Práctica 1

Programación Lineal

Enlace a GitHub: https://github.com/aaronespasa/school-bus-routing

Aarón Espasandín Geselmann

Alejandra Galán Arróspide

Introducción

Este documento consta de 5 partes principales:

Las dos primeras, están confencianadas por la descripción de los modelos que tratan de resolver las problemas propuestos en las partes uno y dos. Dichas partes constarán de una descripción de los componentes de los modelos junto con su implementación en Calc y en GLPK seguida de un análisis de las decisiónes tomadas para llegar a ellos.

Las dos siguientes partes constarán de un análisis de los resultados obtenidos con dichos modelos.

En la penúltima parte trataremos las ventajas y desventajas de utilizar Calc frente a GLPK.

Por último, se encontrará en el documento una sección con las conclusiones acerca del trabajo.

Descripción modelo (Parte 1)

En esta parte es sabido el número de alumnos que se encuentran en las tres paradas. Las variables de decisión y restricciones del modelo han sido escogidas con el objetivo de reducir todo lo posible el número de restricciones y de que este sea lo más generalizable posible. Esto se explicará más a detalle en la sección de análisis de resultados.

Elementos y conjuntos

I: Nodo que representa el parking.

F: Nodo que representa la escuela.

N: Conjunto de nodos -> $N_{\bar{I}} = N \setminus F$, $N_{\bar{F}} = N \setminus F$ y $N_{\bar{I},\bar{F}} = N \setminus \{I, F\}$.

A: Conjunto de rutas (nodo i a nodo j).

Variables de decisión

1. En primer lugar, encontramos una primera matriz de variables de decisión. Dicha matriz representa los caminos por los que pueden pasar los autobuses para lograr su objetivo de recoger los alumnos y llegar al colegio. Esta matriz tiene dimensiones de N x N, siendo N el número de nodos. Estas variables de decisión tienen caracter binario, pudiendo de este modo tomar valores entre 0 y 1 unicamente según haya pasado algún bus por ese camino o no ($x_{i,j}$: Distancia entre el nodo i y el nodo j):

Caminos		XI	X1	X2	X3	XF
	XI	0	1	0	1	0
	X1	0	0	1	0	0
	X2	0	0	0	0	1
	X3	0	0	0	0	1
	XF	0	0	0	0	0

2. En segundo lugar, encontramos una segunda matriz de variables de decisión. Se trata de una matriz de NxN siendo N el número de nodos. Representa los la cantidad de personas que se mueven entre los diferentes enlaces.

   Esta matriz nos es de mucha utilidad ya que nos permite utilizarla como un conjunto de variables en memoria que almacenarán cuantos alumnos se están moviendo entre los diferentes caminos. Esto nos ayuda mucho a la hora de generalizar el problema como veremos en las restricciones.

   Estas variables de decisión tienen caracter entero que las restricciones limitirán que nunca se supere la capacidad máxima de alumnos en un bus ($f_{i,j}$: Flujo desde el nodo i hasta el nodo j):

Movimiento de Alumnos		XI	X1	X2	X3	XF
	XI	0	0	0	0	0
	X1	0	0	15	0	0
	X2	0	0	0	0	20
	X3	0	0	0	0	10
	XF	0	0	0	0	0

Datos

Todas las restricciones emplean los valores asignados a los datos, no se les han asignado valores de forma directa.

1. Matriz N x N (misma estructura que las variables de decisión) que contiene el coste de pasar por cada camino ($d_{i,j}$: Distancia entre el nodo i y el nodo j.):

Longitud Caminos		XI	X1	X2	X3	XF
	XI		8	10	10	
	X1	18		3	7	6
	X2	10	3		5	7
	X3	10	7	5		4
	XF		6	7	4	

2. Máximo número de buses (MB).

Máx. Núm. de buses	3

3. Máxima capacidad de los buses (C).

Capacidad máxima de cada bus	20

4. Alumnos en cada parada (g_i: Número de alumnos en el nodo i.):

Alumnos en S1	15
Alumnos en S2	5
Alumnos en S3	10

5. Precio añadido por bus (bus_{price}).

Precio por bus (€)	120

6. Precio añadido por km (km_{price}).

Precio por km (€)	5

Restricciones

El modelo consta de 10 restricciones;

1. Limita el número de buses que pueden salir de el parking para que este no supere el máximo número de buses. Para esto, contamos el número de rutas activadas que salen desde el parking, lo cual lo podemos hacer al saber que cada bus solamente puede pasar por un único camino:

$$\sum_{n \in N_{\bar{I},\bar{F}}} x_{I,i} \leq MB$$

2. Obliga a que todos los buses que salen del parking (es decir el número de caminos activados con origen en el parking), sea el mismo que el número de buses que llegan al colegio (es decir el número de caminos activados con destino en el colegio):

$$\sum_{n \in N_{\bar{I},\bar{F}}} x_{I,n} - \sum_{k \in N_{\bar{I},\bar{F}}} x_{k,F} = 0$$

3. Obliga a que de cada parada solo pueda salir una única ruta (exceptuando el parking):

$$\sum_{j \in N_{\bar{I}}} x_{i,j} \leq 1 \ , \quad \forall i \in N_{\bar{I},\bar{F}} \ , \ i \neq j$$

4. Restringe que cada nodo (parada) debe ser visitado una vez (exceptuando el colegio) , es decir, un solo bus puede recoger a los alumnos en esa parada. Esta restricción junto con la de control del flujo evita que sea necesario comprobar que todos los alumnos de la parada han sido recogidos:

$$\sum_{i \in N_{\bar{F}}} x_{i,j} = 1 \quad , \quad \forall j \in N_{\bar{I},\bar{F}} \quad , \quad i \neq j$$

5. No permitir que un bus pueda volver por el mismo camino (debiendo ser la suma de la ida y de la vuelta de dos mismos nodos como máximo 1):

$$x_{i,j} + x_{j,i} \leq 1 \ , \ \forall i \in N_{\bar{I},\bar{F}} \ , \ \forall j \in N_{\bar{I},\bar{F}}$$

6. Evitar que por ninguna ruta puedan moverse más alumnos que la capacidad máxima de alumnos que tiene un bus (C). La primera restricción involucra al parking y a todos los nodos internos, exceptuando el colegio ya que este no está conectado con el parking. La segunda restricción involucra las rutas de los diferentes nodos hasta el colegio:

$$f_{i,j} - Cx_{i,j} \leq 0, \quad i \in N_{\bar{F}} \quad j \in N_{\bar{I},\bar{F}} \quad i \neq j$$

$$f_{i,F} - Cx_{i,F} \leq 0, \quad i \in N_{\bar{I},\bar{F}}$$

7. Evitar que por ningún nodo puedan transitar más alumnos que la capacidad máxima de alumnos que tiene un bus (C). Esta será la restricción que modificará la matriz de variables de decisión del movimiento de alumnos. Esto se debe a que tendrá que encargarse de que el la cantidad de alumnos que entran a una parada sumado a los estudiantes que se encuentran esperando en la misma sea igual a la cantidad de estudiantes que salen de ella:

$$\sum_{i \in N_{\bar{F}}} f_{i,j} + g_j - \sum_{j \in N_{\bar{I}}} f_{j,i} = 0, \quad \forall j \in N_{\bar{I},\bar{F}} \quad i \neq j$$

8. Restricción de integridad: Las variables de decisión que representan si un bus ha pasado entre dos nodos (x_{ij}) solo pueden adoptar el valor de 0 o 1. Significando 0 que por esa ruta no ha transicionado ningún bus aún y 1 que un bus pasa por esa ruta:

$$x_{i,j} \in \{0,1\}, \quad (i,j) \in A$$

9. Restricción de integridad: Las variables de decisión que representan el movimiento de alumnos entre dos nodos ($f_{i,j}$) solo pueden adoptar números naturales incluyendo 0 ya que el número de alumnos no se puede expresar con un número decimal:

$$f_{i,j} \in \mathbb{N} \cup \{0\}, \quad (i,j) \in A$$

Como podemos observar, todas las variables de decisión poseen valores mayores o iguales que 0.

Función Objetivo

La función objetivo es de minimización ya que se busca que el coste de las rutas sea el mínimo posible.

El coste está basado en el número de kilómetros recorridos y el número de buses utilizados. Al solamente poder fluctuar un único bus por una ruta y sabiendo que todos los buses parten desde el parking, podemos obtener el número de buses utilizados contando el número de rutas utilizadas desde el parking ($x_{I,i}$). Sabiendo el número de buses utilizados, solamente deberemos multiplicar este valor por el precio de cada bus para obtener el precio por los buses utilizados.

Respecto a los kilómetros recorridos, podemos calcular este valor simplemente observando si una ruta ha sido utilizada o no ($x_{i,j}$) y multiplicándola por el número kilómetros que posee esta ($d_{i,j}$). Una vez sabemos el número de kilómetros recorridos lo multiplicamos por el precio que cuesta recorrer cada kilómetro.

$$min\ TotalCost\colon km_{price} * \sum_{(i,j) \in A} x_{i,j} * d_{i,j} + bus_{price} * \sum_{i \in N_{\bar{I},\bar{F}}} x_{I,i}$$

Descripción modelo (Parte 2)

Para esta sección los alumnos podrán asignarse a distintas paradas pero se deberá tener en cuenta que los alumnos hermanos deberán asignarse a la misma parada.

Este modelo preservará gran parte de las variables de decisión, datos, conjuntos y restricciones de la primera parte. En su respectiva sección, se mencionarán aquellos que serán eliminados, modificados o añadidos con respecto a la parte 1.

Tras un intento insatisfactorio de modelar el problema creando las variables de decisión y restricciones basándonos en los alumnos como en la parte anterior decidimos pensar el problema de otra forma. Optamos por agrupar los alumnos en familias de hermanos donde cada familia de hermanos contenía los alumnos que eran hermanos. De esta forma conseguimos solucionar el problema con pequeñas modificaciones, tratando de aumentar las restricciones lo menos posible con respecto a la parte 1.

Elementos y conjuntos

Hemos añadido dos conjuntos para agrupar los alumnos en familias de hermanos.

$Brothers\ (B)$: Conjunto que contiene los nombres de las familias de hermanos (B1, B2, B3, …).

$GetBrothers\ (GB)$: Conjunto que utilizaremos como un hash map el cual, dado el nombre de la familia de hermanos nos devolverá los nombres hermanos que pertenecen a esta familia (1, 2, 3, …).

Variables de decisión

Añadimos una nueva matriz de variables de decisión que representa a que parada se asigna cada familia de hermanos (z). Las otras dos matrices de variables de decisión se mantienen.

En intentos anteriores habíamos utilizado esta misma pero utilizando los alumnos en vez de las familias de hermanos. Sin embargo, asumimos que los alumnos de cada familia de hermanos iban a poder ir a las mismas paradas como se muestra en el enunciado (los alumnos A4 y A5 son hermanos y pueden ir tanto a las paradas S1, S2 y S3). Nuestro razonamiento tras esta asunción está basado en que los hermanos suelen vivir juntos en la misma casa, lo cual hace que tengan sentido que puedan ir a las mismas paradas. No hemos tenido en cuenta que puedan ir a colegios distintos.

Al agrupar los estudiantes en familias de hermanos es sencillo ver que si asignamos una familia de hermanos a una parada, todos los estudiantes pertenecientes a este conjunto irán a la misma parada.

	S1	S2	S3
B1	0	0	0
B2	0	0	0
B3	0	0	0
B4	0	0	0
B5	0	0	0
B6	0	0	0
B7	0	0	0

Datos

Se elimina el número de alumnos en cada parada (g_i) ya que no será útil para este problema al poder variar según se asignen los alumnos a las paradas.

Se añaden 3 nuevos datos:

1. Familias de hermanos (B). Es un conjunto que posee los nombres de las distintas familias de hermanos.
2. Alumnos pertenecientes a cada una de las familias de hermanos (GB).
3. Paradas a las que pueden asignarse las distintas familias de hermanos (PossibleFamilyStops, PFS). Se marcará con un 1 aquellas paradas a las que la familia de hermanos se puede asignar y con un 0 aquellas a las que no:

	S1	S2	S3
B1	1	0	0
B2	1	0	0
B3	1	0	0
B4	1	1	1
B5	0	0	1
B6	0	0	1
B7	0	0	1

Restricciones

Se añaden dos nuevas restricciones, además de una nueva restricción de integridad:

10. Cada familia de hermanos tiene que asignarse obligatoriamente a una de sus posibles paradas:

$$\sum_{i \in N_{\bar{I},\bar{F}}} z_{b,i} * PFS_{b,i} = 1, \quad \forall b \in B$$

11. La familia de hermanos solamente puede asignarse a una única parada. La restricción anterior no evitaba que los hermanos pudiesen asignarse a paradas a las que no podían ir. Sin embargo, sí que nos asegurábamos de que estuviesen asignados a una única parada a la que podían ir. Por lo tanto, lo único que necesitábamos hacer para solucionar este inconveniente era asegurarnos de que cada familia de hermanos solamente pudiese ser asignada a una única parada (y por lo tanto, gracias a la restricción anterior, solo se asignará a una de las paradas posibles a las que puede acceder).

A la hora de confeccionar esta restricción hemos asumido que cada familia de hermanos tendrá al menos una parada posible a la que ir ya que no tendría sentido trabajar con una familia de hermanos que no se puede asignar a ninguna parada. Por lo tanto, esta restricción utiliza una igualdad en vez un menor o igual:

$$\sum_{i \in N_{\bar{I},\bar{F}}} z_{b,i} = 1, \quad \forall b \in B$$

12. Restricción de integridad: Las variables de decisión que representan la parada asignada a una familia de hermanos (Z_{bi}) solo pueden adoptar el valor de 0 o 1. Significando 0 la familia de hermanos no está asignada a esa parada y 1 la familia de alumnos sí que está asignada a esa parada:

$$z_{b,i} \in \{0,1\}, \quad (b,i) \in B \times N_{\bar{I},\bar{F}}$$

Se modifica la restricción referente a la restricción de flujos de cada nodo ya que ya no utilizará el número de alumnos en cada parada, sino la cantidad de estudiantes de la familia de hermanos asignada a la parada:

7. Ahora esta restricción deberá de calcular el número de hermanos que posee. Para esto, sumamos la cantidad de estudiantes que posee cada familia de hermanos asignada a la parada y así obtenemos el número de alumnos total de la parada.

$$\sum_{i \in N_{\bar{F}}} f_{i,j} + \sum_{b \in B} z_{b,j} * card(z_{b,j}) - \sum_{j \in N_{\bar{I}}} f_{j,i} = 0, \quad \forall j \in N_{\bar{I},\bar{F}} \quad i \neq j$$

Función Objetivo

Se mantiene la misma función objetivo que en la parte 1.

Análisis de los resultados

Parte 1

```
Total Cost: 400
Buses used: 2
Kms: 32
Edge Pk -> S1
Edge Pk -> S3
Edge S1 -> S2
Edge S2 -> Cl
Edge S3 -> Cl
```

La solución hallada usando Calc como usando GLPK es la misma. La región es factible, acotada y con una solución óptima única. En la solución óptima, la función objetivo tiene el valor de 400, lo que significa que el mínimo coste posible que se puede conseguir es 400 euros.

Analizando la solución óptima podemos ver que el modelo utiliza 2 buses. Uno de ellos se dirige a la parada 1 recogiendo 15 alumnos, después procede a ir a la parada 2 recogiendo 5 alumnos más y acaba en el colegio. El otro bus se dirige a la parada 3, recoge los 10 alumnos que hay y acaba en el colegio. En total se recorren 32 km.

Análisis del rendimiento y de las restricciones

Tenemos 30 restricciones y un total de 52 variables de decisión. Como podemos observar, el número de restricciones es significativamente menor que el número de variables de decisión. Además, 20 de estas variables de decisión utilizan valores binarios, haciendo que la cantidad de operaciones que probar sea menor ya que solo tienen dos posibilidades, acelerando la búsqueda de la solución. Esto puede no ser significativo con pocos caminos, pero con un gran número de caminos creemos que sí que sería más notable.

Creemos de relevancia señalar que solo tendremos más restricciones si se incrementa el número de rutas. Por lo tanto, un aumento en el número de alumnos o buses no afectaría al número de restricciones. Además gran parte de las restricciones bastantes restrictivas ya que los valores resultantes de estas se acercan a sus cotas inferior y superior respectivamente.

Diferentes escenario:

1. **Añadir una parada:**

```
param NumOfNodes := 6;
param EdgesCost default 999 :
      1   2   3   4   5   6 :=
1     0   8   10  10  5   .
2     8   0   3   7   6   6
3     10  3   0   5   7   7
4     10  7   5   0   8   4
5     5   6   7   8   0   9
6     .   6   7   4   9   0;

param MaxNumBuses := 3;
param BusMaxCapacity := 20;

param NumOfStops := 4;
param Stops :=
    1 15
    2 5
    3 10
    4 7;
```

En la imagen podemos observar que se añade una parada, incluyendo los costes de la misma. Como podemos observar se encuentra una solución factible, acotada y con solución óptima única.

```
INTEGER OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.3 Mb (275598 bytes)

------------ Output: ------------
Total Cost: 415
Buses used: 2
Kms: 35
```

2. **Añadir alumnos (más de los que soporta el sistema):**

```
param BusMaxCapacity := 20;

param NumOfStops := 3;
param Stops :=
    1 25
    2 5
    3 10;
```

En la imagen podemos observar que se asignan 25 alumnos a una parada, siendo 20 el máximo del bus. Por ello, se obtiene una solución infactible

```
Integer feasibility conditions:

KKT.PE: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
        High quality

KKT.PB: max.abs.err = 2.50e+01 on row 15
        max.rel.err = 9.62e-01 on row 15
        SOLUTION IS INFEASIBLE
```

3. **Cambio de datos :**

```
param NumOfNodes := 5;
param EdgesCost default 999 :
      1   2   3   4   5 :=
  1   0   8   10  10  .
  2   8   0   3   7   6
  3   10  3   0   5   7
  4   10  7   5   0   4
  5   .   6   7   4   0;

param MaxNumBuses := 6;
param BusMaxCapacity := 26;

param NumOfStops := 3;
param Stops :=
    1 15
    2 5
    3 10;



param PriceBus := 200;
param Pricekm := 6;
```

Se espera que al ser cambios menores, la solución sea factible, acotada y con solución óptima única.

```
INTEGER OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.2 Mb (207970 bytes)

------------ Output: ------------
Total Cost: 592
Buses used: 2
Kms: 32
Edge Pk -> S1
Edge Pk -> S3
Edge S1 -> S2
Edge S2 -> C1
Edge S3 -> C1
---------------------------------
```

Parte 2

El número de variables de decisión enteras se mantiene, solo aumenta el número de variables binarias. Si aumenta el número de estudiantes dentro de una familia no hará que aumente el número de variables de decisión, solamente cuando se creen nuevas familias de hermanos. Esto beneficia al rendimiento.

En cuanto a las restricciones, solo son añadidas 11 restricciones con respecto al caso anterior mientras que encontramos un incremento de 52 variables de decisión.

Estas nuevas restricciones solo se incrementarán si aumenta el número de familias de hermanos. Si solo se incrementa el número de estudiantes dentro de alguna familia de hermanos ya existente no hará que se incremente el número de restricciones. Beneficio al rendimiento.

Diferentes escenarios

Pruebas realizadas:

1. **Añadir una parada:**

```
data;

param NumOfNodes := 6;
param EdgesCost default 999 :
        1   2   3   4   5   6  :=
    1   0   8   10  10  7   .
    2   8   0   3   7   6   6
    3   10  3   0   5   8   7
    4   10  7   5   0   9   4
    5   7   6   8   9   0   4
    6   .   6   7   4   4   0;

param MaxNumBuses := 3;
param BusMaxCapacity := 4;
param NumOfStops := 4;
param PriceBus := 120;
param Pricekm := 5;

set Brothers := "B1" "B2" "B3" "B4" "B5" "B6" "B7";

set GetBrothers["B1"] := 1;
set GetBrothers["B2"] := 2;
set GetBrothers["B3"] := 3;
set GetBrothers["B4"] := 4 5;
set GetBrothers["B5"] := 6;
set GetBrothers["B6"] := 7;
set GetBrothers["B7"] := 8;

param PossibleFamilyStops default 0 :
          1  2  3  4 :=
    "B1"  1  .  .  .
    "B2"  1  .  .  .
    "B3"  1  .  .  .
    "B4"  1  1  1  .
    "B5"  .  .  1  .
    "B6"  .  .  1  .
    "B7"  .  .  1  1;

end;
```

Como podemos observar en la imagen se añade una parada, asi como los enlaces que la unen tanto con el resto de paradas. El resultado esperado era una solución factible, acotada y con solución óptima única.

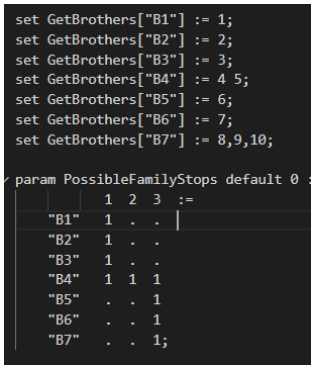
En la segunda imagen podemos observar que como se esperaba, se recibe una solución factible.
En general el comportamiento es correcto, aunque tiene la limitación de que aunque los alumnos son asignados a las paradas de forma que una queda vacía, debido a como fueron definidads las restricciones en nuestro modelo, la ruta pasa por la parada aunque esta esté vacía.

```
------------ Output: ------------
Total Cost: 425
Buses used: 2
Kms: 37

Distribution of brothers accross the stops:
Stop 1: B1 B2 B3
Stop 2:
Stop 3: B4 B5 B6
Stop 4: B7

Edges used:
Edge Pk -> S1
Edge Pk -> S2
Edge S1 -> C1
Edge S2 -> S3
Edge S3 ->
Edge C1 ->
---------------------------------
```

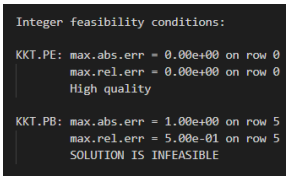
2. **Más alumnos en una parada de los que el autobús puede asumir:**



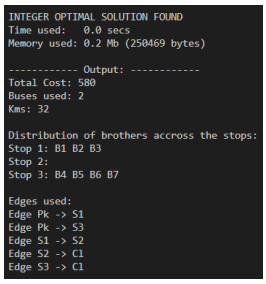
Dado que estamos restringidos a que solo un autobús puede pasar por cada nodo, y un solo autobús no puede recoger todos los alumnos que están en la parada, se espera que la solución sea infactible.

Como podemos observar en la imagen, la configuración asigna 5 alumnos a la misma parada de forma obligatoria, cuando el autobús solo tiene una capacidad máxima de 4.

El comportamiento es el esperado.



3. **Cambio de otros datos:**



En este ejemplo se cambian algunos otros datos como el precio por bus o la capacidad de los mismos. La solución que se espera es factible, acotada y con solución óptima única.

```
param MaxNumBuses := 4;
param BusMaxCapacity := 5;
param NumOfStops := 3;
param PriceBus := 130;
param Pricekm := 10;
```

Podemos concluir que el modelo es lo suficientemente generalizable y encuentra la solución óptima en la mayoría de los casos, exceptuando si una de las paradas se queda vacía.

Ventajas y desventajas de GLPK y Calc

La primera ventaja que hemos encontrado a GLPK con respecto a Calc es la facilidad para describir restricciones derivadas de formulas matemáticas, asi como la comodidad de solo escribir una restriccion cuando esta se repite para diferentes componentes de un set.

En cuanto al uso de los datos, ambos ofrecen funcionalidades parecidas, aunque hemos encontrado más cómodo el uso de GLPK ya que al poder utilizarlo en editores de código como VisualStudio Code, se permite el trabajo interactivo de manera más facil para cambios menores o para visualizar el código de forma simultanea.

Por otra parte, Calc nos ha resultado mucho más visual.

Conclusiones

Uno de los puntos que más apreciamos de la realización de la práctica fue el poder modelar un problema del mundo real utilizando programación lineal ya que nos hizo ver la utilidad que este tiene. Además creemos que Carlos Linares ha sabido transmitirnos muy bien su interés por la programación lineal y hacer que disfrutásemos al realizar la tarea por ver la relevancia que tiene aprender programación lineal en un entorno profesional.

También, ha resultado muy interesante la forma en la cual el enunciado estaba planteado, ya que al tener dos partes, nos hemos visto obligados a desde un principio buscar la forma más general posible para enfocar el problema.