNU Linear Programming Kit), por su parte, es más flexible, es decir, más cómodo a la hora de realizar modificaciones. Además, el problema se encuentra mejor estructurado. Otra ventaja que podemos obtener de este programa con respecto a Calc es la identificación de errores. En nuestra opinión, una vez has aprendido el lenguaje, Mathprog es más sencillo y rápido de utilizar a la hora de modelar un problema de Programación Lineal. También hemos de mencionar que el hecho de haberlo modelado previamente en Calc nos ha ayudado a la hora de usar Mathprog

## 5. Conclusión

En resumen, el modelo que desarrollamos nos permite encontrar soluciones óptimas a los problemas planteados en esta práctica. Además, confirmamos que estos modelos son aplicables a diferentes situaciones siempre que los parámetros tengan un sentido lógico y realista. Durante el transcurso del trabajo, el modelo continúa evolucionando. No obtuvimos el modelo final en nuestro primer intento; en cambio, ajustamos las restricciones y refinamos el método hasta que llegamos a la solución deseada.

En términos de desafíos, la primera parte del trabajo fue particularmente desafiante. Al principio no teníamos experiencia con matrices en problemas de programación lineal y estábamos un poco perdidas. Sin embargo, a través de la búsqueda de información pudimos comprender mejor el uso de matrices, lo que nos facilitó el desarrollo de la segunda parte de nuestro trabajo.

Otra dificultad es la implementación de restricciones que imposibilitan la asignación de aviones a pistas utilizadas. Inicialmente, intentamos modelar esta limitación para evitar la repetición de franjas horarias. Sin embargo, cuando se ejecuta GLPK, encontramos que los errores asociados con la multiplicación de variables de decisión comprometen la linealidad del modelo. Afortunadamente, pudimos resolver este problema reformulando la restricción de suma, simplificando así la implementación.

En resumen, esta práctica nos brinda una valiosa oportunidad para aprender a modelar problemas complejos en programación lineal, mejoramos nuestras habilidades matriciales y aprendimos a utilizar herramientas como Calc (Solver) y Mathprog (glpsol) para resolver este tipo de problemas.