Heurística y Optimización

Grado de Ingeniería en Informática. Curso 2022-2023

Práctica 1. Programación Lineal

Link al repositorio de GitHub de la práctica:
Práctica 1 - 100451170 - 10045158 - Repositorio de Github

Práctica realizada por los alumnos:
Carlos Iborra Llopis: 100451170@alumnos.uc3m.es
Miguel Domínguez Gómez: 100451258@alumnos.uc3m.es

1. Tabla de Contenidos

2. Introducción

La memoria de la práctica 1 de Heurística y optimización la hemos orientado de la siguiente manera:

Primero, explicaremos ambos modelos, el modelo de la parte 1 y el modelo de la parte 2, donde haremos hincapié en los datos, la elección de nuestras variables, nuestras restricciones y por último, en nuestra función objetivo.

Tras los modelos, tendrá lugar el análisis de los resultados obtenidos, dónde explicaremos que hemos decidido imprimir por pantalla en Mathprog, algunos comentarios acerca de los resultados que hemos obtenido tras cambiar los datos del problema, y también analizaremos dichos resultados para comprender cuáles de los datos son los que más afectan sobre nuestra función objetivo.

Por último, sacaremos unas conclusiones acerca de esta misma práctica dónde comentaremos si esta nos ha servido o no, y algunas sugerencias y/o críticas sobre el planteamiento de esta en connivencia con lo dado en clase.

3. Modelo Parte 1

Para empezar con la modelización, primero obtendremos los datos más importantes del enunciado.

Después un análisis del enunciado de la primera parte, destacamos lo siguiente:

- En el tablero hay **5 localizaciones distintas: 3 paradas, 1 parking, y 1 colegio**.
- En cada tramo se ha de indicar el número de estudiantes que pasan por ella (**flujo de estudiantes** que pasan por un **tramo**).
- Los arcos entre dos paradas indican los tramos para viajar entre localizaciones (km).
- Cada **autobús** puede **llevar como máximo a 20** estudiantes simultáneamente **(Restricción)**.
- Como máximo habrán **3 autobuses** en el tablero **(Restricción).**
- A cada **parada** solo puede **llegar una ruta (Restricción).**
- De cada **parada** solo puede **salir una ruta (Restricción).**
- Todas **las rutas salen del párking y llegan al colegio**, es decir, el número de rutas que salen del parking (que es igual al número de autobuses saliendo del parking) ha de ser igual al número de rutas que llegan al colegio. Con esto nos aseguramos de que todas las rutas terminan en el colegio **(Restricción).**
- El **flujo de alumnos** en un tramo **no puede superar la capacidad** máxima del **autobús (Restricción).**
- El **flujo de alumnos que salen de una parada** es la **suma del número de alumnos que entran** en esa parada **más el número de alumnos que esperaban** en la parada **(Restricción).**

3.1. Obtención de los datos

Tras identificar los distintos datos del enunciado, crearemos los siguientes parámetros y conjuntos:

1. Un conjunto que contiene las **distintas paradas existentes**, $paradas = \{S1, S2, S3\}$.

2. Un conjunto que contiene las **diferentes localizaciones** del mapa, $localizacion = \{P, S1, S2, S3, C\}$.

3. Un conjunto que contiene las **localizaciones origen** de donde parten los autobuses, $origen = \{P\}$.

4. Un conjunto que contiene las **localizaciones destino** donde llegan los autobuses, $destino = \{C\}$.

5. También crearemos conjunto resultante de la **combinación de todas las localizaciones** *"localizacion cross localizacion"* y lo llamaremos $localizaciones_{i,j}$.

Nota: no es necesario pero acortará las formalizaciones: $\forall i \in localizacion,\ j \in localizacion \Rightarrow \forall (i,j) \in localizaciones$

6. Un parámetro con el **número** máximo de **estudiantes** que puede transportar cada **autobús** (**capacidad)**, $bus\_capacidad = 20$.

7. Un parámetro que contiene el **coste fijo** de usar un **autobús**, $bus\_coste\_fijo = 120$.

8. Un parámetro con el **coste por kilómetro de los tramos recorridos** por los autobuses, $bus\_coste\_km = 5$.

9. Un parámetro que consiste en el **número máximo de autobuses** que se pueden usar, $bus\_numero\_max = 3$.

10. La **longitud de cada tramo** (en kilómetros): un parámetro que muestra la longitud que hay entre las distintas combinaciones de localizaciones (tramos), $longitud_{i,j}$.

i, j	P	S1	S2	S3	C
P	M	8	10	10	M
S1	M	M	3	7	6
S2	M	3	M	5	7
S3	M	7	5	M	4
C	M	M	M	M	M

Esta es su representación matricial, donde M es un valor arbitrariamente grande.

Este parámetro está formado por la combinación de todas las localizaciones al igual que ocurre con el parámetro $localizaciones$, obteniéndose los distintos tramos, con sus longitudes (en kilómetros).

11. Un parámetro representando el número de **estudiantes esperando en cada parada:** $estudiantes = \{15, 5, 10\}$

Donde $estudiantes_i$ representa el número de estudiantes que hay en la parada i (con $i \forall \{S1, S2, S3\}$)

3.2. Elección de las variables de decisión

De cara a la elección de las variables de decisión hemos de pensar que para delimitar la ruta de cada autobús necesitamos el recorrido de estos sobre el tablero (tramos por los que se pasa) y la cantidad de estudiantes que se transportan por estos tramos. Para poder solucionar esta incógnita, crearemos dos variables de decisión.

1. La primera variable de decisión está compuesta por la combinación de todas las localizaciones, dando lugar a los distintos tramos existentes en el tablero. Estos contendrán valores booleanos para indicar si: 1 (ha pasado un autobús por el tramo), o si por el contrario, 0 (no ha pasado ningún autobús por el tramo).

De esta forma, podremos delimitar la ruta que ha seguido cada uno de los autobuses con tan solo ir recorriendo los distintos tramos usados, ya que sabemos que todos los tramos empiezan en el parking y acaban en el colegio, y que además, ningún tramo se podrá utilizar más de una vez. Nótese cómo con este planteamiento, también seremos capaces de conocer la dirección que sigue el bus.

Cabe recalcar que puede que con este planteamiento se incremente el número de variables, pero tenemos que tener en cuenta durante todo el proceso de formalización es esta práctica que de cara al rendimiento, afecta mucho más el número de restricciones que el número de variables, por lo que este planteamiento sería aceptable.

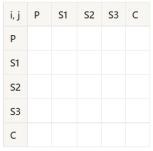
Por tanto, obtenemos la siguiente variable de decisión, $VISITADO_{i,j}$:

i, j	P	S1	S2	S3	C
P					
S1					
S2					
S3					
C					

Dónde el eje i, representa la localización inicial y el eje j, la localización a la que se llega, creando los diferentes tramos (ya que es la combinación de todas las posibles localizaciones en el tablero).
Este tablero se completará con el número de veces que se ha pasado por cada tramo, que al ser un valor booleano, como máximo será una vez ($VISITADO_{i,j} \in \{0, 1\}$).

2. La segunda variable de decisión, creada también a partir de la combinación de todas las distintas localizaciones y representa el **flujo de alumnos que han pasado por cada tramo**.
Esto es necesario ya que debemos tener un conocimiento de cuantas personas han pasado por los distintos tramos del mapa de cara a cumplir ciertas restricciones que se nos piden.

Por tanto, obtenemos la siguiente variable de decisión, $FLUJO_{ij}$:



Dónde al igual que con la primera variable de decisión, el eje i, representa la localización inicial y el eje j, la localización a la que se llega. Creando los diferentes tramos.
Este tablero se completará con el número de alumnos (entero) que han pasado por cada uno de los tramos ($FLUJO_{ij} \in N$).

3.3. Modelado de las restricciones

Tras la obtención de los datos y variables de decisión, ahora nos centraremos en las restricciones:

1. A cada parada llega una única ruta, $UnicaParada$:

$$\forall j \in paradas, \quad \sum_{i \in localizacion} VISITADO_{i,j} = 1$$

Ejemplo: $VISITADO_{P,S1} + VISITADO_{S1,S1} + VISITADO_{S2,S1} + VISITADO_{S3,S1} + VISITADO_{C,S1} = 1; 1 + 0 + 0 + 0 + 0 = 1;$

2. De cada parada sale una única ruta, $UnicaSalida$:

$$\forall i \in paradas, \quad \sum_{j \in localizacion} VISITADO_{i,j} = 1$$

Ejemplo: $VISITADO_{S1,P} + VISITADO_{S1,S1} + VISITADO_{S1,S2} + VISITADO_{S1,S3} + VISITADO_{S1,C} = 1; 0 + 0 + 1 + 0 + 0 = 1;$

3. El número de rutas no puede superar el número de autobuses disponibles, $RutasMenosAutobuses$:
Para esta restricción, podríamos usar tanto el número de salidas de P cómo el número de entradas en C para delimitar el número de rutas, ya que como restringiremos más tarde, tendrán el mismo valor.

$$\forall i \in origen, \quad \sum_{j \in localizacion} VISITADO_{i,j} \leq bus\_numero\_max$$

Ejemplo: $VISITADO_{P,P} + VISITADO_{P,S1} + VISITADO_{P,S2} + VISITADO_{P,S33} + VISITADO_{P,C} \leq 3;\ 0 + 1 + 0 + 1 + 0 \leq 3;$

4. Todas las rutas salen del parking y llegan al colegio. El número de rutas que salen del parking es igual al número de rutas que llegan al colegio, $ParkingColegio$:

$$\sum_{i \in origen, j \in localizacion} VISITADO_{i,j} - \sum_{i \in localizacion, j \in destino} VISITADO_{i,j} = 0$$

Ejemplo: $VISITADO_{P,P} + VISITADO_{P,S1} + VISITADO_{P,S2} + VISITADO_{P,S3} + VISITADO_{P,C} - (VISITADO_{P,C} + VISITADO_{S1,C} + VISITADO_{S2,C} + VISITADO_{S3,C} + VISITADO_{C,C}) = 0;\ 0 + 1 + 0 + 1 - (0 + 1 + 1 + 0) = 0;$

El flujo de alumnos desde X hasta Y no puede superar la capacidad máxima del autobús (en caso de que haya una ruta de X a Y). Con las variables de decisión que tenemos, deberemos acotar el problema mediante el uso de condicionales de tal forma que nos quedarían las siguientes dos restricciones (restricciones 5 y 6):

5. Si pasan ≤ 20 estudiantes, el número de veces que se ha podido pasar por ese tramo ($VISITADO$) puede tomar valor 0 o 1, $CapacidadAceptable$:

$$\forall (i,j) \in localizaciones,\ FLUJO_{i,j} - (bus\_capacidad \cdot VISITADO_{i,j}) \leq 0$$

Ejemplo: $FLUJO_{P1,P2} - (20 \cdot VISITADO_{P1,P2}) \leq 0;\ 15 - (20 \cdot 1) = -5 \leq 0;$

6. Si suben más de 20 estudiantes, $VISITADO$ debe tomar el valor 0 sí o sí (ya que excede la capacidad máxima del bus), $CapacidadExcedida$:

$$\forall (i,j) \in localizaciones,\ -M \cdot VISITADO_{i,j} + FLUJO_{i,j} \leq (bus\_capacidad + 1)$$

Nota: M es un valor arbitrariamente grande. **Ejemplo**: $-99999 \cdot VISITADO_{12} + FLUJO_{12} \leq 20 + 1;\ -9999 \cdot 1 + 15 \leq 21;$

7. El flujo de alumnos que sale de una parada debe ser exactamente el flujo de alumnos que entra en esa parada más el número de alumnos que esperan en ella, $FlujoEntradaSalida$:

$$\forall j \in paradas,\ \sum_{i \in localizacion} (FLUJO_{i,j} - FLUJO_{j,i}) + estudiantes_j = 0$$

Ejemplo: $(FLUJO_{11} - FLUJO_{11}) + (FLUJO_{21} - FLUJO_{12}) + (FLUJO_{31} - FLUJO_{13}) + estudiantes_0 = 0;\ (0-0) + (0-15) + (0-0) + 15 = 0;$

8. Todas las variables de decisión han de ser no negativas.
La variable VISITADOS debe de ser booleana y la variable FLUJO un número natural (al ser el flujo de alumnos).

$$VISITADO_{i,j} \in \{1,0\}\ \text{y}\ FLUJO_{i,j} \in N$$

Ejemplo: $VISITADO_{12} \in \{0,1\};\ 1 \in \{0,1\};$

3.4. Diseño de la función objetivo

Finalmente, diseñaremos la función objetivo que se nos indica en el enunciado con la finalidad de minimizar el coste. Este coste lo calcularemos como el número de los buses que se usan por el coste fijo de cada bus y a esto sumamos todos los kilómetros recorridos por el coste por kilómetro.

$$min \sum_{i \in origen,\ j \in localizacion} VISITADO_{i,j} \cdot bus\_coste\_fijo + bus\_coste\_km \cdot \sum_{(k,l) \in localizaciones} (VISITADO_{k,l} \cdot longitud_{k,l})$$

Ejemplo: $(1 \cdot 120 + 0 \cdot 120 + 1 \cdot 120) + (5 \cdot (1 \cdot 8 + 0 \cdot 10 + 1 \cdot 10 + 1 \cdot 3 + 0 \cdot 7 + 0 \cdot 6 + 0 \cdot 3 + 0 \cdot 5 + 1 \cdot 7 + 0 \cdot 7 + 0 \cdot 5 + 1 \cdot 4)) = 400$

4. Modelo Parte 2

Tras leer el enunciado de la parte 2, destacamos la siguiente información relevante:

- Si varios alumnos son **hermanos**, deben acudir siempre a la **misma parada**.
- La **distancia entre su casa y la parada** a la que vaya **no puede ser mayor** que una determinada (los alumnos podrían acudir a distintas paradas, pero solo van a una de ellas).
- El **número de alumnos** en una parada **no** puede **superar** la **capacidad** del autobús (**ya restringido**).

4.1. Obtención de los datos

En base a los datos que ya teníamos de la parte 1 y de los obtenidos en el nuevo enunciado de la parte 2, podremos crear los siguientes 3 nuevos datos (1 conjunto y 2 parámetros):

1. Un conjunto que contiene los **distintos alumnos,** $alumno = \{A1, A2, A3, A4, A5, A6, A7, A8\}$.

2. Un parámetro booleano con las posibles **paradas a las que puede ir** cada **estudiante**. A este parámetro lo llamaremos: $paradasAlumnos_{i,j}$

i, j	S1	S2	S3
A1	1	0	0
A2	1	0	0
A3	1	0	0
A4	1	1	1
A5	1	1	1
A6	0	0	1
A7	0	0	1
A8	0	0	1

Este parámetro es el resultado de la combinación de todas las paradas y alumnos (*"alumnos cross paradas"*), donde las columnas representan las distintas paradas y las filas pertenecen a cada uno de los estudiantes.

Si la celda tiene valor 1, el estudiante de esa fila tiene la posibilidad de acudir a la parada de esa columna; por el contrario, si es 0, el alumno no acudirá a esa parada.

3. El siguiente dato que hemos creado será otro parámetro booleano que muestra si **dos alumnos son hermanos** o no. A este parámetro lo llamaremos: $hermanos_{i,j}$

i, j	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	0	0	0	0	0	0	0
A2	0	0	0	0	0	0	0	0
A3	0	0	0	0	0	0	0	0
A4	0	0	0	0	1	0	0	0
A5	0	0	0	1	0	0	0	0
A6	0	0	0	0	0	0	0	0
A7	0	0	0	0	0	0	0	0
A8	0	0	0	0	0	0	0	0

Este es el resultado de la combinación de alumnos (*"alumnos cross alumnos"*), resultando en los distintos pares de alumnos posibles.
Si los alumnos son hermanos, el valor de la celdas que los relacionen será 1, en cambio, si no son hermanos, su valor será 0.
Nota: habrá dos celdas que relacionen a cada par de alumnos, ya que es una relación bidireccional (A4 es hermano de A5 y A5 es hermano de A4).

Además, de estos nuevos datos, eliminaremos el parámetro $estudiantes$ de la parte 1, ya que su función ahora la ejercen los nuevos datos y la variable de decisión como veremos a continuación.

4.2. Elección de las variables de decisión

1. Del enunciado de la parte 2, sacamos una nueva variable de decisión booleana que representa la **parada a la que va** cada uno de los **estudiantes**. A esta variable la llamaremos: $PARADA_{i,j}$.

i, j	S1	S2	S3
A1			
A2			
A3			
A4			
A5			
A6			
A7			
A8			

Al igual que el parámetro $paradasAlumnos$ de la parte 2, esta variable será una combinación de los distintos alumnos y paradas, donde las columnas representan las distintas paradas y las filas pertenecen a cada uno de los estudiantes.
Si una celda tiene valor 1, el estudiante acudirá a esa misma parada, en cambio, si tuviese valor 0, el estudiante no acudirá a dicha parada.

Esta nos indicará la parada final a la que se asignará cada estudiante. Como se puede ver, esta variable sustituye funcionalmente al parámetro $estudiantes$, como comentamos anteriormente.

4.3. Modelado de las restricciones

1. Un alumno debe acudir a solo una parada, $UnEstudianteUnaParada$.

$$\forall i \in alumnos, \quad \sum_{j \in paradas} PARADA_{i,j} = 1$$

Ejemplo: $PARADAS_{A1,S1} + PARADAS_{A1,S2} + PARADAS_{A1,S3} = 1; 1 + 0 + 0 = 1;$

2. Un estudiante solo puede ir a una parada que pueda alcanzar, $EstudiantesSoloParadaAlcanzable$.

$$\forall i \in alumnos, \sum_{j \in paradas} (PARADA_{i,j} \cdot paradasAlumnos_{i,j}) = 1$$

Ejemplo: $PARADAS_{A4,S1} \cdot paradasAlumnos_{A4,S1} + PARADAS_{A4,S2} \cdot paradasAlumnos_{A4,S2} + PARADAS_{A4,S3} \cdot paradasAlumnos_{A4,S3} = 1;\ 1 \cdot 0 + 1 \cdot 1 + 1 \cdot 0 = 1;$

3. Si dos estudiantes son hermanos, estos deben acudir a la misma parada, $RutasMenosAutobuses$.

$$\forall i \in alumnos, j \in alumnos, k \in paradas,\ hermanos_{i,j} \cdot (PARADA_{i,k} - PARADA_{j,k}) = 0$$

Ejemplo: $hermanos_{A4,A5} \cdot (PARADAS_{A4,S3} - PARADAS_{A5,S3}) = 0;\ 1 \cdot (1 - 1) = 0;$

4. Todas las variables de decisión han de ser no negativas, Par.

$$PARADA_{i,j} \in \{1, 0\}$$

Ejemplo: $PARADAS_{ij} \in \{0, 1\};\ 1 \in \{0, 1\};$

Aparte de las restricciones creadas, habremos de **modificar las restricciones de la parte 1** dependientes de los datos creados o modificados. Estas son las **restricciones 1, 2 y 7 de la parte 1**.

En la parte 2 los autobuses podrán ir o no a un parada, ya que al distribuirse los alumnos por las distintas paradas, puede que una de esas distribuciones deje alguna parada sin estudiantes, y por tanto no sería óptimo ir hacía dicha parada (Por ejemplo este caso se daría si en el problema de la parte 2, aumentásemos la capacidad de los autobuses a 5 estudiantes). Por tanto, debemos modificar las siguientes restricciones:

5. A cada parada llegan una o menos rutas, $UnicaParada$.

$$\forall j \in paradas, \sum_{i \in localizacion} VISITADO_{i,j} \leq 1$$

Ejemplo: $VISITADO_{P,S1} + VISITADO_{S1,S1} + VISITADO_{S2,S1} + VISITADO_{S3,S1} + VISITADO_{C,S1} \leq 1;\ 1 + 0 + 0 + 0 + 0 \leq 1;$

6. De cada parada salen una o menos rutas, $UnicaSalida$.

$$\forall i \in paradas, \sum_{j \in localizacion} VISITADO_{i,j} \leq 1$$

Ejemplo: $VISITADO_{S1,P} + VISITADO_{S1,S1} + VISITADO_{S1,S2} + VISITADO_{S1,S3} + VISITADO_{S1,C} \leq 1;\ 0 + 0 + 1 + 0 + 0 \leq 1;$

Además de la modificación de las dos anteriores restricciones, deberemos crear una nueva restricción que nos asegure que una parada sin entrada no pueda tener salida, por tanto:

7. El número de entradas a una parada debe de ser igual al número de salidas, $EntradaSalida$.

$$\forall i \in paradas, \sum_{j \in localizacion} (VISITADO_{i,j} - VISITADO_{j,i}) = 0$$

Ejemplo: $PARADAS_{A1,S1} + PARADAS_{A1,S2} + PARADAS_{A1,S3} = 1;\ 1 + 0 + 0 = 1;$

Para el caso de la restricción 7 de la parte 1, la cantidad de alumnos que ahora esperan en una parada de la parte 2 será el sumatorio de todos los alumnos que van a una parada j, indicado por la variable de decisión $PARADA_{i,j}$.

8. El flujo de alumnos que sale de una parada debe ser exactamente el flujo de alumnos que entra en esa parada más el número de alumnos que esperan en ella, $FlujoEntradaSalida$.

$$\forall j \in paradas, \sum_{i \in localizacion} (FLUJO_{i,j} - FLUJO_{j,i}) + \sum_{k \in alumnos} PARADA_{k,j} = 0$$

Ejemplo: $(FLUJO_{S1,S1} - FLUJO_{S1,S1}) + (FLUJO_{S2,S1} - FLUJO_{S1,S2}) + (FLUJO_{S3,S1} - FLUJO_{S1,S3}) + PARADAS_{A0,S1} + PARADAS_{A1,S1} + PARADAS_{A2,S1} + PARADAS_{A3,S1} + PARADAS_{A4,S1} + PARADAS_{A5,S1} + PARADAS_{A6,S1} + PARADAS_{A7,S1} = 0;$

El resto de datos, restricciones y variables de decisión de la parte 1 estarán incluidos en la parte 2 sin necesidad de ser modificadas, ya que no son dependientes de ninguna variable ni dato introducido o modificado en esta parte.

4.4. Diseño de la función objetivo

La función objetivo, al no contener datos ni variables directamente dependientes que hayan sido cambiados durante parte dos, no ha de ser modificada y será la igual a la definida en la parte 1.

$$min \sum_{i \in origen,\, j \in localizacion} VISITADO_{i,j} \cdot bus\_coste\_fijo + bus\_coste\_km \cdot \sum_{(k,l) \in localizaciones} (VISITADO_{k,l} \cdot longitud_{k,l})$$

Ejemplo: $(1 \cdot 120 + 0 \cdot 120 + 1 \cdot 120) + (5 \cdot (1 \cdot 8 + 0 \cdot 10 + 1 \cdot 10 + 1 \cdot 3 + 0 \cdot 7 + 0 \cdot 6 + 0 \cdot 3 + 0 \cdot 5 + 1 \cdot 7 + 0 \cdot 7 + 0 \cdot 5 + 1 \cdot 4)) = 400$

5. Análisis de los Resultados

De cara al análisis de los resultados, lo más sencillo será ejecutar los distintos problemas en Mathprog, donde imprimimos por terminal los datos más importantes (que se corresponden con los valores no nulos de nuestras variables de decisión) con la finalidad de analizar de manera más sencilla y eficaz dichos resultados.
Es por esto que nuestra terminal imprimirá los siguientes datos:

1. **Los distintos tramos por dónde han pasado los buses**, a través de los cuales se pueden reconstruir las distintas rutas seguidas.
2. **La cantidad de personas que se transportan por los distintos tramos** (de una localización a otra).
3. Además, en la parte 2 se imprimirá por terminal **a que parada se decide moverse cada estudiante**.

A partir de estos resultados, analizaremos tanto para la parte 1 como para la parte 2 la complejidad de cada problema, las restricciones más limitantes de cara al resultado de la función objetivo, crearemos un nuevo problema en base a la modificación de los datos del enunciado, y por último, discutiremos las ventajas y desventajas de LibreOffice Calc y GLPK.

5.1. Parte 1

La ejecución de la parte 1 según los datos iniciales del enunciado nos daría el siguiente resultado:

```
Resolución del modelo...

Coste óptimo: 400

Tramos visitados:
Se ha pasado por el tramo P-S1
Se ha pasado por el tramo P-S3
Se ha pasado por el tramo S1-S2
Se ha pasado por el tramo S2-C
Se ha pasado por el tramo S3-C

Flujo de estudiantes en cada tramo:
Se han transportado 15 estudiantes de S1 a S2
Se han transportado 20 estudiantes de S2 a C
Se han transportado 10 estudiantes de S3 a C

Model has been successfully processed
Writing MIP solution to 'solucion.txt'...
```

El **coste óptimo** sería de **400** (€)
Las **rutas** seguidas serían: **P-S1, S1-S2, S2-C** y **P-S3, S3-C**.

Si reproducimos las rutas que sigue cada autobús y nos fijamos en el flujo de estudiantes en cada tramos podremos observar como no se incumple ninguna de las restricciones definidas en ningún momento.

Por tanto, podemos observar como solo hay **una entrada y salida** por cada localización; hay **menos rutas que** número máximo de **autobuses** (3); hay un **mismo número** de **salidas del parking** y **entradas en el colegio**; para todas las rutas habrán **exactamente 20 o menos de 20 estudiantes** (que es la capacidad de los buses en este caso) y por último, el **flujo de salida** de una parada es la **suma del flujo de estudiantes que entra** en esa misma parada **más el número de estudiantes que esperan** en ella (cómo se puede observar en los datos de flujo de S1 a S2 y de S2 a C, dónde el flujo pasa de 15 a 20, ya que el bus ha recogido a los 5 estudiantes esperando en la parada S2).

5.1.1. Complejidad del problema

Para la parte 1, nuestro problema consta de **2 variables de decisión** ($VISITADO$ y $FLUJO$), y de **7 restricciones**, una por cada restricción identificada en el enunciado dado en la parte 1.

Estas variables y restricciones ya las definimos en la parte 3.2. Elección de las variables de decisión y 3.3. Modelado de las restricciones.

5.1.2. Restricciones limitantes

Las restricciones que más limitan el problema son las siguientes:

- $UnicaParada$ y $UnicaSalida$, ya que fuerzan a que solo se pueda pasar por una parada una sola vez, si se eliminan dichas restricciones del problema, los buses podrán pasar más de una vez por la misma parada. En este caso, la restricción limitante para este mismo problema será $UnicaSalida$, ya que tras probar la ejecución eliminando cada una de estas restricciones, el resultado solo se vería afectado con la eliminación de $UnicaSalida$.
- $ParkingColegio$: hace que todas las rutas empiecen en el parking y acaben en el colegio, sin esta restricción, las rutas podrían empezar y terminar en cualquier lado. Esto nos haría imposible el identificar cada ruta y haría que la solución óptima careciese de sentido alguno.
- $CapacidadAceptable$ y $CapacidadExcedida$: limitan la capacidad de recoger estudiantes del autobús a máximo 20 estudiantes lo que restringe el problema de manera enorme, ya que sin estas restricciones (que se complementan), un solo autobús podría recoger infinitos estudiantes. Por tanto limita en gran medida este problema.
- $FlujoEntradaSalida$: esta restricción nos permite recrear el flujo de alumnos, por lo que si ejecutamos el problema sin esta restricción, el flujo de alumnos dejaría de existir, tomando el valor 0 para todos los tramos, con la consecuencia de que un autobús podría recoger a todos los alumnos independientemente de su capacidad (ya que el flujo habría dejado de existir), lo que convierte a esta restricción en limitante y muy necesaria, debido a que sin esta restricción (al eliminar el flujo de estudiantes), no habría necesidad de usar autobuses, ni se necesitaría salir por el parking ni tampoco se necesitaría llegar al colegio.
- $RutasMenosAutobuses$: esta es la menos limitante de las 7 restricciones, ya que para poder afectar al problema, se tiene que cumplir que el parámetro bus_numero_max sea un número igual al número de rutas óptimas en el problema, y solo si ocurriese esto, el reducir el número de autobuses que se puedan usar tendrá efecto. Nota: si reducimos en demasía el número de autobuses, por ejemplo $bus\_numero\_max = 1$ (para el problema del enunciado), este dejaría de tener solución, ya que no hay capacidad suficiente en un autobús para poder recoger a todos los estudiantes.

5.1.3. Nuevo problema

Para la parte 1, crearemos un nuevo problema a través de la modificación de todos los datos disponibles (ya que no disponemos de espacio suficiente en la memoria para poder comentar uno a uno lo que sucede con el cambio de un solo dato en problemas distintos), además, para añadir complejidad al problema y probar la versatilidad nuestra solución, también crearemos una nueva parada con sus estudiantes y también haremos que no todas las paradas estén conectadas entre ellas para añadir complejidad al problema.

```
data;

set localizacion := P S1 S2 S3 S4 C;
set paradas := S1 S2 S3 S4;
set origen := P;
set destino := C;


param mGrande := 999999;
param bus_capacidad := 15;
param bus_coste_fijo := 100;
param bus_coste_km := 3;
param bus_numero_max := 6;

param longitud:
     P       S1       S2       S3       S4       C:=
  P  999999   8        10       10       10       999999
  S1 999999   999999   3        7        7        6
  S2 999999   3        999999   5        5        7
  S3 999999   7        5        999999   999999   4
  S4 999999   7        5        999999   999999   4
  C  999999   999999   999999   999999   999999   999999
  ;

param  estudiantes :=
  S1 15
  S2 5
  S3 10
  S4 3
  ;


end;
```

Para este nuevo problema, el **coste óptimo** será de **441** (€)

Se recorren las rutas: **P-S1,S1-C**; **P-S2,S2-S3,S3-C**; **P-S4,S4-C**.
Con un flujo de 15 alumnos de S1 a C, 5 de S2 a S3, 15 de S3 a C y 3 de S4 a C.

Cómo podemos observar de los resultados obtenidos, pese a añadir una nueva parada y cambiar todos los datos de este problema, las restricciones han seguido surtiendo efecto y llevándonos a una solución óptima que cumple todas estas.
Esto muestra la versatilidad de nuestra aproximación.
Nota: el archivo con los datos de este problema está en la carpeta *1.mathprog/test* en el repositorio de Github.

También creemos pertinente comentar que en comparación con el problema base de la parte 1 (más simple), no se han detectado grandes cambios en el tiempo usado ni en memoria usada para la resolución.
Tiempo usado: problema base: 0.0s, problema nuevo: 0.0s
Memoria usada: problema base: 0.3Mb, problema nuevo: 0.3Mb

5.2. Parte 2

La ejecución de la parte 1 según los datos iniciales del enunciado nos daría el siguiente resultado:

```
Resolución del modelo...

Coste óptimo: 585

Tramos visitados:
Se ha pasado por el tramo P-S1
Se ha pasado por el tramo P-S2
Se ha pasado por el tramo P-S3
Se ha pasado por el tramo S1-C
Se ha pasado por el tramo S2-C
Se ha pasado por el tramo S3-C

Flujo de estudiantes en cada tramo:
Se han transportado 3 estudiantes de S1 a C
Se han transportado 2 estudiantes de S2 a C
Se han transportado 3 estudiantes de S3 a C

Paradas a las que va cada estudiante:
El estudiante A1 va a la parada S1
El estudiante A2 va a la parada S1
El estudiante A3 va a la parada S1
El estudiante A4 va a la parada S2
El estudiante A5 va a la parada S2
El estudiante A6 va a la parada S3
El estudiante A7 va a la parada S3
El estudiante A8 va a la parada S3
```

El **coste óptimo** para este problema es de **585** (€)
Las **rutas** seguidas serían: **P-S1, S1-C**; **P-S2, S2-C**; **P-S3, S3-C**.

Si nos fijamos en las rutas que sigue cada autobús, en el flujo de estudiantes en cada tramo y en las paradas a las que acude cada estudiante, observamos que no hay ninguna restricción que se incumpla.

Ahora, no solo nos fijaremos en las nuevas restricciones añadidas en la parte dos y las que se han tenido que modificar de la parte 1 (ya que el resto de restricciones se mantienen iguales y limitan el resultado del problema de la misma manera que en lo limitaban antes); por tanto, tenemos que se cumplen lo siguiente:

debe **haber o no una entrada entrada y salida** por cada localización (ya que ahora puede que existan paradas a las que no vaya ningún alumno) y que toda **parada que tenga una entrada**, también **tenga una salida**, y al contrario, es decir, que toda parada con una salida, debe tener también una entrada. El **flujo de estudiantes que sale** de una parada es la **suma del flujo que entra en esa misma parada más el número de estudiantes que esperaban en ella** (que se calculará ahora con el sumatorio de los alumnos que acuden a una parada, representado en la variable de decisión PARADA). Continuando con las restricciones añadidas en la parte 2, comprobamos que cada **alumno sólo irá a una parada** y que dicha parada es una de las que tienen **a su alcance**, por último también comprobamos que los estudiantes que sean **hermanos acuden a la misma parada**.

5.2.1. Complejidad del problema

Para la parte 2, nuestro problema consta de **3 variables de decisión**: las dos variables de decisión heredadas de la parte 1 más la nueva variable $PARADA$, y de **11 restricciones**: las 7 restricciones de la parte 1 con sus correspondientes modificaciones más las 4 nuevas restricciones identificadas en el enunciado de la parte 2 ($EntradaSalida$, $UnEstudianteUnaParada$, $EstudiantesSoloParadaAlcanzable$ y $RutasMenosAutobuses$).

Todas estas restricciones y variables ya las hemos comentado en la parte 4.2. Elección de las variables de decisión, 4.3. Modelado de las restricciones.

5.2.2. Restricciones limitantes

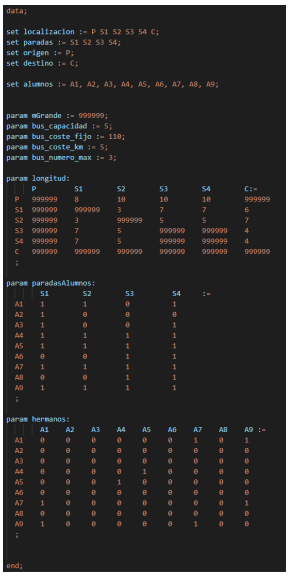
De la misma forma que hemos comentado en el apartado 5.2. Parte 2, solo haremos hincapié en las restricciones modificadas o añadidas durante la formalización de la parte 2, ya que las restricciones ya comentadas de la parte 1 que no han sido modificadas, serán igual de limitantes para este nuevo problema propuesto (esto se ha probado a parte pero debido las páginas limitadas de la memoria y de lo repetitivo que resultaría, se ha decidido no incluirlo).

Dicho esto, las restricciones nuevas que limitan el problema son las siguientes:

- $UnicaParada$ y $UnicaSalida$: de manera parecida a como afectan al problema anterior, estas acotan bastante el problema (aunque de manera menos limitante que en la parte 1, debido a que ahora pueden tomar tanto valor el valor 0 como el valor 1), ya que si se suprimieran dichas restricciones, los autobuses podrían ir más de una vez a una parada y como podemos intuir, esto afectaría mucho al resultado. Por ejemplo, para este mismo problema, saldrán y llegarán dos buses a S2 cuando no debería, lo que elimina el coste de usar un tercer bus y por tanto nos desbarata el resultado.
- $EntradaSalida$: esta restricción es necesaria ya que como acabamos de comentar, los buses pueden decidir si ir o no a una parada (debido a que los estudiantes hayan dejado libre una parada, lo que haría que pasar por esta parada vacía careciese de sentido, resultando en un resultado no óptimo). Es por esto que si ejecutamos el problema sin esta restricción, ahora los autobuses podrían entrar en una parada sin necesidad de salir y viceversa, lo que quiebra el sentido del ejercicio y su resultado, por lo que podríamos calificar a esta restricción como limitante.
- $FlujoEntradaSalida$: esta restricción nos permite recrear el flujo de alumnos por cada tramo, por lo que si ejecutamos el problema sin esta restricción, el flujo de alumnos dejaría de existir, tomando el valor 0 para todos los tramos y haciendo que la variable FLUJO perdiese todo el sentido. Esta sería una de las más limitantes, ya que si se decidiese eliminar esta restricción, el coste óptimo para cualquier problema sería 0.
- $UnEstudianteUnaParada$: esta restringe que un estudiante solo pueda ir a una parada, pero al contrario de lo que pueda parecer a priori, si eliminaremos dicha restricción, la resolución para este problema no se vería afectada ya que todos los estudiantes a excepción de A4 y A5 tienen solo una parada a su alcance, además, si A4 y A5, al ser hermanos están limitados a solo poder acudir a S2 ya que es la única en la que no excederán la capacidad del bus. Por tanto, *para este problema en concreto*, no sería limitante.
- $EstudiantesSoloParadaAlcanzable$: como hemos comentado en la restricción anterior, esta restricción acota la anterior restricción para este problema ya que obliga a los estudiantes a ir solo a las paradas que estén a su alcance; y como en este problema, a excepción de los hermanos, los estudiantes solo tienen una parada a su alcance, estos solo podrán acudir a esa misma parada, lo que limita en gran medida el problema.
- $UnaMismaParadaHermanos$: esta restricción asegura que los hermanos siempre vayan juntos a una parada, por tanto, ésta afectaría de gran manera a este problema ya que al no necesitar ir A4 y A5 juntos, estos podrían repartirse uno en S1 y el otro en S3, bajando el coste del problema (400€), y reduciendo el número de paradas y rutas significativamente (ahora solo harían falta dos autobuses), por lo que podemos comprobar que esta restricción también sería bastante limitante para este problema.

5.2.3. Nuevo problema

Para la parte 2, de forma parecida a lo que hicimos en 5.1.3. Nuevo problema, modificaremos todos los datos que tenemos del problema base, y no solo eso, sino que también añadiremos una parada nueva (S4) y un estudiante nuevo. Pero eso no es todo, ya que para terminar de hacer todavía más compleja la resolución de este problema, a los 2 hermanos (A4 y A5) heredados del problema base, añadiremos otros 3 hermanos nuevos (A1, A7 y A9).



```
data;

set localizacion := P S1 S2 S3 S4 C;
set paradas := S1 S2 S3 S4;
set origen := P;
set destino := C;

set alumnos := A1, A2, A3, A4, A5, A6, A7, A8, A9;


param mGrande := 999999;
param bus_capacidad := 5;
param bus_coste_fijo := 110;
param bus_coste_km := 5;
param bus_numero_max := 3;

param longitud:
      P       S1      S2      S3      S4      C:=
  P   999999  8       10      10      10      999999
  S1  999999  999999  3       7       7       6
  S2  999999  3       999999  5       5       7
  S3  999999  7       5       999999  999999  4
  S4  999999  7       5       999999  999999  4
  C   999999  999999  999999  999999  999999  999999
  ;

param paradasAlumnos:
      S1      S2      S3      S4    :=
  A1  1       1       0       1
  A2  1       0       0       0
  A3  1       0       0       1
  A4  1       1       1       1
  A5  1       1       1       1
  A6  0       0       1       1
  A7  1       1       1       1
  A8  0       0       1       1
  A9  1       1       1       1
  ;

param hermanos:
      A1  A2  A3  A4  A5  A6  A7  A8  A9 :=
  A1  0   0   0   0   0   0   1   0   1
  A2  0   0   0   0   0   0   0   0   0
  A3  0   0   0   0   0   0   0   0   0
  A4  0   0   0   0   1   0   0   0   0
  A5  0   0   0   1   0   0   0   0   0
  A6  0   0   0   0   0   0   0   0   0
  A7  1   0   0   0   0   0   0   0   1
  A8  0   0   0   0   0   0   0   0   0
  A9  1   0   0   0   0   0   1   0   0
  ;

end;
```

Para este nuevo problema, el **coste óptimo** será de **360** (€).

Se recorren las rutas: **P-S1,S1-C**; **P-S4,S4-C**.
Con un flujo de 4 estudiantes de S1 a C y de 5 de S4 a C.

Cómo podemos observar de los resultados obtenidos, pese a añadirse una nueva parada, otro alumnos, tres nuevos hermanos y cambiar todos los datos de este problema, las restricciones han seguido surtiendo efecto y llevándonos a una solución óptima que cumple todas estas restricciones.

Esto nos muestra la versatilidad de nuestra aproximación y cómo podemos escalar casi indefinidamente el tamaño de nuestro tablero, añadiendo nuevos datos (localizaciones, estudiantes, buses…) sin que esto termine en un error debido a nuestro planteamiento, ya que hemos sido cuidadosos con diseñar una solución lo suficientemente abierta y tolerable a modificaciones que se puedan llegar a hacer en un futuro (como se ha podido comprobar en este nuevo problema).

Nota: para tener una vista más detallada de los datos usados, el archivo con los datos de este problema está situado en la carpeta *2.mathprog/test* en el repositorio de Github.

Al igual que con el nuevo problema de la parte 2, creemos pertinente comentar que en comparación con el problema base de la parte 2 (mucho más simple que nuestro nuevo problema), no se han detectado grandes cambios en el tiempo usado para resolver el problema pero si en memoria usada para la resolución (algo normal teniendo en cuenta que hemos aumentado bastante la cantidad de datos).
Tiempo usado: problema base: 0.0s, problema nuevo: 0.0s
Memoria usada: problema base: 0.4Mb, problema nuevo: 0.6Mb

5.3. Ventajas y desventajas: LibreOffice y GLPK

5.3.1. LibreOffice Calc

De LibreOffice en sí, hay poco que comentar, una de las desventajas sería que para poder usar Calc, debemos descargarnos todas las herramientas LibreOffice, lo que lo convierte en un pequeño inconveniente si se dispone de poco almacenamiento. Otro pequeño inconveniente es la necesidad de tener Java instalado en nuestras máquinas para poder usar todo el potencial de LibreOffice.
Dentro del propio LibreOffice Calc, el único inconveniente que nos ha surgido ha sido que en uno de nuestros ordenadores a veces daba errores en el color del tema y se volvía todo negro, lo que lo hacía imposible de utilizar.
Por otro lado, ofrece muchas ventajas, ya que al parecerse a Microsoft Excel tanto en diseño como en herramientas y usabilidad nos resultó mucho más fácil el familiarizarnos con él. Otra ventaja sería la sencillez con la que se usa el solver, lo que facilita mucho el trabajo de cara a probar las soluciones óptimas obtenidas.

5.3.2. GLPK

Uno de los mayores inconvenientes de usar GLPK es que está diseñado para Linux, por lo que obviamente será mucho más sencillo su uso en distribuciones de Linux. Es por eso que decidimos utilizar Ubuntu para poder ejecutar por terminal GLPK - Mathprog.

Por otro lado, se necesita utilizar los editores de texto VIM o NANO para abrir y modificar los ficheros, lo que frena mucho el desarrollo de la práctica, por eso lo que decidimos al final fue programar el model.mod y el data.dat en Visual Studio Code y luego pegarlos en el editor NANO, para luego guardarlos y ejecutarlos. Así pudimos acelerar drásticamente el proceso de programación y escritura.
Las ventajas de utilizar GLPK son básicamente la velocidad de ejecución, los comentarios que devuelve por terminal acerca de los errores que han podido suceder (especificando la línea y el error, lo cuál ha sido de gran ayuda) y el archivo solucion.txt que nos devuelve con todos los datos de la *solución MIP*.

6. Conclusiones

Esta práctica nos ha sido muy útil a la hora de asentar los conocimientos adquiridos durante el tema de *"Programación Lineal"* estudiado en clase. Esto se debe a que nos ha tocado resolver un problema que bien podría ser real, con datos y restricciones que podrían ser reales.

Es por esto que nos ha sido muy útil para asentar los conocimientos, ya que al poder imaginar el problema como real, nos ayuda a comprender de forma realista con usos para la vida real lo dado en clase, lo que particularmente a nosotros, nos ha ayudado mucho de cara a la compresión del temario.

Pese a esto, sí que hemos notado cierta dificultad para empezar con la parte 1, que a nuestro parecer es la más importante, ya que es el modelo base sobre el que se sustentan tanto la parte 2 como el análisis de resultados (parte 3).

Una de las causas de esta dificultad es que hemos tenido que esforzarnos mucho para poder sacar las distintas restricciones de la parte 1. Lo peor de todo es que un par de semanas más tarde en la clase magistral nos enseñaron los problemas de transporte, de asignación y los condicionales, lo cuál nos hubiese ahorrado mucho tiempo y esfuerzo de haberlo visto antes en clase.

A causa de esto, comparativamente, de las 4 semanas que hemos dedicado a este proyecto, 3 de ellas han sido solo para la parte 1 y la semana restante para las partes 2 y 3, ya que una vez obtenida y formulada correctamente la parte 1, hacer el resto de partes ha sido muy directo.

Otra pequeña causa de la dificultad ha sido a la hora de comprender bien cómo crear matrices, vectores, conjuntos, etcétera, en Mathprog. Nos hubiese gustado disponer de más ejemplos de clase, ya que hemos notado como los archivos adjuntos con sus ejemplos eran pobres respecto a lo que se nos pedía en esta práctica y carecían de ejemplos, datos y restricciones suficientes y comparables.
Esto nos ha obligado a tener que orientarnos usando otros archivos y documentos de internet para poder comprender bien el uso de mathprog.

De la misma forma, en la parte 1, encontramos demasiado sencillos los ejemplos proporcionados en el Aula Global, siendo estos de mucha menor complejidad de lo que se nos proponía en este ejercicio. Debido a que eran tan básicos, no nos fueron de mucha utilidad de cara a facilitarnos el comienzo de la práctica en la parte 1, la cual, como hemos comentado anteriormente es la que más tiempo nos ha consumido de todas.

En resumen, esta práctica, pese a la extrema dificultad que hemos tenido para poder empezar, nos ha ayudado gratamente de cara a comprender mejor lo que hacemos en programación lineal y ahora sí, tras completar la práctica, podemos decir que sabemos lo que es la programación lineal y nos vemos capacitados para poder enfrentar este tipo de problemas en el exámen final.