Práctica: *Programación Lineal.*
Heurística y Optimización
Grado de Ingeniería en Informática. Curso 2019-2020.
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

Un importante museo 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

El museo para el que ahora trabajamos nos explica que está en pleno proceso de renovación y quiere adquirir nuevo equipamiento en el mercado para mejorar sus instalaciones. Uno de los principales motivos de preocupación del museo es la gestión de sus entradas. En este sentido, el museo nos describe que dispone de tres entradas: la principal situada en el lado este, y dos secundarias situadas en el lado norte y oeste respectivamente (Figura 1 (a)).

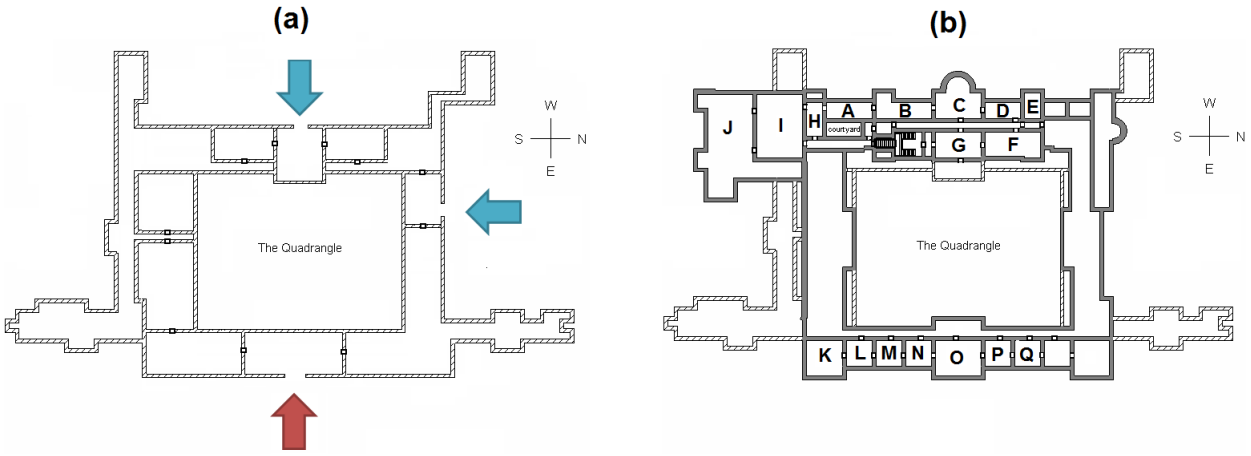


Figura 1: (a) Vista de la planta baja y cada una de las entradas y (b) vista de la primera planta del museo.

Actualmente, cada entrada es atendida por una única persona encargada de la venta de los *tickets*, la descripción de las instalaciones, y del control de acceso. Sin embargo, estas tres personas son insuficientes para atender el volumen de visitas, y cada día se forman enormes colas de acceso que hacen que muchos visitantes abandonen la espera o prefieran visitar otros lugares. En este sentido, el museo nos proporciona el tiempo de espera que actualmente emplea cada visitante para acceder al museo en cada entrada (Tabla 1). Además, nos informa que cada una de estas entradas permanecen abiertas durante 8 horas al día.

Para solucionar este problema, el museo desea adquirir máquinas expendedoras de *tickets*, tornos de acceso, y contratar más personal para agilizar la entrada de visitantes al museo. Cada uno de estos elementos permiten

	Entrada Este	Entrada Oeste	Entrada Norte
Tiempo espera/visitante (min.)	130	100	90

Tabla 1: Tiempo de espera de cada visitante en cada entrada.

reducir la espera de cada visitante para entrar en el museo de forma diferente. En la tabla 2, en la primera fila, se muestra esta reducción en minutos por visitante. Aplicando estas reducciones, tendríamos que si se instala una máquina expendedora en la entrada principal, el tiempo de espera por visitante pasaría de 130 a 128 minutos. En cambio, si se instalan dos máquinas expendedoras y se contrata una nueva persona para esta entrada, el tiempo de espera por visitante pasaría de 130 a 122 minutos. En la segunda fila de la tabla 2 se muestra el coste por hora de servicio de cada uno de los elementos.

	Expendedora	Torno	Persona
Reducción espera (min./visitante)	2	3	4
Coste (€/hora)	4	5	8

Tabla 2: Información sobre la reducción del tiempo y el coste de las máquinas expendedoras, tornos, y personas que desea situar el museo en cada entrada.

El museo nos impone algunas restricciones a la hora de solucionar el problema:

- En primer lugar, el gasto total generado por las máquinas expendedoras, tornos y personas que el museo desea realizar al día para todas sus entradas no debe superar los 1000€.
- Además, la entrada principal no debe superar en más del 10 % el gasto destinado a cada una de las otras dos entradas.
- Por otro lado, la suma del número de máquinas expendedoras y tornos de cada entrada secundaria debe ser menor que la misma suma de la entrada principal, y el número de tornos de la entrada norte debe ser menor que en la entrada oeste.
- Además, como mínimo, debe haber dos máquinas expendedoras, dos tornos y dos personas en la entrada principal y una máquina expendedora, un torno y una persona en cada una de las entradas secundarias.
- Por último, se espera que la reducción de tiempo de espera por visitante en la entrada principal sea mayor que en cada una de las otras dos entradas, puesto que es donde se reciben más visitantes.

En esta parte, el objetivo es determinar cuál es el número de máquinas expendedoras, tornos y nuevo personal que el museo debe situar en cada entrada para minimizar el tiempo medio de espera de los visitantes considerando todas las entradas, y teniendo en cuenta las restricciones impuestas.

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

El museo está muy satisfecho con los resultados que hemos obtenido en la primera parte, y ahora nos plantea otro problema que desea resolver. En particular, el museo nos cuenta que ha adquirido 8 robots (Figura 2).

Estos robots son capaces de presentar cada una de las 17 salas que el museo tiene en la primera planta distribuidas en dos alas, la oeste y la este, y que están nombradas con una letra (Figura 1 (b)). Las especificaciones de cada uno de los robots adquiridos por el museo se detallan en la tabla 3.

En la primera fila de la tabla 3 se describe el tiempo que tarda cada robot en presentar al visitante un objeto de la sala en la que se encuentra. Se puede ver, en este caso, que el robot $R5$ es el más rápido en presentar

Figura 2: Ejemplo de robot adquirido por el museo.

	R1	R2	R3	R4	R5	R6	R7	R8
Presentación (min./objeto)	4	6	5	3	2	3	4	5
Energía (unid./objeto)	7	5	3	1	2	4	4	5
Límite Energía (unid.)	100	90	95	40	45	75	85	60

Tabla 3: Información sobre cada uno de los robots adquiridos por el museo.

cada uno de estos objetos, y el $R2$ el más lento. En la segunda fila de la tabla 3 se describe las unidades de energía que consume el robot en presentar un objeto de la sala, y por último, la última fila describe el límite de energía que el robot puede consumir antes de quedarse sin batería. El museo desea que nos encarguemos de la asignación de estos robots a cada una de las salas del museo.

La tabla 4 muestra el número de objetos que contienen cada una de estas salas. Así, se puede ver que las salas A, B y C contienen 5 objetos cada una, las salas D y E contienen 6 objetos cada una, y así sucesivamente. Una sala se considerará presentada si el robot que tiene asignado ha presentando todos los objetos de la misma.

	A, B, C	D, E	F, G, H	I, J	K, L, M	N, O, P, Q
Objetos	5	6	4	7	3	2

Tabla 4: Número de objetos que hay en cada sala del museo.

Para realizar la asignación de robots a salas, debemos tener en cuenta las siguientes restricciones. En primer lugar, no puede haber más de un robot en una sala, y todas las salas deben tener asignado un robot. Además, un robot no puede estar asignado a menos de dos salas y a más de tres. El museo nos informa, además, que los robots $R3$, $R5$ y $R6$ no pueden ser asignados a salas que pertenezcan al ala oeste y los robots $R2$ y $R4$ no pueden ser asignados a salas del ala este. Por otro lado, sólo los robots que están asignados a las salas A, B o ambas pueden estar asignados a las salas C, D o ambas. En cuanto al consumo de energía, un robot no puede estar asignado a salas cuyas presentaciones requieran una energía superior al límite que puede consumir antes de quedarse sin batería. Por último, puesto que las salas del ala oeste son más grandes, se requiere que el tiempo de las presentaciones de las salas en este ala, sea un 10 % mayor que el tiempo de las presentaciones de las salas en el ala este.

El objetivo en esta parte es minimizar el tiempo de espera medio de los visitantes para acceder al museo mediante la compra de máquinas expendedoras, tornos, y personal, y el tiempo de presentación de todas las salas del museo. Para ello, hay que tener en cuenta las restricciones impuestas por el museo.

Se pide:

1. Modelar el problema como un problema de programación lineal entera.
2. Implementar el modelo en un lenguaje de modelado más sofisticado (MathProg).

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 robots y explica cómo esto afecta a la dificultad de resolución del problema resultante.

Además, se deben contestar la siguientes cuestiones: ¿cuál es el tiempo de espera de acceso de un visitante que entre por la entrada principal? ¿y por cada una de las entradas secundarias? ¿cuánto tiempo le llevará visitar todas las salas del museo? ¿qué robots se encargarán de hacerle la presentación en cada una de estas salas?

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

2.4. Parte extra: Problema de Programación Dinámica

Dado el éxito que ha supuesto el modelo propuesto para el museo, al cabo de unos días se pone en contacto con nosotros nuevamente para solucionar otro tipo de problema: el reparto de algunas de sus obras de arte a otros museos que realizan exposiciones temporales. Para hacer el reparto, el museo cuenta con un único camión, el cual tiene capacidad para hacer frente a todos los pedidos recibidos.

Antes de empezar con el reparto, se debe trazar un itinerario. El objetivo es definir una ruta de reparto de tal modo que comience en el museo de origen, visite todos los destinos una única vez y vuelva al punto de partida. Asumiendo que el reparto se puede realizar en una jornada laboral sin repostar, se pide: realizar un programa que decida el itinerario utilizando programación dinámica para minimizar la longitud de la ruta. El programa recibirá un fichero de texto plano que contendrá la matriz de distancias entre todos los destinos, incluyendo el museo de origen, el cual se identificará mediante la cadena "MO". A continuación se muestra un ejemplo de fichero de entrada:

	MO	M1	M2	M3	M4
MO	0	24	16	23	24
M1	16	0	27	14	11
M2	15	14	0	16	27
M3	14	22	19	0	23
M4	13	11	26	18	0

Para realizar pruebas, se recomienda generar ficheros aleatoriamente con diferente número de museos de destino, asumiendo distribuciones uniformes, entre 10 y 30 kilómetros para las distancias. Se debe tener en cuenta que la distancia de una localización a sí misma debe ser cero, por lo que la diagonal principal de la matriz debe ser 0.

Deberá entregarse el programa en el lenguaje que se desee, junto a un script en bash que lo ejecute: `itinerario.sh`. Dicho script recibirá como argumento el fichero de distancias e imprimirá por pantalla 2 líneas: la primera indicando la distancia total de la ruta y la segunda el itinerario generado. A continuación se muestra un ejemplo de fichero de salida:

```
Distancia total:                78
Itinerario:     MO, M2, M3, M1, M4, MO
```

Como se puede ver las rutas comienzan y finalizan en el museo de origen, pasando una única vez por cada museo. En la memoria deberán documentarse las ecuaciones de programación dinámica empleadas, así como posibles optimizaciones adicionales que ayuden a que el programa sea más eficiente. Además, deben contestarse las siguientes preguntas:

- Si se deben realizar repartos a 100 museos diferentes, ¿cuál es el número de combinaciones que haría un algoritmo de fuerza bruta?. ¿cuál es el número de combinaciones que considera tu solución?
- ¿Sería más eficiente la ejecución del algoritmo si los museos estuvieran ordenadas en función de algún criterio?

- Imagina ahora que en vez de querer calcular la ruta más corta, es decir, una ruta cuya longitud fuese la mínima posible, se quisiera obtener todas las rutas de longitud mínima. ¿Sería posible mediante programación dinámica? Justifica tu respuesta

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)
4. Parte Extra: Programación Dinámica (1 punto)

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 27 de Octubre a las 23:55. El deadline es firme y no se extenderá.

Sólo un miembro de cada pareja de estudiantes debe subir:

- Un único fichero `.zip` a la sección de prácticas de Aula Global denominado "*Entrega Práctica 1*".

  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 en formato pdf debe entregarse a través del enlace Turnitin denominado "*Entrega Memoria Práctica 1*".

  La memoria debe entregarse en formato pdf y debe llamarse `NIA1-NIA2.pdf` —después de sustituir convenientemente el NIA de cada estudiante.

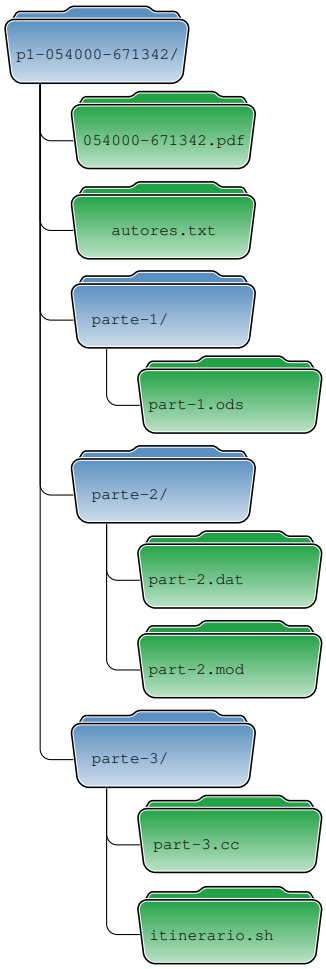
  Ejemplo: `054000-671342.pdf`.

La descompresión del fichero entregado en primer lugar debe generar un directorio llamado `p1-NIA1-NIA2`, 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. Este directorio debe contener: primero, la misma memoria en formato pdf que ha sido entregada a través de Turnitin, y debe llamarse `NIA1-NIA2.pdf` —después de sustituir convenientemente el NIA de cada estudiante; segundo, un fichero llamado `autores.txt` que identifique a cada autor de la práctica en cada línea con el formato: `NIA Apellidos, Nombre`. Por ejemplo:

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

Además, este directorio debe producir al menos dos directorios llamados exactamente "`parte-1`" y "`parte-2`". Si se entregara la parte extra (de Programación Dinámica), entonces debe incluirse un tercer directorio "`parte-3`" que la contenga.

Se muestra a continuación una distribución posible de los ficheros que resultan de la descompresión:



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