Práctica: *Programación Lineal.*
Heurística y Optimización
Grado de Ingeniería en Informática. Curso 2020-2021.
Departamento de Informática

1. Objetivo

El objetivo de esta práctica es que el alumno aprenda a modelar problemas de programación lineal y a resolverlos con dos tipos de herramientas: hojas de cálculo y algoritmos de resolución usando técnicas de modelización.

2. Enunciado del problema

Una importante compañía aérea se ha puesto en contacto con alumnos del curso de Heurística y Optimización. Tras varias entrevistas os han seleccionado a tu compañero y a ti para dar solución a varios problemas mediante técnicas de Programación Lineal.

2.1. Parte 1: Modelo básico en Calc

Uno de los problemas a los que se enfrentan las compañías aéreas es el número de billetes que han de ofertar para cada una de las tarifas que tienen disponibles. En este sentido, una compañía aérea nos informa de que tiene tres tipos de tarifas disponibles: estándar, leisure plus, y business plus, y cada una aporta una serie de ventajas al pasajero. Estas ventajas están descritas en la Tabla 1.

Tarifa	Equipaje permitido	Precio
Estándar	1 kg.	19€
Leisure plus	20 kg.	49€
Business plus	40 kg.	69€

Tabla 1: Descripción de las tarifas de la compañía aérea.

La Tabla 1 muestra por cada tarifa el equipaje permitido, y el precio de un billete para esa tarifa. Así, por ejemplo, la tarifa estándar permite que el pasajero lleve equipaje de mano de máximo 1 kg. por un precio de 19€. Además, la compañía nos informa de que dispone de una flota compuesta de 5 aviones cuyas características se muestran en la Tabla 2.

Avión	Asientos	Capacidad
AV1	90	1.700 kg.
AV2	120	2.700 kg.
AV3	200	1.300 kg.
AV4	150	1.700 kg.
AV5	190	2.000 kg.

Tabla 2: Descripción del número de asientos y la capacidad de los aviones de la compañía.

En la Tabla 2 se muestra la cantidad de asientos y la capacidad de equipaje total que puede transportar cada avión. Se pide determinar el número de billetes de cada tarifa que se deben ofertar para cada avión con el fin de maximizar el beneficio de la empresa considerando que:

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

- No se puede superar en ningún caso la capacidad máxima de cada avión.
- Para cada avión, se deben ofertar como mínimo 20 billetes leisure plus y 10 billetes business plus.
- 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.

Se pide:

1. Modelar el problema como un problema de Programación Lineal entera.
2. Implementar el modelo en una hoja de cálculo (LibreOffice).

2.2. Parte 2: Modelo avanzado en GLPK

Otro de los problemas importantes que afrontan los aeropuertos cada día es la asignación de pistas para el aterrizaje de los aviones. En este sentido, un aeropuerto nos comunica que tiene cuatro pistas de aterrizaje, cada una con la disponibilidad que se muestra en la Figura 1. Cada fila en la cuadrícula de la Figura 1 muestra la disponibilidad de la pista para diferentes *slots* de tiempo. Las celdas en verde indican que ese *slot* de tiempo está libre y puede ser asignado para el aterrizaje de un avión, mientras que las celdas en negro indican que ese *slot* ya ha sido asignado.

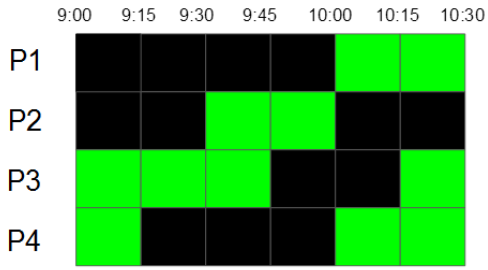


Figura 1: Disponibilidad de las pistas de aterrizaje del aeropuerto. Cada celda en la cuadrícula indica un *slot* de tiempo que puede estar disponible (en verde) u ocupado por el aterrizaje de un avión (en negro).

La compañía nos informa que van a aterrizar sus 5 aviones en el aeropuerto. La horas programadas de llegada de estos aviones, junto con la hora máxima de aterrizaje, se muestran en la Tabla 3. Para que estos aviones aterricen se les debe asignar una pista y un *slot* de tiempo cuya hora de inicio sea igual o posterior a la hora programa de aterrizaje del avión, e igual o anterior a la hora límite de aterrizaje. De esta forma, el avión AV1 con llegada a las 9:10, no puede estar asignado a la pista P3 y el *slot* de las 9:00. Sí podría asignarse al *slot* de las 10:15 pero eso implicaría que los pasajeros deben alargar su vuelo 65 minutos más antes de poder aterrizar. En cambio, el avión AV2 podría asignarse al *slot* de las 9:00 de la pista P4, pero no a *slots* posteriores a las 9:30, puesto que el avión se quedaría sin combustible. Por último, la compañía nos informa de que cada minuto de retraso en la hora programada de aterrizaje de un avión le supone un coste adicional, puesto que se requiere quemar más combustible. Este coste adicional por minuto está reflejado en la última columna de la Tabla 3.

Por tanto, se debe realizar la asignación de los aviones a *slots* de tiempo de aterrizaje minimizando el coste adicional por los retrasos, y teniendo en cuenta que:

- Todos los aviones tienen que tener asignado un *slot* de tiempo para efectuar el aterrizaje.
- Un *slot* de tiempo puede estar asignado como máximo a un avión.
- El *slot* que se asigna a un avión debe ser un *slot* libre (en verde en la Figura 1).
- El inicio del *slot* de aterrizaje debe ser igual o posterior a la hora de llegada del avión.

Avión	Hora programada	Hora límite	Coste
AV1	9:10	10:15	100€
AV2	8:55	9:30	200€
AV3	9:40	10:00	150€
AV4	9:55	10:15	250€
AV5	10:10	10:30	200€

Tabla 3: Tabla con la hora de llegada de los vuelos.

- El inicio del *slot* de aterrizaje debe ser igual o anterior a la hora límite de aterrizaje del avión.
- Por cuestiones de seguridad, no se pueden asignar dos *slots* consecutivos en la misma pista.

Se pide:

1. Modelar el problema como un problema de programación lineal entera.
2. Implementar el modelo de la parte 1 de la práctica en un lenguaje de modelado más sofisticado, MathProg. El modelo de la parte 1 es más sencillo, lo que permitirá practicar con la sintaxis de este lenguaje antes de abordar la implementación del modelo de la parte 2.
3. Una vez implementado el modelo de la parte 1, se harán las modificaciones necesarias a este modelo para incluir el problema de optimización descrito en la parte 2. Para ello, se aconseja primero implementar el modelo de la parte 2 de forma separada, y después estudiar la forma de combinar ambos modelos, puesto que ambos se refieren a optimizar los beneficios/costes de la compañía. Al final, se obtendrá un único modelo MathProg que contemple la optimización tanto de la parte 1 como de la parte 2.

2.3. Parte 3: Análisis de Resultados

En este apartado se deben analizar todos los resultados obtenidos, describir la solución obtenida (comprobando que cumple con las restricciones de este enunciado) y analizar qué restricciones están limitando la solución del problema.

Análisis de la complejidad del problema: ¿cuántas variables y restricciones has definido?, modifica el problema, variando los parámetros o añadiendo/eliminando aviones/pistas y explica cómo estas modificaciones afectan a la dificultad de resolución del problema resultante.

Considerando la segunda parte de la práctica, otro de los problemas que pueden surgir es que haya retrasos en las llegadas de los vuelos, lo que puede requerir replanificar la asignación de aviones a pistas de despegue. Si tuvieras que considerar los posibles retrasos de los aviones en el modelo, ¿cómo lo modelarías? Considera ahora que el avión AV1 tiene un retraso de 20 minutos, ¿se encuentra solución? y, si es así, ¿cuál es la nueva asignación de aviones a pistas y *slots* de tiempo?

Por último, discute las ventajas y desventajas de las dos herramientas utilizadas en la asignatura: LibreOffice y GLPK.

3. Directrices para la Memoria

La memoria debe entregarse en formato .pdf y tener un máximo de 15 hojas en total, incluyendo la portada, contraportada e índice. Al menos, ha de contener:

1. Breve introducción explicando los contenidos del documento.
2. Descripción de los modelos, argumentando las decisiones tomadas.
3. Análisis de los resultados.
4. Conclusiones acerca de la práctica.

La memoria **no debe incluir código fuente** en ningún caso.

4. Evaluación

La evaluación de la práctica se realizará sobre 10 puntos. Para que la práctica sea evaluada deberá realizarse al menos el apartado 1 y la memoria. La distribución de puntos es la siguiente:

1. Parte 1 (3 puntos)
    - Modelización del problema (1 punto)
    - Implementación del modelo (2 puntos)
2. Parte 2 (5 puntos)
    - Modelización del problema (3 puntos)
    - Implementación del modelo (2 puntos)
3. Parte 3 (2 puntos)

En la evaluación de la modelización del problema, un modelo correcto supondrá la mitad de los puntos. Para obtenerse el resto de puntos, la modelización del problema deberá:

- Ser formalizada correctamente en la memoria.
- Ser, preferiblemente, sencilla y concisa.
- Estar bien explicada (ha de quedar clara cuál es la utilidad de cada variable/restricción).
- Justificarse en la memoria todas las decisiones de diseño tomadas.

En la evaluación de la implementación del modelo, un modelo correcto supondrá la mitad de los puntos. Para obtenerse el resto de puntos, la implementación del problema deberá:

- Hacer uso (en la implementación) de las capacidades que ofrecen las herramientas para que hacer/actualizar el modelo sea lo más sencillo posible (por ejemplo, utilizar `sumaproducto` si es posible en el caso de la hoja de cálculo o el uso de `sets` en MathProg).
- Mantener el código (hoja de cálculo o ficheros de MathProg) correctamente organizado y comentado. Los nombres deben ser descriptivos. Deberán añadirse comentarios en los casos en que sea necesario para comprenderlo.

Al puntuar el análisis de resultados, se valorará positivamente el hecho de incluir en la memoria conclusiones personales acerca de la dificultad de la práctica y de lo aprendido durante su elaboración.

Importante: los modelos implementados en la hoja de cálculo y GLPK deben ser correctos. Esto es, han de funcionar y obtener soluciones óptimas al problema que se solicita. En ningún caso se obtendrá una calificación superior a 1 punto por un modelo que no es correcto. Por tanto, si la parte 1 no está correctamente acabada, la nota máxima será de 1 punto y, si la parte 2 no está correctamente acabada, como mucho se obtendrá una calificación de 4 puntos.

5. Entrega

Se tiene de plazo para entregar la práctica hasta el 8 de Noviembre a las 23:55. Este límite es fijo y no se extenderá de ningún modo.

Sólo un miembro de cada pareja debe subir los siguientes ficheros:

- Un único fichero `.zip` a través del punto de entrega de 'Aula Global' llamado "*Primera práctica*".

  El fichero debe nombrarse `p1-NIA1-NIA2.zip`, donde `NIA1` y `NIA2` son los últimos 6 dígitos del NIA (rellenando con `0`s por la izquierda si fuera preciso) de cada miembro de la pareja.

  Ejemplo: `p1-054000-671342.zip`.

- La memoria debe subirse separadamente a través del punto de entrega Turnitin de 'Aula Global' llamado "*Primera práctica (sólo pdf)*".

  La memoria debe llamarse `NIA1-NIA2.pdf` —después de sustituir adecuadamente los NIAs de cada estudiante como en el caso anterior.

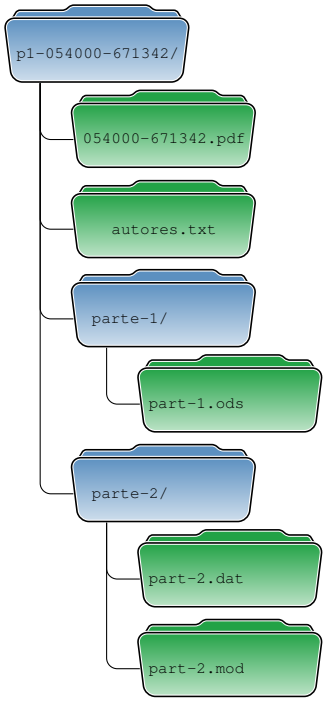
  Ejemplo: `054000-671342.pdf`.

La descompresión del fichero `.zip` descrito en el primer punto debe producir un directorio llamado `p1-NIA1-NIA2`, donde `NIA1` y `NIA2` son los últimos 6 dígitos del NIA de cada estudiante rellenando con `0`s si fuera preciso. Este directorio debe contener: primero, la misma memoria entregada a través del segundo enlace descrito anteriormente que debe nombrarse como `NIA1-NIA2.pdf` —después de sustituir convenientemente los NIAs de cada estudiante; segundo, un fichero llamado `autores.txt` que identifica a los miembros de cada pareja, con una línea por cada uno con el siguiente formato: `NIA Surname, Name`. Por ejemplo:

```
054000 Von Neumann, John
671342 Turing, Alan
```

Además, este directorio debe contener un directorio por cada parte realizada llamados "`parte-1`" y "`parte-2`". Las soluciones de cada parte (tanto el código fuente como la hoja de cálculo generada) deben incluirse en sus respectivos directorios.

La siguiente figura muestra una distribución posible de los ficheros después de descomprimir el fichero `.zip`:



Importante: no seguir las normas de entrega puede suponer una pérdida de hasta 1 punto en la calificación final de la práctica.