Universidad Carlos III

Heurística y Optimización

2023-24

Práctica 1

Ingeniería Informática, Tercer Curso

Adrián Fernández Galán (NIA: 100472182, e-mail: 100472182@alumnos.uc3m.es)

César López Mantecón (NIA: 100472092, e-mail: 100472092@alumnos.uc3m.es)

Prof. María Fernanda Salerno Garmendia

Grupo: 81

Indice

1. Introducción

En este documento se recoge el desarrollo de la práctica 1 de Heurística y Optimización. La práctica consiste en resolver un problema de programación lineal a través de hojas de cálculo y el lenguaje de programación MathProg.

El problema describe una serie de restricciones para asignar llamadas a distintos parkings. La primera parte trata con un número fijo de parkings y llamadas. Mientras que la segunda parte añade la posibilidad de construir nuevos parkings.

A continuación se incluye la descripción del modelo, el análisis de los resultados y las conclusiones de la práctica.

2. Descripción del modelo

2.1. Modelo Parte 1

El problema trata de minimizar el tiempo en atender todas las llamadas de 5 distritos con 3 parkings.

El modelo consta de 15 variables (3 parkings, multiplicados por 5 distritos), representando el número de llamadas que se asignan a cada parking desde cada distrito.

El problema cuenta con 4 restricciones:

- Restricción sobre el número máximo de llamadas atendidas por un parking: un parking no puede atender más de 10.000 llamadas.
- Restricción sobre el número total de llamadas atendidas: se deben atender absolutamente todas las llamadas.
- Restricción sobre el tiempo máximo en atender una llamada: un parking no puede atender llamadas de distritos a los que tarde más de 35 minutos.
- Restricción de balance: un parking no puede atender más del 150% de las llamadas de otro parking.

Con esto, la descripción algebraica del modelo queda de la siguiente forma:

Modelo algebráico

- Constantes del Problema:
  C_{ij}: =Tiempo que tarda una ambulancia del distrito *j* al parking *i*.
  L_j: =Número total de llamadas de cada distrito *j*.
  T_{max}: =Tiempo máximo que puede tardar un parking en atender una llamada.

F_B: =Factor de balanceo entre parkings.

P_i: =Máximo número de llamadas que puede atender el parking *i*.

- Variables de decisión:
  l_{ij}: =Número de llamadas que atiende el parking *i* del distrito *j*.

- Función objetivo: Sean P el conjunto de parkings y D el conjunto de distritos.

$$\min\ t\ = \sum_{j \in D} \sum_{i \in P} C_{ij} \cdot l_{ij}$$

- Restricciones: Sean P el conjunto de parkings y D el conjunto de distritos.

$$\sum_{j \in D} l_{ij} \leq P_i,\ \forall i \in P$$

$$\sum_{i \in P} l_{ij} = L_j,\ \forall j \in D$$

$$l_{ij}(T_{max} - C_{ij}) \geq 0,\ \forall i \in P, \forall j \in D$$

$$\sum_{j \in D} l_{ij} \leq F_B \sum_{j \in D} l_{kj},\ \forall i,\ k \in P,\ i \neq k$$

2.2. Modelo Parte 2

La parte dos describe un problema similar a la primera parte. Trata de asignar un número de llamadas a cada parking, añadiendo la posibilidad de construir algunos, de manera que se minimice el coste de atender todas las llamadas.

El modelo ahora consta de 63 variables, 30 enteras (llamadas asignadas a cada parking desde cada distrito) y 33 binarias (la construcción de un parking y el hecho de que un parking atienda llamadas de un distrito).

El problema cuenta con muchas restricciones:

- Restricción sobre el número máximo de llamadas atendidas por un parking: un parking no puede atender más de 10.000 llamadas.

- Restricción de repartición de llamadas: si un parking consume más del 75% de su capacidad máxima atendiendo llamadas de un único distrito, dicho distrito debe ser repartido entre dos o más parkings.

- Restricción sobre el número total de llamadas atendidas: se deben atender absolutamente todas las llamadas.

- Restricción sobre el tiempo máximo en atender una llamada: un parking no puede atender llamadas de distritos a los que tarde más de 35 minutos.

- Restricción de balance: un parking no puede atender más del 150% de las llamadas de otro parking. Esta restricción se ha partido en dos para tener en cuenta el hecho de que pueden no existir algunos Parkings.

- Condición para acoger llamadas: un parking acoge llamadas de un distrito si y sólo si se le asigna alguna llamada de dicho distrito.

- Condición de cota mínima: si un parking existe y acoge llamadas de un distrito, entonces debe acoger al menos el 10% del total de las llamadas de dicho distrito.

- Condición de existencia: un parking existe sí y sólo sí se le asigna alguna llamada.

Modelo algebraico:

- Constantes del Problema:

  C_{ij}: =Tiempo que tarda una ambulancia del distrito *j* al parking *i*.
  L_j: =Número total de llamadas de cada distrito *j*.
  T_{max}: =Tiempo máximo que puede tardar un parking en atender una llamada.

  $CosteParking_k$: =Coste de construir un parking *k* que era candidato

  $CosteLlamada_i$: =Coste de atender de que un parking *i* atienda una llamada

  F_B: =Factor de balanceo entre parkings.

  P_L: =Máximo número de llamadas que puede atender el parking *i*.

  Inf: = Cota inferior de llamadas que un parking puede atender de un distrito

  P_{max}: = Porcentaje máximo del número de llamadas que un parking puede atender de un distrito

  M: = Constante con un valor cercano a infinito

- Variables de decisión

  l_{ij}: =Número de llamadas que atiende el parking *i* del distrito *j*.

  At_{ij}: =Variable binaria que relaciona si un parking *i* atiende al menos una llamada de *j*

  X_i: =Variable binaria que determina qué parkings reciben

- Función objetivo: Sean P el conjunto de parkings, D el conjunto de distritos, E el conjunto de parking existentes y C el conjunto de parkings candidatos.

$$min\, c = \sum_{i \epsilon P} \sum_{j \epsilon D} (C_{ij} \cdot CosteLlamada_i \cdot l_{ij}) + \sum_{k \epsilon C} X_k \cdot CosteParking_k$$

- Restricciones

$$\sum_{j \in D} l_{ij} \leq P_{max}$$

$$\forall d \in D: L_d > P_{max} \cdot P_L, \sum_{i \in P} At_{id} \geq 2$$

$$\sum_{i \in P} l_{ij} = L_j, \forall j \in D$$

$$l_{ij}(T_{max} - C_{ij}) \geq 0, \forall i \in P, \forall j \in D$$

$$\sum_{j \in D} l_{ij} \leq F_B \sum_{h \in D} l_{kh} + M \cdot (1 - X_{k'}), \forall i, k \in P, i \neq k$$

$$l_{ij} \leq M \cdot At_{ij}, \forall i \epsilon P, \forall j \epsilon D$$

$$l_{ij} \geq Inf \cdot At_{ij} \cdot L_j, \forall i \epsilon P, \forall j \epsilon D$$

$$\sum_{h \in D} At_{kh} \leq M \cdot X_k, \forall k \epsilon P$$

Pese a que el valor de la variable X_i es arbitrario si At_{ij} toma el valor 0, dado que la función objetivo es de minimización, se le asignará valor 0 para reducir el coste.

3. Análisis de resultados

En ambos modelos se alcanza una solución entera factible. En el primero de 483800 minutos, mientras que en el segundo de 872030€. El primer resultado coincide con el resultado obtenido con la implementación en LibreOffice, lo que nos lleva a concluir que el modelo es correcto. El segundo modelo se verifica mediante el análisis de la solución en comparación con las restricciones descritas en el enunciado. Al comparar el resultado obtenido con estas restricciones, y no con nuestro modelo algebraico, podemos ver que nuestro modelo está bien definido y resuelve adecuadamente el problema.

Solución de la parte 1:

Para el problema descrito en el enunciado, se obtiene un valor de la función de optimización de 483800 minutos, quedando la distribución de las llamadas de la siguiente forma:

	D1	D2	D3	D4	D5

L1	2400	0	0	7500	0
L2	2600	1900	0	0	5500
L3	0	4600	5400	0	0

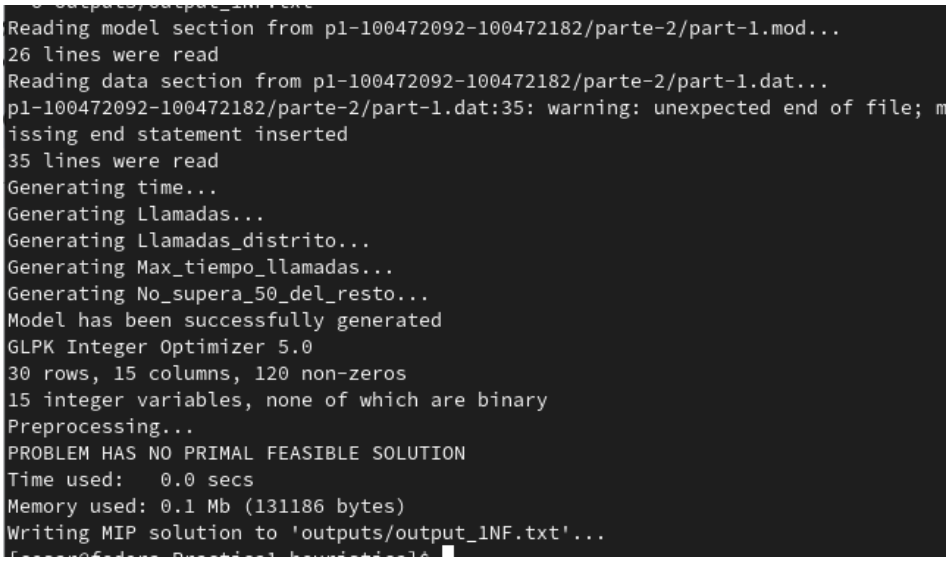
El problema cuenta con 15 variables de decisión (producto de parkings y distritos) y 30 restricciones (resultado de desdoblar las descritas anteriormente).

Si definimos otro set de datos vemos diferentes soluciones, con distinto número de variables y restricciones:

Problema infactible

Para que no exista solución definimos un número de llamadas totales que no sea imposible de atender (i.e. número de llamadas mayor al producto de la máxima capacidad del parking y el número de parkings).

Al ejecutar el modelo con este set de datos obtenemos la siguiente salida:



Problema con un set de datos reducido

Para obtener un problema con datos reducidos vamos a trabajar únicamente con 2 parkings (L1 y L2) y 2 distritos (D3 y D5).

Con esto, la nueva solución para la variable objetivo es 318620 minutos, con la siguiente distribución de llamadas:

	D3	D5

L1	0	4360
L2	5400	1140

Se aprecia claramente que cumple todas las restricciones.

Solución de la parte 2:

La función de optimización ahora modela el coste de atender las llamadas, teniendo un valor de 872030€. El problema aumenta considerablemente su complejidad, subiendo a 30 el número de variables de decisión enteras (producto de parkings y distritos) y 33 nuevas variables binarias (producto de parkings y distritos + número de parking) . Además de 169 restricciones. Con esto presente, la nueva distribución de llamadas queda de la siguiente forma:

	D1	D2	D3	D4	D5
L1	0	0	0	0	0
L2	0	0	0	6750	0
L3	0	1214	5400	0	0
L4	5000	4171	0	750	0
L5	0	1115	0	0	5500
L6	0	0	0	0	0

Vemos que todos los parkings reciben llamadas salvo el parking 6, lo que significa que los parkings 4 y 5 se construyen.

Si definimos otro set de datos vemos diferentes soluciones, con distinto número de variables y restricciones:

Problema infactible

Para que no exista solución, definimos un número de llamadas total que sea imposible de atender (i.e. número de llamadas mayor al producto de la máxima capacidad del parking y el número de parkings)

```
Reading model section from p1-100472092-100472182/parte-2/part-2.mod...
42 lines were read
Reading data section from p1-100472092-100472182/parte-2/part-2.dat...
p1-100472092-100472182/parte-2/part-2.dat:64: warning: unexpected end of file; m
issing end statement inserted
64 lines were read
Generating coste...
Generating Llamadas...
Generating Redistribucion_distrito...
Generating Llamadas_distrito...
Generating Max_tiempo_llamadas...
Generating No_supera_50_del_resto...
Generating Condicion_acoger_llamadas...
Generating Condicion_acoger_llamadas2...
Generating Cota_minima...
Generating Condicion_existencia...
Model has been successfully generated
GLPK Integer Optimizer 5.0
170 rows, 66 columns, 680 non-zeros
66 integer variables, 36 of which are binary
Preprocessing...
PROBLEM HAS NO PRIMAL FEASIBLE SOLUTION
```

Al ejecutar el modelo con el nuevo set de datos obtenemos la siguiente salida:

Problema con set de datos reducidos

Para obtener una solución fácilmente evaluable definimos un set de datos reducido, con dos Parkings y dos distritos. Uno de los parkings existe, mientras que el otro es candidato. También, se ha reducido el coste de construir un nuevo parking.

Con esto, :

	D1	D2
L1	100	500
L6	0	0

Se aprecia claramente que cumple todas las restricciones.

Comparación entre LibreOffice y Mathprog

Ambas herramientas son igualmente efectivas, llegando a la solución en poco tiempo. Sin embargo, creemos que Mathprog presenta claras ventajas dado que cuenta con la versatilidad propia de un lenguaje de programación.

La modelización en LibreOffice tiene claras limitaciones como son el formato de las hojas de cálculo y la difícil navegación en problemas ligeramente complejos. Además, no permite la generalización de un modelo a distintos set de datos.

En cambio, en Mathprog se pueden modelar problemas de cualquier complejidad (siempre que cumplan la restricción de linealidad) y estos serán escalables a cualquier tamaño del set de datos. También, facilita estructuras para definir las restricciones de forma mucho más compacta (i.e. matrices, arrays, parámetros…). La única desventaja de Mathprog es aprender el propio lenguaje, pero una vez superada esa barrera presenta múltiples ventajas que lo hacen una opción preferible al uso de hojas de cálculo.

4. Conclusiones

Tras finalizar esta práctica podemos concluir que la principal dificultad es el diseño del modelo del problema, en concreto el diseño de las restricciones relacionadas con variables binarias. La traducción de lógica proposicional a lenguaje algebraico es una tarea ardua que nos ha obligado a repetir varias veces el modelo hasta llegar a unas restricciones que cumplían correctamente con lo descrito en el enunciado.

Otro obstáculo que nos hemos encontrado es la validación del modelo. Las soluciones intermedias obtenidas, pese a caer en la zona factible, no eran la solución óptima, ya que eran producto de un modelo incorrecto. No obstante, el análisis de estas soluciones y la revisión del enunciado nos ha permitido descartarlas adecuadamente.

Esta práctica nos ha permitido conocer distintas herramientas para resolver problemas de programación lineal, tales como el *solver* de las hojas de cálculo y *Mathprog*. Para esta última ha sido entender y aprender cómo funciona el lenguaje y los resultados. Además esta práctica nos ha enfrentado a un diseño más complejo que nos ha permitido afianzar los conceptos de la programación lineal.