UNIVERSIDAD CARLOS III DE MADRID

GRADO EN INGENIERÍA INFORMÁTICA

HEURÍSTICA Y OPTIMIZACIÓN

Práctica: Programación Lineal

Autores
ALBERTO VILLANUEVA NIETO 100374691
CRISTIAN CABRERA PINTO 100363778

4 de noviembre de 2018

Índice

1. Intorducción

En este documento se explican los modelos de los ejercicios de programación lineal y dinámica, en los de lineal explicando que representa cada conjunto, variable, y restricción. A continuación se analizan los resultados de los modelos, su complejudad y las ventajas y desventajas de usar calc repecto a GLPK y finalmente se procede a la conclusión de la práctica.

2. Descripción de los modelos

2.1. Modelo para la compra

En esta parte se pide minimizar el coste de la compra de unos elementos entre 2 fábricas en función de sus límites de inventario donde el precio del elemento y del transporte desde la fábrica hasta donde se quieren tener varia en función de la fábrica.

2.1.1. Conjuntos

Para poder expresar las variables, los parámetros y las ecuaciones de una forma más descriptiva se va a hacer uso de un conjunto que describe los distintos tipos de productos que hay:

$$productos := \{loc, vag1, vag2, cont20, cont40\}$$

2.1.2. Variables de decisión

Aunque a primera vista puede parecer que se necesitan 10 variables de decisión (por cada tipo de elemento en cada fábrica) sin embargo, si se sabe los elementos totales que se quieren obtener y los que se van a comprar en una de las dos fábricas, los elementos que se compran en la fábrica restante es la diferencia entre los elementos que se quieren obtener y los que se compran en la primera fábrica, de este modo nuestras variables de decisión serían:

$$\begin{aligned}
x_{loc} &: Locomotoras\ compradas\ en\ la\ primera\ fábrica \\
x_{vag1} &: Vagones\ de\ tipo\ 1\ comprados\ en\ la\ primera\ fábrica \\
x_{vag2} &: Vagones\ de\ tipo\ 2\ comprados\ en\ la\ primera\ fábrica \\
x_{cont20} &: Contenedroes\ de\ 20'\ comprados\ en\ la\ primera\ fábrica \\
x_{cont40} &: Contenedroes\ de\ 40'\ comprados\ en\ la\ primera\ fábrica
\end{aligned}$$

2.1.3. Parámetros

Para el cálculo de la función objetivo y de las restricciones se han hecho con respecto a varios parámetro que se nos proporcionaban en el enunciado:

$$\begin{aligned}
Necesidad_p &: Cantidad\ de\ productos\ tipo\ p\ que\ se\ necesitan.\\
Stock\ Primera_p &: Cantidad\ de\ productos\ tipo\ p\ que\ la\ primera\ fábrica\ puedeproveer.\\
Stock\ Segunda_p &: Cantidad\ de\ productos\ tipo\ p\ que\ la\ segunda\ fábrica\ puedeproveer.\\
Coste\ Primera_p &: Coste\ de\ comprar\ y\ transportar\ un\ elemento\ p\ de\ la\ primerafábrica\ al\ destino.\\
Coste\ Segunda_p &: Coste\ de\ comprar\ y\ transportar\ un\ elemento\ p\ de\ la\ segundafábrica\ al\ destino.\\
& \forall p \in productos
\end{aligned}$$

Cuadro 1: Parámetros de compra

Parámetro	Locomotoras	Vagones		Contenedores	
		Tipo 1	Tipo 2	20′	40′
Necesidad	3	3	2	4	4
Stock Primera	2	3	2	4	4
Stock Segunda	3	3	2	2	3
Coste Primera	100200	10500	20040	20080	30040
Coste Segunda	200220	3440	5660	25220	40220

2.1.4. Función objetivo

Para la función objetivo se pretende minimizar el coste de comprar los productos en ambas fábricas:

$$minimizar\ coste = \sum_{p}^{productos} (x_p * CostePrimera_p + (Necesidad_p - x_p) * CosteSegunda_p)$$

2.1.5. Restricciones

El problema de optimización esta restringido con que las variables sean no negativas (1) enteras (2), que se compren más productos en la primera fábrica que en la segunda (3), que el precio gastado en cada fábrica no supere en un 10 % al gastado en al otra (4)(5) y que los productos que se compren en cada una de las fábricas esten dentro del límite que tenga esa fábrica (6)(7).

$$x_p \geq 0, \forall p \in productos \tag{1}$$

$$x_p \in \mathbb{Z}, \forall p \in productos \tag{2}$$

$$-1 + \sum_{p}^{productos} x_p > \sum_{p}^{p} roductos(Necesidad_p - x_p) \tag{3}$$

$$1{,}1 * \sum_{p}^{productos} x_p * CostePrimera_p \geq \sum_{p}^{productos} (Necesidad_p - x_p) * CosteSegunda_p \tag{4}$$

$$\sum_{p}^{productos} x_p * CostePrimera_p \leq 1{,}1 * \sum_{p}^{productos} (Necesidad_p - x_p) * CosteSegunda_p \tag{5}$$

$$x_p \leq StockPrimera_p, \forall p \in productos \tag{6}$$

$$Necesidad - x_p \leq StockSegunda_p, \forall p \in productos \tag{7}$$

2.2. Modelo para la asignación

En esta parte se pide minimizar el coste del transporte de una cantidad de toneladas en 2 rutas, para ello se dispone de los elementos que se han comprado en la parte anterior donde los contenedores y vagones tienen un límite dado por su tipo.

2.2.1. Conjuntos

Para expresar las variables de decisión y los parámetros con respecto a los elementos en especifico creamos varios conjuntos para asi poder tener bien controlado a que parámetro se hace referencia:

$$
\begin{aligned}
contenedores\ tipos :=& \{Cont20, Cont40\} \\
contenedores_{Cont20} :=& \{C1, C2, C3, C4\} \\
contenedores_{Cont40} :=& \{C5, C6, C7, C8\} \\
contenedorestodos :=& \bigcup_{ct}^{contenedores\ tipos} contenedores_{ct} \\
vagones\ tipos :=& \{Vag1, Vag2\} \\
vagones_{Vag1} :=& \{V1, V2, V3\} \\
vagones_{Vag2} :=& \{V4, V5\} \\
vagonestodos :=& \bigcup_{vt}^{vagones\ tipos} vagones_{vt} \\
locomotoras :=& \{L1, L2, L3\} \\
rutas :=& \{Madrid - Valencia, Madric - Cádiz\}
\end{aligned}
$$

2.2.2. Variables de decisión

Al tratarse de un ejercicio de programación lineal de asignación, las variables son matrices binarias que representa la distribución de los elementos de las filas entre los elementos de las columnas. Por los requisitos que pide el enunciado las matrices que se necesitan son las siguientes:

$$
\begin{aligned}
CV_{contenedores\ todos\ x\ vagones\ todos} :& Indica\ en\ que\ vagón\ va\ cada\ contenedor.^{1} \\
CR_{contenedores\ todos\ x\ rutas} :& Indica\ en\ que\ ruta\ va\ cada\ contenedor. \\
VR_{vagones\ todos\ x\ rutas} :& Indica\ en\ que\ ruta\ va\ cada\ vagón. \\
LR_{locomotoras\ x\ rutas} :& Indica\ en\ que\ ruta\ va\ cada\ locomotora.^{1}
\end{aligned}
$$

[1]Estas variables solo son necesarias porque el enunciado dice que se tienen que tener bien controladas donde va cada elemento, pero realmente si conoces que contenedores y que vagones van en una ruta, asignar los contenedores a los vagones es una tarea elemental, y con respecto a las locomotoras es aun más elemental ya que no hay restricciones en el enunciado luego se puede asumir que con una locomotora por ruta es suficiente.

2.2.3. Parámetros

Para el cálculo de la función objetivo y las restricciones, se han sintetizado los datos del enunciado en los siguientes parámetros:

$$
\begin{aligned}
Distancia_r &: Distancia\ que\ tiene\ que\ recorrer\ un\ elemento\ en\ la\ ruta\ r. \\
&\forall r \in rutas \\
Mercancias_r &: Cantidad\ de\ mercancias\ que\ tiene\ que\ se\ tienen\ que\ trasladar\ en\ la\ ruta\ r. \\
&\forall r \in rutas \\
Traslado_{ct,r} &: Lo\ que\ cuesta\ trasladar\ un\ contenedor\ de\ tipo\ ct\ en\ la\ ruta\ r. \\
&\forall ct \in contenedores\ tipos, \forall r \in rutas \\
Limite\ Contenedor_{ct} &: La\ cantidad\ de\ mercancia\ que\ puede\ trasladar\ un\ contenedor\ de\ tipo\ ct. \\
&\forall ct \in contenedores\ tipos \\
Limite\ Vagón_{vt,ct} &: La\ cantidad\ de\ contenedores\ tipo\ ct\ que\ puede\ trasladar\ un\ vagón\ tipo\ vt. \\
&\forall vt \in vagones\ todos, \forall ct \in contenedores\ tipos
\end{aligned}
$$

Cuadro 2: Parámetros de asignación. Rutas

		Madrid-Valencia	Madrid-Cadiz
Distancia		360	650
Mercancias		90	150
Traslado	Cont20	10	20
	Cont40	15	30

Cuadro 3: Parámetros de asignación. Contenedores

		Cont20	Cont40
Limite Contenedor		32	67
Limite Vagón	Vag1	2	0
	Vag2	2	1

2.2.4. Función objetivo

Para la función objetivo se pretende minimizar el coste de trasladar los contenedores en las rutas.

$$minimizar\ coste = \sum_{r}^{rutas} \left(\sum_{ct}^{contenedores\ tipos} \left(\sum_{c}^{contenedores_{ct}} CR_{c,r} \right) * Traslado_{ct,r} \right) * Distancia_r$$

2.2.5. Restricciones

El problema, por como esta presentado, presenta 3 restricciones muy claras; que se tienen que transportar cierta cantidad de mercancias en cada ruta (9), que las rutas cumplan los límites de contenedores que los vagones asignados a esa ruta le permitan(10) y que se tiene que evitar el uso innecesario de vagones(11). También, al tratarse de un problemade asignación, en nuestras variables, un elemento solo puede ser asignado a una cosa (16-19) y además se asume tiene que haber al menos una locomotora en cada ruta (15).

Para asegurarse de que la coherencia entre las variables CV, VR, y VR se restringe que los vagones cumplan los límites de contenedores que pueden trasladar (12), que se asignen a vagones los mismos contenedores que se asignan a rutas (13) y que, teniendo en cuenta esas otras dos restricciones, si un contenedor esta asignado a una vagón y un vagón esta asignado a una ruta, que el contenedor este asignado a esa ruta (14).[2]

$$CV_{c,v}, CR_{c,r}, VR_{v,r}, LR_{l,r} \in \mathbb{Z}_2$$

$$\forall c \in contenedores\ todos, \forall v \in vagones\ todos, \forall l \in locomotoras, \forall r \in rutas \tag{8}$$

$$\sum_{ct}^{contenedores\ tipos} \left(\left(\sum_{c}^{contenedoresct} CR_{c,r} \right) * Limite\ Contenedor_{ct} \right) \geq Mercancias_r;$$

$$\forall r \in rutas \tag{9}$$

$$\sum_{c}^{contenedores_{ct}} CR_{c,r} \leq \sum_{vt}^{vagones\ tipos} \left(\left(\sum_{v}^{vagones_{vt}} VR_{v,r} \right) * Limite\ Vagón_{vt,ct} \right);$$

[2] La forma más directa de hacer esto es usar un and, sin embargo trasladar un and del mundo binario al entero lo convierte en una multiplicación de variables por lo que deja de ser lineal, para solucionarlo se ha hecho una restricción (14) que te garantiza: a=1 and b = 1 –>c = 1, pero si a o b es 0 no te garantiza que c sea 0, para completar esto y que así tenga la funcionalidad de una and se han usado otras 2 restricciones(12)(13).

$$\forall r \in rutas, \forall ct \in contenedores\ tipos \tag{10}$$

$$-1 + \sum_{c}^{contenedores\ todos} CR_{c,r} \geq \sum_{v}^{vagones\ todos} VR_{v,r};\ \forall r \in rutas \tag{11}$$

$$\sum_{c}^{contenedores_{ct}} CV_{c,v} \leq Limite\ Vag\acute{o}n_{vt,ct};$$

$$\forall vt \in vagones\ tipos, \forall v \in vagones_{vt}, \forall ct \in contenedores\ tipos \tag{12}$$

$$\sum_{r}^{rutas} CR_{c,r} = \sum_{v}^{vagones\ todos} CV_{c,v};\ \forall c \in contenedores\ todos \tag{13}$$

$$CR_{c,r} + CV_{c,v} - 1 <= VR_{v,r};\ \forall r \in rutas, \forall c \in contenedores\ todos, \forall v \in vagones\ todos \tag{14}$$

$$\sum_{l}^{locomotoras} LR_{l,r} \geq 1;\ \forall r \in rutas \tag{15}$$

$$\sum_{r}^{rutas} CR_{c,r} \leq 1;\ \forall c \in contenedores\ todos \tag{16}$$

$$\sum_{r}^{rutas} VR_{v,r} \leq 1;\ \forall v \in vagones\ todos \tag{17}$$

$$\sum_{r}^{rutas} LR_{l,r} \leq 1;\ \forall l \in locomotoras \tag{18}$$

$$\sum_{v}^{vagones\ todos} CV_{c,v} \leq 1;\ \forall c \in contenedores\ todos \tag{19}$$

2.3. Programación Dinámica

2.3.1. Modelo

$$V_i^C = max\{V_{i-1}^C, V_{i-1}^{C-p_i} + v_i\}$$

Donde:
V_i^C es la utilidad máxima que se puede conseguir con los primeros i elementos con una capacidad de C.
p_i es la capacidad que ocupa el elemento p_i.
v_i es la utilidad de elemento v_i.

Para ello se ha usado el uso de una tabla y se ha ido calculando todos los elementos por columnas empezando con que la primera fila y primera columna tienen valor 0 (la máxima utilidad de 0 elementos o de 0 de capacidad es 0 asumiendo que la capacidad tiene que ser positiva).
Una vez se tiene la utilidad máxima se puede recuperar los elementos que se usan de una forma muy fácil, si en una casilla cualquiera, $V_i^C = V_{i-1}^C$ eso quiere decir que el elemento i forma parte de la solución de utilidad máxima, y el resto de elementos son los correspondientes a la solución de $V_{i-1}^{C-p_i}$. Si $V_i^C \neq V_{i-1}^C$ el elemento i no pertenece a la solución optima luego los elementos que si pertenecen son los correspondientes a al solución de V_{i-1}^C.

2.3.2. Cuestiones

Un algoritmo de fuerza bruta tendria que comprobar todas las combinaciones de inclusión/exclusión de los contenedores, es decir $2^{100} \approx 1{,}26 * 10^{30}$, este algoritmo de programación dinámica realiza $9 * 10^5$ (sin contabilizar la primera fila y columna ya que son solo 0).
En este caso en concreto, al usarse una tabla que se calcula entera, no hay diferencia, sin embargo si se hiciera con el algoritmo de forma recursiva, el algoritmo se beneficiaria de ordenarlo en función de la capacidad que ocupa ya que asi C decrece más rapido al principio y se acaba ramificando menos.

3. Analisis de los resultados

3.1. Solucion

A continuación se muestra una solución obtenida, en una de las ejecuciones, sin embargo hay más de una solución optima, ya que 2 vagones o contenedores del mismo tipo son completamente intercambiables sin que afecte a la solución.

Productos comprados en la Fabrica 1:
    locomotoras: 2
    vagones1: 2
    vagones2: 0
    contenedores20: 3
    contenedores40: 2
Productos comprados en la Fabrica 2:

locomotoras: 1
vagones1: 1
vagones2: 2
contenedores20: 1
contenedores40: 2

Disposición de contenedores en vagones:
C1 –>V1
C2 –>V2
C3 –>V1
C4 –>V4
C5 –>V4
C6 –>V5
C7 –>No Asignado
C8 –>No Asignado

Disposición de contenedores en rutas:
C1 –>Madrid-Valencia
C2 –>Madrid-Valencia
C3 –>Madrid-Valencia
C4 –>Madrid-Cadiz
C5 –>Madrid-Cadiz
C6 –>Madrid-Cadiz
C7 –>No Asignado
C8 –>No Asignado

Disposición de vagones en rutas:
V1 –>Madrid-Valencia
V2 –>Madrid-Valencia
V3 –>No Asignado
V4 –>Madrid-Cadiz
V5 –>Madrid-Cadiz

Disposición de locomotoras en rutas:
L1 –>Madrid-Valencia
L2 –>Madrid-Cadiz
L3 –>No Asignado

Coste Minimizado de la compra: 662360
Coste Minimizado del traslado: 62800
Coste Minimizado total : 725160

3.1.1. Cumplimiento de restricciones

En la primera fábrica se compran 9 productos y en la fábrica 2 se compran 7 luego se compran más en la fábrica que esta más cerca.
En la primera fábrica, el precio total de las compras es de 341720 y en la segunda de 320640 luego evita que alguna fábrica tenga monopolio.
Mirando el cuadro 1 se puede comprobar que no se supera el límite permitido por ninguna fábrica.
Segun los contenedores asignados a rutas, se pueden llevar 96 y 166 toneladas respectivamente a la ruta de Madrid-Valencia y Madrid-Cadiz luego se satisfacen la cantidad de mercancias que se tienen que transportar.
Mirando el cuadro 3 y la asignación de contenedores a vagones se ve que no se supera el límite que cada tipo de vagón permite.
En ambas rutas se usan 2 vagones y 3 contenedores luego se evita el uso innecesario.

3.1.2. Restricciones que limitan el problema

En el problema de la compra, el límite de stock no cambia en nada a no ser que el stock de la fábrica que tiene el producto más barato no tenga suficiente como para suministrar la necesidad, la prioridad de tiempo tiene un ligero impacto pero no muy grande y el que más impacto tiene es evitar el monopolio (quitando esa restricción el coste de la compra pasa de 662360 a 622740) ya que se obliga a comprar productos más caros para que el balance se mantenga. En el problema de la asignación, la restricción de los límites que tienen los vagones afecta un poco (ya que usar contenedores de 40′ es más eficiente que los de 20′ segun lo que cuesta trasladarlos) y la restricción de usar más contenedores que vagones no afecta en nada pero la que realmente afecta, claramente, es la carga que se tiene que trasladar, ya que si no se tiene que trasladar nada hay que usar menos contenedores (se sigue teniendo que usar algunos por la restricción de usar más contenedores que vagones y 0 no es mayor que 0 luego se tienen que usar contenedores y suficientes vagones para llevarlos)

3.2. Analisis de complejidad

Para el problema de la compra, teniendo en cuenta los datos y conjuntos, el problema esta modelizado con 5 variables y 13 restricciones (sin tener en cuenta las restricciones para que las variables sean numeros enteros no negativos). Las ultimas 10 restricciones estan hechas para juntar el límite

de stock y el máximo de elementos que se pueden comprar y que de esta forma el modelo permita incluir otro tipo de productos para comprar y otras restricciones de forma más fácil, para este caso en concreto se podria modelizar con solo 7 restricciones ya que muchas de ellas vuelven a ser la restricción de que las variables sean no negativas(como esta puesto en el archivo de calc). Teniendo en cuenta las restricciones de stock, el problema puede tomar $3^3 + 4^2$, es decir, 43 posibles valores.
Para el problema de la asignación, teniendo en cuenta los datos y conjuntos, el problema esta modelizado con 72 variables binarias, es decir, que hay $2^{72} \approx 4{,}7 * 10^{21}$ posibles valores que pueden tomar nuestro resultado, y con 132 restricciones (sin tener en cuenta las que limitan a que las variables sean binarias).
A la hora de cambiar las variables y parámetros, en el problema de la compra, solo afecta la cantidad que se quiera comprar total y el stock máximo de cada fábrica, al disminuir el stock de la primera fábrica decrece el límite superior de las variables y al decrecer el el stock de la segunda aumenta el límite inferior. Con respecto a el problema de la asignación, la formula para sacar la cantidad de restricciones es:

$$3 * r + r * ct + v * ct + 3 * c + r * v * c + v + l$$

$$\begin{aligned} r &: \textit{número de rutas} \\ l &: \textit{número de locomotoras} \\ v &: \textit{número de vagones} \\ c &: \textit{número de contenedores} \\ vt &: \textit{número de tipos de vagones} \\ ct &: \textit{número de tipo de contenedores} \end{aligned}$$

Sacando los valores a los que afecta el crecimiento de cada variable sacamos lo siguiente:

$$\begin{aligned} r &: 3 + ct + v * c \\ l &: 1 \\ v &: 1 + ct + r * c \\ c &: 3 + r * v \\ vt &: 0 \\ ct &: r + v \end{aligned}$$

Tomando las siguientes suposiciones

$$\begin{aligned} c &> v > r \\ c &> ct \\ c*v &> c*r > v*r \end{aligned}$$

Sacamos la siguiente relación de cuanto afecta cada variable a la complejidad compuesta por las restricciones:

$$r > v > c > ct > l > vt = 0$$

Y con respecto a la complejidad de las variables, siguiendo el mismo planeteamiento:

$$\begin{aligned} &Ecuación: \; c*v + c*r + v*r + l*r \\ &Suposiciones: \; c > v > l > r \\ &Lo\ que\ afecta\ cada\ variable\ al\ crecimiento: \; r > v > c > l > ct = vt = 0 \end{aligned}$$

Lo que además es consistente con lo que afecta a la complejidad de las restricciones, excepto los tipos de contenedores, y también es consistente con los ejemplos hechos.

3.3. Cuestiones

3.3.1. Composición final

Madrid-Valencia: C1-C3->V1-C2->V2_L1
    Contenedores: 3
    Vagones: 2
Madrid-Cadiz: C4-C5->V4-C6->V5_L2
    Contenedores: 3
    Vagones: 2

3.3.2. Mejor tipo de Vagón

Mirando a el cuadro 1 y 2, se puede ver que un contenedor de 40′ es objetivamente mejor ya que el coste de tranportar una tonelada por kilómetro y el precio que te cuesta una tonelada de almacenamiento al comprar los ocntenedores es menor en los de 40′ para ambas rutas y ambas fábricas.

3.4. Calc frente a GLPK

Habiendo modelado al menos el primer trabajo en las dos plataformas, y habiendo explorando en profundidad las posibilidades que ofrece GLPK, calc es una herramienta para el modelado de tareas de programación lineal que es muy gráfico y permite ver con facilidad las restricciones, las variables y los resultados, no obstante, a diferencia de GLPK, es muy complicado hacer muchas restricciones ya que en GLPK se puede hacer de forma iterativa y en calc se tiene que poner explicitamente todas y cada una de ellas.

4. Conclusiones acerca de la práctica

La práctica nos ha llevado a la realización de ejercicios de programación lineal por medio de calc y de GLPK. Con ello hemos podido comprobar que el método simplex en calc puede ser beneficioso en el caso de que sea un problema muy sencillo ya que estar poniendo restricciones una a una como pasa en el caso del segundo ejercicio sería muy tedioso. Sin embargo, con GLPK se nos permite la realización de restricciones de forma iterativa lo cual puede facilitar el desarrollo del problema aunque el lenguaje empleado sea un poco complicado en un principio.
También hemos conseguido entender mejor el método simplex y los tipos de funciones que se permiten en la programación lineal ya que se intentó hacer un modelo más eficiente para el modelo de la asignación pero lo más cercano que se pudo llegar a una funcion lineal con ese modelo fue usando el valor absoluto de una función, lo que no sabiamos que no era una función lineal.