uc3m | Universidad Carlos III de Madrid

MEMORIA PRÁCTICA 1

Heurística y optimización.

Descripción breve

Enfoque práctico a la realización de tareas de programación lineal con las herramientas solucionador de LibreOffice y Mathprog sobre la optimización de la entrada y exposiciones de salas de un museo.

Daniel Zubieta Pascual 100346138

100346138@alumnos.uc3m.es

Índice

Introducción

El objetivo de la práctica uno de Heurística y Optimización es la de aprender a modelar e implementar la tarea de programación lineal de ayudar a un museo a colocar controles de entrada en las entradas para mejorar el tiempo de espera y la mejora de tiempo de exposición de las salas usando unos robots, tanto en una hoja de cálculo de LibreOffice con su solucionador lineal como con un lenguaje más sofisticado como es MathProg.

El propósito de este documento es la de explicar de forma concisa y clara el modelo que hemos llevado a cabo para la implementación de la solución y entrar en profundidad en las principales decisiones tomadas a lo largo de toda la modelización.

El documento se dividirá en dos partes, la primera en explicar dicha modelización e implementación en una hoja de cálculo de LibreOffice y su posterior análisis de datos.

En la segunda parte se explicará el nuevo modelado de la segunda parte con las restricciones nuevas del museo para los agentes nuevos agregados, además del correspondiente análisis de datos de esta nueva parte. No se explicará en este caso la modelización de la parte 1 porque las restricciones y variables usadas en este caso son las mismas, solo que puestas en MathProg.

Por último, se encontrará unas conclusiones respecto a la práctica y las impresiones personales del uso de ambas herramientas de resolución de tareas de programación lineal.

Modelización parte 1

La primera parte consiste en resolver una tarea de programación linear con el objetivo de reducir el tiempo de espera en las tres entradas del museo, la entrada principal Este y las entradas secundarias Norte y Oeste. Para ello, el museo nos impone unas restricciones que deberán quedar reflejadas en nuestro modelo de la hoja de cálculo, y en base a ellas, asignar un número de máquinas expendedoras, tornos y personal a cada una de las entradas del museo.

Para ello, nos harán falta variables. En este caso hemos definido un total de 9 variables, que se comprenderán para definir el número de elementos asignados a cada puerta. En la hoja de cálculo, se ha representado usando una matriz de 3 filas y 3 columnas, además de contener en la hoja los distintos parámetros dados en el enunciado. Las variables se han nombrado bajo la nomenclatura X_{ij}, donde "j" se refieren a los distintos recursos que podremos colocar (máquinas expendedoras, tornos y personas) y la "i" se refiere a las entradas (Este, Norte, Oeste).

Todos los parámetros usados son los dados por el enunciado y estarán representados en la hoja de cálculo bajo la sección de Datos. La reducción de espera y los costes por recursos serán usados en las restricciones, especificando en cada una de ellas con una letra para diferenciarlas.

Para las restricciones impuestas por el museo, hemos modelizado:

- La suma del gasto máximo de todas las entradas no podrá ser mayor de 1000 euros. Aquí entra en juego la variable G_i que indicará el coste de cada recurso.
  - $\sum_{i=1}^{3}(\sum_{j=1}^{3} X_{ij}G_j) \leq 1000$

- El gasto de la entrada Este no debe superar en un diez por ciento el gasto de las entradas secundarias. En esta restricción modelaremos dos restricciones, una por cada entrada. Cabe destacar que no hemos introducido ninguna constante en esta restricción.
  - $\sum_{j=1}^{3} X_{1j}G_j \leq (\sum_{j=1}^{3} X_{2j}G_j)1.1$
  - $\sum_{j=1}^{3} X_{1j}G_j \leq (\sum_{j=1}^{3} X_{3j}G_j)1.1$

- La suma del número de máquinas expendedoras y de tornos de las entradas secundarias no debe ser superior a la misma suma en la entrada principal (este). Esta restricción también ha sido dividida en dos, una para cada entrada.
  - $\sum_{j=1}^{2} X_{2j} G_j \leq (\sum_{j=1}^{2} X_{1j} G_j) - 1$
  - $\sum_{j=1}^{2} X_{3j} G_j \leq (\sum_{j=1}^{2} X_{1j} G_j) - 1$

- El número de tornos en la entrada Norte debe ser menor que el número de tornos en la entrada Oeste. En esta restricción, al ser una única línea, pondremos directamente la ecuación.
  - $X_{32} \leq X_{22} - 1$

- Como mínimo, debe haber dos máquinas expendedoras, dos tornos y dos personas en la entrada principal; 1 máquina expendedora, 1 torno y 1 persona en cada una de las entradas secundarias.
  - $\sum_{j=1}^{3} X_{1j} \geq 2$
  - $\sum_{j=1}^{3} X_{2j} \geq 1$
  - $\sum_{j=1}^{3} X_{3j} \geq 1$

- La reducción de tiempo en la entrada principal debe ser mayor que en cada una de las secundarias. Se define la variable R_j, para representar la reducción por tiempo de espera de cada recurso representado por "j", para recorrer las máquinas expendedoras, tornos y personas.
  - $\sum_{j=1}^{3} X_{1j} R_j \leq \sum_{j=1}^{3} X_{2j} R_j$
  - $\sum_{j=1}^{3} X_{1j} R_j \leq \sum_{j=1}^{3} X_{3j} R_j$

- Por último, cabe destacar una restricción implícita en el problema, que es que todas las variables sean no negativas.
  - $X_{ij} \geq 0$

En la modelización en MathProg para la parte dos hemos añadido tres restricciones al modelo auxiliares (comentadas en el código) para evitar fallos en la ejecución que nos tomasen valores superiores al tiempo de espera predefinido.

Ya solo falta definir la función objetivo para lograr la correcta resolución de la tarea de programación lineal. Dicho objetivo es lograr minimizar el tiempo de espera medio en las puertas del museo. En la hoja de cálculo de LibreOffice, se ha optado por separar la función objetivo en tres ecuaciones. La primera ecuación se encargará de restar el tiempo de espera inicial a la reducción de tiempo por el número de recursos metidos en la puerta principal; la segunda ecuación realizará lo mismo para la puerta secundaria 1; y por último la tercera ecuación realizará lo mismo para la puerta secundaria 2. La función objetivo es la suma de estas tres ecuaciones y luego la división entre 3, por ser el tiempo medio. Esta separación se debe a una simplificación a la hora de realizar la hoja y colocar las variables. En la modelización, la variable EC_i corresponde a la reducción de cada ecuación por separado, donde la "i" corresponde a cada entrada del museo.

$$\min z = \frac{1}{3}\left(\sum_{i=1}^{3} EC_i\right)$$

Para la realización de la parte 2, al pedirnos que también diseñemos la solución en el lenguaje MathProg, se han copiado exactamente las mismas restricciones, por lo que así nos lo ahorramos en etapas posteriores de esta memoria.

Análisis de resultados parte 1

La solución óptima según el solucionador es de 84,33, que corresponde con la minimización del tiempo de espera en las tres puertas del museo. Las soluciones finales del solucionador del LibreOffice son las siguientes.

	Máquinas	Tornos	Personas
Entrada este	3	3	2
Entrada norte	2	2	3
Entrada oeste	1	4	2

Para lograr la solución óptima a la primera parte, se deben colocar exactamente ese número de recursos en esas entradas. Según los valores en las restricciones, parece que la restricción del gasto total es la que más penaliza al modelo, porque se ajusta al presupuesto máximo que tiene el museo.

En esta parte, se han definido 9 variables y 17 restricciones. Es problemático porque cuantas más restricciones tiene una tarea, más tiempo tardará en hallar la solución y en este problema que haya 17 restricciones, con otro tipo de datos, puede llegar a generar una solución que no sea factible.

En este caso es rápido al ser un problema pequeño, pero una tarea más compleja no sería buena idea resolverlo con LibreOffice. En cuanto al tiempo de espera de los visitantes, se ha logrado reducir en la entrada principal Este de 130 minutos a 107; en la entrada secundaria Norte de 90 a 68; y en la entrada secundaria Oeste de 100 a 78. 22 minutos reducidos en las puertas secundarias y 23 en la principal, una buena cantidad de tiempo reducida gracias a la adición de esos componentes a cada entrada.

Respecto a la herramienta LibreOffice, queremos destacar como ventajas el fácil uso de ella, así como la interfaz gráfica de la que dispone para resolver la tarea. El solucionador es sencillo de usar, y al poder ir restricción a restricción podemos ver que restricciones causan contradicciones entre ellas para poder corregirlas.

Sin embargo, encontramos la herramienta deficiente en cuanto a la definición de restricciones. El solucionador no admite los operadores < y >, lo que añaden más operandos a las restricciones; hay que definir cuidadosamente cada una de ellas, lo que hace que se generen muchas restricciones que, si es una tarea con muchas de ellas, tardaría en encontrar solución.

Modelización parte 2

En esta segunda parte, además de modelar la tarea de colocar los recursos de la mejor manera posible y reducir el tiempo de espera para entrar, debemos ser capaces de encontrar la solución óptima para la tarea de programación lineal de encontrar una disposición de los robots en las salas del museo con ayuda de robots para minimizar el tiempo de presentaciones de las salas.

El museo tiene ocho robots, y deben ser capaces de presentar cada una de las diecisiete salas que el museo tiene. Cada robot tendrá su propia especificación que nos ayudará a obtener la solución final.

Primero definiremos el espacio de la tarea. La variable de decisión principal será una matriz binaria para exponer qué robot está en qué sala, representadas con un 1 en caso afirmativo. El museo nos

asigna que tienen un total de 17 salas (nombradas de la "A" a la "Q") y 8 robots (R1 a R8) con sus características propias. La variable de decisión será R_{ij}, donde "i" representan las salas y "j" los robots.

Para las nuevas restricciones modelamos:

- Debe haber un robot en cada sala y cada sala debe tener un robot. La segunda parte se cumple al cumplirse la primera.
  - $\sum_{i=1}^{17}(\sum_{j=1}^{8} R_{ij}) = 1$

- Los robots deben tener como mínimo dos salas asignadas.
  - $\sum_{j=1}^{8}(\sum_{i=1}^{17} R_{ij}) \geq 2$

- Los robots deben tener como máximo tres salas asignadas.
  - $\sum_{j=1}^{8}(\sum_{i=1}^{17} R_{ij}) \leq 3$

- Los robots asignados al ala oeste no pueden estar en el ala este. Esto es, los robots R2 y R4 no pueden estar en el ala este. Las salas del este son las numeradas de la 11 a la 17 (K-Q).
  - $(\sum_{i=11}^{17} R_{i2}) = 0$
  - $(\sum_{i=11}^{17} R_{i4}) = 0$

- Los robots asignados al ala este no pueden estar en el ala oeste. Esto es, los robots R3, R5 y R6 no pueden estar en el ala oeste. Las salas del oeste son las numeradas de la 1 a la 10 (A-J).
  - $\sum_{i=1}^{10} R_{i3} = 0$
  - $\sum_{i=1}^{10} R_{i5} = 0$
  - $\sum_{i=1}^{10} R_{i6} = 0$

- Los robots asignados a las salas A y B o ambas pueden ir a las salas C y D. En esta restricción la haremos de una línea al acceder a un valor en concreto.
  - $R_{i1} + R_{i2} = R_{i3} + R_{i4}$

- Los robots no pueden presentar salas cuyas prestaciones sean mayores que la energía límite de los robots. Lo que consumen los objetos de la sala se define como O y lo que tarda la prestación con P. El límite se define como L de cada robot.
  - $\sum_{i=1}^{17}(\sum_{j=1}^{8} R_{ij} P_i O_i) \leq \sum_{j=1}^{8} L_i$

- El tiempo de las presentaciones del ala oeste debe ser mayor en un diez por ciento al tiempo de presentaciones del ala este.

- $\sum_{i=1}^{17}(\sum_{j=1}^{8} R_{ij}P_iO_i)\ 1.1 \leq \sum_{i=1}^{10}(\sum_{j=1}^{8} R_{ij}P_iO_i)$

Ya solo falta definir la función objetivo de la segunda parte y la función objetivo para resolver la tarea. Así pues, la función objetivo será la suma de los productos de cada robot con los objetos de las salas asignadas y lo que tarda el robot en presentarla. El robot de la sala estará asignado como R_{ij}, los objetos como O_i y la presentación de cada robot como P_j, siendo de nuevo "i" las salas y "j" los robots.

$$\min z = \sum_{i=1}^{17}(\sum_{j=1}^{8} R_{ij}O_iP_i))$$

Añadiendo la función objetivo de la parte 1, la función objetivo del modelo final será la siguiente.

$$\min z = \sum_{i=1}^{17}(\sum_{j=1}^{8} R_{ij}O_iP_i)) + \frac{1}{3}(\sum_{k=1}^{3} EC_k)$$

Análisis de resultados parte 2

La diferencia que vemos según la solución de LibreOffice es que en nuestro caso el número de máquinas, tornos y personal difiere en las entradas. Sin embargo, al introducir estos números en la hoja de cálculo podemos ver que las restricciones impuestas en la parte 1 se siguen cumpliendo y la función objetivo no cambia. Es interesante ver como diferentes solucionadores dan diferentes resultados, pero mantienen las restricciones cumpliéndose la función objetivo.

Respecto a la resolución de la parte 2, el fichero de salida muestra donde están los robots en la solución óptima. Lo expondremos en la siguiente tabla.

Salas	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
Robot	1	7	7	1	4	2	2	8	4	4	5	5	6	3	8	6	3

Al ver el fichero de salida, podemos advertir claramente que todas las restricciones se cumplen. El robot que está en A está también en B, los robots se mantienen en el ala donde deben estar y ningún robot está en menos de 2 salas y más de 3. Las restricciones más restrictivas, valga la redundancia, son que los robots tengan una sala

asignada, y las que definen que no se pase de unos límites establecidos de sala.

En esta tarea se han definido 8 restricciones, por lo que es un avance respecto de la solución LibreOffice y una matriz bidimensional de 17 columnas por 8 filas.

Añadir robots hace que algunas restricciones puedan no cumplirse, incluso hasta encontrar una solución que no sea factible, como por ejemplo que un robot no cumpla las restricciones de las salas este y oeste y que se comprenda entre el número mínimo y máximo de salas a las que deben ser asignados.

Conclusiones

En primer lugar, me ha parecido una práctica muy interesante porque expone un problema de la vida real que se puede resolver con dos herramientas, una de más fácil uso como la hoja de cálculo de LibreOffice y otra de más difícil comprensión como MathProg. Sin embargo, el principal problema a la hora de afrontar la resolución de estas tareas es la peculiaridad del lenguaje MathProg, que sin duda con alguna ayuda más de la dada creo que podría ser más de ayuda a los que más problemas puedan tener con su compresión, aunque una vez entendido es fácil modelar todo.

Entrando en la realización de la práctica, la parte de LibreOffice ha sido sencilla de realizar porque el lenguaje tiene un enfoque puramente matemático que es muy familiar y al ver claramente que restricciones no se cumplen se puede corregir. Es una herramienta además fácil de usar y que

Respecto a MathProg, modelizar los problemas es más sencillo al poder escribir menos restricciones porque se agrupan entre ellas al usar los mismos datos. Sin embargo, una restricción mal escrita o un diseño mal enfocado no solo produce errores, sino que es imposible saber en qué te has equivocado al no salir mensaje de error más allá de que no ha encontrado solución óptima.