

# Práctica 1- Programación lineal

---



## Heurística y optimización- Curso 25/26

---

Nombre, apellidos y NIA
Alumn@ 1: Ana Grima Vázquez de Prada (100495785) 100495785@alumnos.uc3m.es Grupo 80 - Grado en Ingeniería Informática
Alumn@ 2: Alejandra de los Santos Blanco (100495843) 100495843@alumnos.uc3m.es Grupo 80- Grado en Ingeniería Informática

# Índice

---

<b>Introducción.....</b>	<b>3</b>
<b>Parte 1: Modelo básico en Calc.....</b>	<b>3</b>
1.1 Modelización del problema.....	3
1.1.1 Conjuntos.....	3
1.1.2 Parámetros.....	3
1.1.3 Variables de decisión.....	4
1.1.4 Función objetivo.....	4
1.1.5 Restricciones.....	4
1.1.6 Modelización.....	5
1.2 Implementación del modelo.....	5
<b>Parte 2: Modelo avanzado en GLPK.....</b>	<b>6</b>
2.1 Minimización del impacto de las averías.....	6
2.1.1 Conjuntos.....	6
2.1.2 Parámetros.....	7
2.1.3 Variables de decisión.....	7
2.1.4 Función objetivo.....	7
2.1.5 Restricciones.....	7
2.1.6 Modelización.....	8
2.1.7 Implementación del modelo general.....	8
2.1.8 Script para generar ficheros de datos.....	8
2.2 Maximización de la satisfacción de los pasajeros.....	8
2.2.1 Conjuntos.....	8
2.2.2 Parámetros.....	9
2.2.3 Variables de decisión.....	9
2.2.4 Función objetivo.....	9
2.2.5 Restricciones.....	9
2.2.6 Modelización.....	10
2.2.7 Implementación del modelo general.....	10
2.2.8 Script para generar ficheros de datos.....	11
<b>Parte 3: Análisis y pruebas.....</b>	<b>11</b>
3.1 Análisis de resultados.....	11
3.2 Casos de pruebas.....	13
3.2.1 Pruebas para parte-2-1.mod.....	13
3.2.2 Pruebas para parte-2-2.mod.....	13
<b>Conclusiones de la práctica.....</b>	<b>14</b>

# Introducción

En esta práctica abordamos la resolución de problemas de programación lineal entera aplicados a la optimización en el contexto de una empresa de transporte por autobuses. El objetivo principal es diseñar modelos que permitan minimizar los costes totales de atención y planificación de los autobuses, considerando tanto la distancia al taller como el número de pasajeros afectados. Para ello utilizamos LibreOffice, GLPK y un script en Python para generar y resolver los datos de forma automática.

La práctica se divide en tres partes principales. En la primera, formulamos un modelo básico con un solo taller para comprender la estructura general del problema y validar la función objetivo. En la segunda, ampliamos el modelo introduciendo franjas horarias, lo que permite optimizar la asignación de autobuses teniendo en cuenta la capacidad temporal del taller. Finalmente, en la tercera parte, incorporamos varios talleres y analizamos la coincidencia de pasajeros entre autobuses, buscando minimizar los solapamientos y mejorar la eficiencia global del sistema.

En esta memoria explicamos cómo hemos desarrollado y resuelto cada modelo, los resultados obtenidos en las distintas pruebas y las conclusiones que extraemos sobre la utilidad de las herramientas empleadas y de las técnicas de optimización aplicadas a problemas reales de planificación.

## Parte 1: Modelo básico en Calc

En la primera parte de la práctica, el problema consiste en asignar autobuses a las franjas disponibles de un taller de mantenimiento, con el objetivo de minimizar el coste total de desplazamientos. Cada autobús se encuentra a una distancia determinada del taller, y cada franja puede atender únicamente a un autobús. El coste asociado a cada autobús depende directamente de la distancia que debe recorrer para ser atendido, por lo que la asignación óptima debe realizarse de forma que se reduzca al mínimo la suma total de estas distancias.

### 1.1 Modelización del problema

El problema se plantea como un modelo de programación lineal entera, en el que se busca minimizar el coste total de asignación de autobuses a las franjas disponibles en el taller. A continuación, se define la notación empleada para modelar formalmente el problema, incluyendo conjuntos, parámetros, variables de decisión, función objetivo y restricciones correspondientes.

#### 1.1.1 Conjuntos

Los conjuntos que creamos separan los datos de decisiones y nos permiten extender el modelo si cambian de tamaño.

$A$ : autobuses  $\rightarrow \{A1, A2, A3, A4, A5\}$  (Autobús 1, Autobús 2, Autobús 3, Autobús 4, Autobús 5)

$T$ : talleres  $\rightarrow \{T1, T2, T3, T4, T5\}$  (Taller 1, Taller 2, Taller 3, Taller 4, Taller 5)

#### 1.1.2 Parámetros

Denominamos los elementos de los conjuntos "Autobuses" y "Talleres" como " $a$ " y " $t$ ". En donde " $a$ " representa cada autobús y " $t$ " cada taller.

- $a$ :  $\forall a \in \text{Autobuses}$ , tomará valores de 1-5
- $t$ :  $\forall t \in \text{Talleres}$ , tomará valores de 1-5

Representamos el coste de la distancia (km) entre el autobús " $a$ " y el taller " $t$ " como:

$$Coste = c_{t,a} \in R \geq 0$$

A continuación, mostramos el valor que toma dicha variable para cada taller con cada autobús:

	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$
$t_1$	206	66	263	104	154
$t_2$	285	86	111	300	110
$t_3$	164	251	109	220	177
$t_4$	264	175	217	147	149
$t_5$	141	75	218	87	228

### 1.1.3 Variables de decisión

La variable de decisión “x” indica si un autobús se asigna o no a un determinado taller.

$$Asignado = x_{t,a}$$

$$x_{t,a} \in \{0, 1\} \forall t \in T, \forall a \in A$$

Semántica:

- 1 si el autobús  $a$  se asigna al taller  $t$
- 0 en caso contrario

### 1.1.4 Función objetivo

El objetivo es minimizar la distancia total recorrida por todos los autobuses al desplazarse hasta los talleres. Para conseguirlo, multiplicamos el coste de la distancia recorrida entre cada taller y autobús por la variable de decisión “x”. De este modo, solo sumamos las distancias correspondientes a las asignaciones realizadas ( $x_{t,a} = 1$ ). Finalmente, calculamos la suma de todas esas distancias para obtener la distancia total recorrida más pequeña posible.

$$\min z = \sum_{a=1}^5 \sum_{t=1}^5 c_{t,a} \cdot x_{t,a}$$

### 1.1.5 Restricciones

1. Restricción de asignaciones: cada autobús debe asignarse exactamente a un taller, la suma de asignaciones de un taller para cada bus debe ser exactamente uno.

$$\sum_{t=1}^5 x_{t,a} = 1, \forall a \in A$$

2. Restricción capacidad de talleres: cada taller puede tener como máximo un bus asignado, la suma de asignaciones de un bus para cada taller debe ser como máximo 1.

Con esta restricción evitamos asignar más de un autobús al mismo taller. Se modela  $\leq 1$  porque el enunciado dice “no más de uno”; en el óptimo habrá igualdad.

$$\sum_{a=1}^5 x_{t,a} \leq 1, \forall t \in T$$

### 1.1.6 Modelización

Por lo tanto, la modelización de nuestro problema en la Parte 1 para que la pérdida sea mínima será:

$$\begin{aligned} \min z &= \sum_{a=1}^5 \sum_{t=1}^5 c_{t,a} \cdot x_{t,a} \\ \text{s.a } &\begin{cases} 1. \sum_{t=1}^5 x_{t,a} = 1, \forall a \in A \\ 2. \sum_{a=1}^5 x_{t,a} \leq 1, \forall t \in T \end{cases} \\ &x_{t,a} \in \{0, 1\}, \forall a \in A, \forall t \in T \\ &a, t \geq 0 \end{aligned}$$

### 1.2 Implementación del modelo

La implementación se realiza en la hoja de cálculo “part-1.ods” de LibreOffice. Al comienzo de este fichero se puede observar:

- Tabla con todos las distancias proporcionados en el enunciado, que muestra en kilómetros la distancia de cada autobús a cada taller,  $c_{t,a}$ .
- Tabla donde se incluyen las asignaciones de cada taller a cada bus,  $x_{t,a}$ , donde; 1 asignado, 0 no asignado. Estos datos han sido calculados con la herramienta “Solver” explicada más adelante.  
Tanto en esta tabla como en la anterior, las filas corresponden a talleres y las columnas a autobuses.
- Celda “Función objetivo” en donde se muestra el valor total de la función objetivo. Se usa la función “=SUMA.PRODUCTO(C9:G13; L9:P13)”, que multiplica las distancias por las variables de asignación y suma los resultados.
- Dos tablas de restricciones separadas por:
  - Restricción 1: restricción de igualdad. En cada fila, una por autobús, se calcula la suma de las celdas correspondientes a dicho autobús en la matriz de asignaciones. Para ello se usa la función “SUMA”, por ejemplo, autobús uno (A1): “=SUMA(L9:L13)”.  
El valor al que se tienen que igualar estas restricciones está indicado en la columna “Parámetro 1”, en este caso 1.
  - Restricción 2: restricción menor o igual. En cada fila, una por taller, se calcula la suma de las asignaciones a dicho taller en la matriz de asignaciones. Para ello se usa la función “SUMA”, por ejemplo, taller uno (T1): “=SUMA(L9:P9)”.  
El valor máximo permitido se indica en la columna “Parámetro 2”, en este caso 1.

Los valores de los parámetros se indican en celdas separadas para facilitar cualquier modificación del modelo, de modo que las restricciones pueden ajustarse simplemente cambiando estos valores.

A continuación, se muestra la configuración de la herramienta “Solver”:

Al tratarse de un problema lineal entero, se emplea el algoritmo “Solver lineal de LibreOffice”, y se establecen las variables como enteras y no negativas en la configuración de opciones de la herramienta.

Se define la celda objetivo “\$N\$17”, que almacena cómo se ha dicho antes el valor resultante de la función objetivo. Se establece la optimización de resultados a mínimo, puesto que es lo que se busca en el problema e indicamos la combinación de celdas que debe hacer la herramienta: “\$L\$9:\$P\$13”, la matriz de asignaciones. A continuación, se establecen las condiciones limitantes:

- “\$C\$24:\$C\$28 = \$F\$24”, la parte izquierda representa los valores de la celda C-24 hasta la celda C-28, es decir, la columna de la tabla “Restricción 1”. La parte derecha representa el valor de la celda F-24, el “Parámetro 1”.
- “\$J\$24:\$J\$28 ≤ \$M\$24”, la parte izquierda representa los valores de la celda J-24 hasta la celda J-28, es decir, la columna de la tabla “Restricción 2”. La parte derecha representa el valor de la celda M-24, el “Parámetro 2”.
- “solver\_opt” establecido con operador binario para indicar que las variables de asignación sólo pueden tomar valores 0 o 1.

Tras ejecutar el Solver, se muestra en la hoja de cálculo los valores óptimos de las variables de asignación  $x_{t,a}$ , en la matriz de asignaciones y el valor mínimo de la función objetivo  $z$  en la celda N-17, en este caso **573**.

## Parte 2: Modelo avanzado en GLPK

En la segunda parte de la práctica, el problema se centra en la asignación de franjas horarios de mantenimiento para un grupo de autobuses que deben acudir a un único taller. El taller dispone de un número limitado de franjas de tiempo en las que solo puede atender a un autobús por franja. Cada autobús se caracteriza por su distancia al taller y por el número de pasajeros que transporta, lo que influye directamente en el coste asociado a su asignación o no asignación. Si un autobús es atendido, se incurre en un coste proporcional a su distancia recorrida. En cambio, si no se atiende, se aplica una penalización proporcional al número de pasajeros afectados. El objetivo del modelo es determinar qué autobuses deben ser atendidos y en qué franja, de modo que se minimice el coste total del sistema, respetando las limitaciones de capacidad del taller y las condiciones de asignación.

### 2.1 Minimización del impacto de las averías

En esta parte, se pide asignar una franja horaria de mantenimiento a cada autobús, buscando minimizar el coste total derivado de los desplazamientos y de las penalizaciones por aquellos autobuses que no puedan ser atendidos. El modelo debe respetar las restricciones de capacidad del taller, que solo puede atender a un vehículo por franja, y decidir qué autobuses serán asignados. Comenzaremos por definir la notación empleada para modelar formalmente el problema.

#### 2.1.1 Conjuntos

Este conjunto representa los autobuses que deben ser asignados a franjas.

$A$ : índice de autobuses  $\rightarrow \{1, \dots, m\}$

Este conjunto representa las franjas disponibles en el taller para la asignación de los autobuses.

$F$ : índice de franjas del único taller  $\rightarrow \{1, \dots, n\}$

### 2.1.2 Parámetros

Estos parámetros representan los datos conocidos del problema, como las distancias, el número de pasajeros, los costes y la disponibilidad de cada autobús.

$d_a \geq 0$ : distancia (km) desde el autobús  $a$  al taller.  $\forall a \in A$ .

$p_a \geq 0$ : número de pasajeros del autobús  $a$ .  $\forall a \in A$ .

$k_d \geq 0$ : coste (€) por kilómetro si el autobús es atendido.

$k_p \geq 0$ : penalización (€) por pasajero si el autobús no es atendido.

### 2.1.3 Variables de decisión

Estas variables son mutuamente excluyentes: si un autobús es asignado a una franja, no puede ser no ser asignado a ninguna, y viceversa.

$x_{a,f} \in \{0, 1\} \forall a \in A, f \in F$ : vale 1 si el autobús  $a$  se asigna a la franja  $f$ .

$y_a \in \{0, 1\} \forall a \in A$ : vale 1 si el autobús  $a$  no se asigna a ninguna franja.

### 2.1.4 Función objetivo

El objetivo es minimizar el coste total, que está compuesto por los costes de desplazamiento de los autobuses a los talleres, y las penalizaciones por los autobuses no atendidos.

$$\min z = \sum_{a \in A} \sum_{f \in F} k_d \cdot d_a \cdot x_{a,f} + \sum_{a \in A} k_p \cdot p_a \cdot y_a$$

Si  $a$  se atiende en alguna franja ( $\sum_{f \in F} x_{a,f} = 1$ ), incurre  $k_d d_a$ .

Si no se atiende ( $y_a = 1$ ), incurre  $k_p p_a$ .

### 2.1.5 Restricciones

Estas restricciones garantizan la viabilidad operativa indicada: exclusividad por franja y posibilidad de dejar autobuses sin atender cuando  $n < m$ .

1. Restricción de asignación única:

Se asegura que cada autobús  $a$  sea asignado a una franja  $f$  o no se asigne a ninguna franja  $y_a$ .

$$\sum_{f \in F} x_{a,f} + y_a = 1, \forall a \in A$$

2. Restricción de capacidad de franja:

Limita la cantidad de autobuses que pueden ser asignados a una franja, asegurando que no haya más autobuses de los que se pueden manejar en cada franja.

$$\sum_{a \in A} x_{a,f} \leq 1, \forall f \in F$$

3. Restricción de dominio binario:

Asegura que las variables  $x_{a,f}, y_a$  solo puedan tomar los valores de 0 o 1, lo que es adecuado para una asignación de tipo binario.

$$x_{a,f} \in \{0, 1\}, \forall a \in A, \forall f \in F$$

$$y_a \in \{0, 1\}, \forall a \in A$$

### 2.1.6 Modelización

Por lo tanto, la modelización de nuestro problema en la Parte 2 para que el coste sea mínimo será:

$$\begin{aligned} \min z &= \sum_{a \in A} \sum_{f \in F} k_d \cdot d_a \cdot x_{a,f} + \sum_{a \in A} k_p \cdot p_a \cdot y_a \\ \text{s.a.} \quad &\left\{ \begin{array}{l} 1. \sum_{f \in F} x_{a,f} + y_a = 1, \forall a \in A \\ 2. \sum_{a \in A} x_{a,f} \leq 1, \forall f \in F \end{array} \right. \\ &x_{a,f} \in \{0, 1\}, \forall a \in A, \forall f \in F \\ &y_a \in \{0, 1\}, \forall a \in A \end{aligned}$$

### 2.1.7 Implementación del modelo general

El fichero *parte-2-1.mod* define los conjuntos  $A$  (*Autobuses*) y  $F$  (*Franjas*), junto con los parámetros: el coste por kilómetro recorrido  $k_d$  (*coste\_distancia*), la penalización por pasajero no atendido  $k_p$  (*coste\_pasajero*), la distancia al taller  $d_a$  (*distancia*), el número de pasajeros  $p_a$  (*pasajero*). Las variables binarias *asignado* y *sin\_asignar* indican si el autobús  $a$  se asigna a la franja  $f$  o queda sin atender. La función objetivo minimiza el coste total derivado de los autobuses atendidos y los no atendidos, combinando el coste de desplazamiento y la penalización por pasajeros no atendidos. El modelo incluye restricciones que aseguran que cada autobús se asigne, como máximo, a una franja (o quede sin asignar) y que cada franja del taller solo pueda atender a un autobús.

### 2.1.8 Script para generar ficheros de datos

El script *gen-1.py* automatiza la ejecución del modelo. A partir de un fichero de entrada *.in*, genera un fichero de datos *.dat* con los parámetros correspondientes y ejecuta el solver GLPK para resolver el problema. El programa muestra por pantalla el valor óptimo de la función objetivo, el número de variables y restricciones creadas, e indica para cada autobús si ha sido asignado a una franja o ha quedado sin atender.

## 2.2 Maximización de la satisfacción de los pasajeros

En esta parte de la práctica se busca asignar cada autobús a una franja horaria disponible en alguno de los talleres, con el objetivo de maximizar la satisfacción de los pasajeros. Para ello, se minimiza el número total de usuarios que coinciden en la misma franja en talleres diferentes, considerando la disponibilidad de cada franja y garantizando que todos los autobuses queden asignados y que cada franja solo sea ocupada por un autobús.

### 2.2.1 Conjuntos

Los conjuntos deben representar los elementos del problema que varían su tamaño: autobuses, franjas y talleres disponibles.

$A$ : índice de autobuses  $\rightarrow \{1, \dots, m\}$

$F$ : índice de franjas  $\rightarrow \{1, \dots, n\}$

$T$ : índice de talleres  $\rightarrow \{1, \dots, u\}$

### 2.2.2 Parámetros

Los parámetros recogen la información necesaria para cuantificar la relación entre autobuses y la disponibilidad de los talleres.

$c_{a,b} \geq 0$ : número de pasajeros que han contratado simultáneamente los servicios de los autobuses  $a$  y  $b$ .  $\forall a, b \in A$ . Este valor mide el grado de relación o coincidencia entre los clientes de ambos autobuses.

$o_{f,t} \in \{0, 1\}$ : disponibilidad de la franja  $f$  en el taller  $t$ , tomando el valor 1 si está disponible, y 0 si no lo está.  $\forall f \in F, \forall t \in T$ .

### 2.2.3 Variables de decisión

Las variables de decisión determinan la asignación de los autobuses a las franjas de los talleres y las posibles coincidencias entre ellos.

$x_{a,f,t} \in \{0, 1\}$ : vale 1 si el autobús  $a$  se asigna a la franja horaria  $f$  del taller  $t$ , y 0 en caso contrario.  $\forall a \in A, \forall f \in F, \forall t \in T$ .

$u_{a,b,f} \in \{0, 1\}$ : vale 1 si los autobuses  $a$  y  $b$  coinciden en la misma franja  $f$ ; 0 en caso contrario.  $a < b; \forall a, b \in A; \forall f \in F$ .

### 2.2.4 Función objetivo

El objetivo de la modelización es un problema de Programación Lineal entera que busca minimizar el número total de usuarios que están asignados a la misma franja horaria. El objetivo matemático es reducir el solapamiento de clientes para que así se maximice la satisfacción de estos.

$$\min z = \sum_{a < b} \sum_{f \in F} c_{a,b} u_{a,b,f}$$

### 2.2.5 Restricciones

Las restricciones deben garantizar que cada franja en cada taller solo pueda ser ocupada por un autobús, que todos los autobuses sean asignados, y que solo se usen franjas disponibles.

1. Restricción de asignación única por autobús:

Cada autobús debe asignarse exactamente a una combinación de franja y taller, garantizando que solo ocupe un turno de atención.

$$\sum_{f \in F, t \in T} x_{a,f,t} = 1, \forall a \in A$$

2. Restricción de capacidad máxima:

Cada par franja-taller puede atender, como máximo, a un único autobús.

$$\sum_{a \in A} x_{a,f,t} \leq 1, \forall f \in F, \forall t \in T$$

3. Restricción de disponibilidad de franja:

Solo pueden asignarse aquellas franjas que estén disponibles en el taller correspondiente, según el parámetro  $o_{f,t}$ .

$$x_{a,f,t} \leq o_{f,t}, \forall a \in A, \forall f \in F, \forall t \in T$$

4. Restricciones de coincidencias (linealización):

Definen la variable  $u_{a,b,f}$ , que toma valor 1 únicamente cuando dos autobuses  $a$  y  $b$  coinciden en la misma franja, sin importar el taller. Se formulan tres desigualdades para garantizar la relación correcta.

$$4.1 \quad u_{a,b,f} \leq \sum_{t \in T} x_{a,f,t}$$

$$4.2 \quad u_{a,b,f} \leq \sum_{t \in T} x_{a,f,t}$$

$$4.3 \quad u_{a,b,f} \geq \sum_{t \in T} x_{a,f,t} + \sum_{t \in T} x_{a,f,t} - 1$$

$$a < b; \quad \forall a, b \in A; \quad \forall f \in F$$

## 2.2.6 Modelización

Por lo tanto, la modelización de nuestro problema en la Parte 2.2 para que la satisfacción sea máxima será:

$$\min z = \sum_{a < b} \sum_{f \in F} c_{a,b} u_{a,b,f}$$

$$\text{s.a} \left\{ \begin{array}{l} 1. \quad \sum_{f \in F, t \in T} x_{a,f,t} = 1, \quad \forall a \in A \\ 2. \quad \sum_{a \in A} x_{a,f,t} \leq 1, \quad \forall f \in F, \forall t \in T \\ 3. \quad x_{a,f,t} \leq o_{f,t}, \quad \forall a \in A, \forall f \in F, \forall t \in T \\ 4. \quad u_{a,b,f} \leq \sum_{t \in T} x_{a,f,t} \\ \quad \quad u_{a,b,f} \leq \sum_{t \in T} x_{a,f,t} \\ \quad \quad u_{a,b,f} \geq \sum_{t \in T} x_{a,f,t} + \sum_{t \in T} x_{a,f,t} - 1, \quad a < b \end{array} \right.$$

$$x_{a,f,t} \in \{0, 1\}, \quad \forall a \in A, \quad \forall f \in F, \quad \forall t \in T$$

$$u_{a,b,f} \in \{0, 1\}, \quad a < b, \quad \forall a, b \in A, \quad \forall f \in F$$

$$a < b; \quad \forall a, b \in A$$

## 2.2.7 Implementación del modelo general

El fichero *parte-2-2.mod* define los conjuntos  $A$  (*Autobuses*),  $F$  (*Franjas*) y  $T$  (*Talleres*), junto con los parámetros: el número de pasajeros compartidos  $c_{a,b}$  (*suma\_pasajeros*) y la disponibilidad de cada franja en cada taller  $o_{f,t}$  (*disponibilidad*). La variable binaria *asignado* indica la asignación del autobús  $a$  a una combinación franja-taller, mientras que la variable binaria *coinciden* toma valor 1 cuando los autobuses  $a$  y  $b$  coinciden en la misma franja  $f$ . La función objetivo minimiza el número total de pasajeros que coinciden en una misma franja, con el propósito de maximizar la satisfacción de los usuarios. El modelo incorpora restricciones que garantizan que cada autobús se asigna exactamente a combinación de franja y taller, que cada franja-taller atienda como máximo a un autobús, que solo se utilicen las franjas disponibles y que la variable  $u_{a,b,f}$  refleje correctamente las coincidencias entre autobuses.

### 2.2.8 Script para generar ficheros de datos

El script *gen-2.py* automatiza la ejecución del modelo. A partir de un fichero de entrada .in, genera un fichero de datos .dat con los parámetros correspondientes y ejecuta el solver GLPK para resolver el problema. El programa comprueba el formato del fichero de entrada, genera las matrices de pasajeros y disponibilidad, crea el archivo .dat e invoca el solver. La salida muestra en pantalla el valor óptimo de la función objetivo, el número de variables y las restricciones creadas, e indica las asignaciones de cada autobús a cada conjunto de franja y taller correspondientes.

## Parte 3: Análisis y pruebas

### 3.1 Análisis de resultados

En este apartado analizaremos todos los resultados obtenidos en las tres partes.

#### Análisis de los resultados obtenidos en la Parte 1

El modelo en la parte 1 representa el problema base de asignación de autobuses a talleres con el objetivo de minimizar la distancia recorrida.

En la resolución de este se observa, en la matriz de asignaciones, que a cada autobús se le asigna exactamente un taller y que, a su vez, cada taller tiene como máximo un bus asignado, cumpliendo con las restricciones de asignación y capacidad definidas en la formulación. También se puede observar que se verifican dichas restricciones observando las filas de cada una, en donde los valores de la primera restricción toman valor 1 y los de la segunda no superan el 1. Además, las variables de decisión toman valores 0-1 como lo establecido.

Para comprobar que la distancia obtenida es realmente la mínima, el Solver evalúa todas las combinaciones posibles de asignaciones que satisfacen las restricciones del modelo y descarta aquellas que superen la mejor solución encontrada hasta el momento. De esta forma, el valor final obtenido es 573 km.

#### Análisis de los resultados obtenidos y verificación de restricciones

En total, los modelos combinados alcanzan aproximadamente entre 40 y 50 restricciones y entre 9 y 18 variables en los casos base, todas enteras y binarias.

Coste total (parte 2.1): Los costes óptimos obtenidos en las pruebas varían entre 20€ y 104€, dependiendo de los parámetros de distancia y pasajeros. El modelo asigna correctamente los autobuses a las franjas disponibles, respetando que cada autobús se asigne a una solo franja o quede sin asignar.

- En la Prueba1-1, que se detalla más adelante junto al resto de las pruebas, se asignan los autobuses 1 y 3 y el 2 queda libre, obteniéndose un coste mínimo de 50€.
- En la Prueba1-3, el modelo decide no asignar ningún autobús, lo que confirma que la función objetivo prioriza correctamente el coste mínimo.

Minimización de coincidentes (parte 2.2): Los objetivos óptimos varían entre 0 y 60 pasajeros coincidentes, según la disponibilidad de franjas y talleres.

- En la Prueba2-1, el modelo separa los autobuses 1 y 2 (que comparten muchos pasajeros) y reduce el solapamiento a 1.

- En la Prueba2-4, el solver detecta inviabilidad, lo que confirma que las restricciones de capacidad y disponibilidad se aplican correctamente.

Tras analizar los resultados, se confirma que todas las restricciones del modelo se cumplen: Cada autobús se asigna una sola vez, las franjas no superan su capacidad máxima y las franjas no disponibles se respetan en todas las soluciones.

### **Análisis de restricciones**

Las restricciones que más influyen en el comportamiento del modelo son las de capacidad máxima y las de asignación posibles y, por tanto, determinan la factibilidad del problema.

En la parte 2.2.1 son limitantes las restricciones de capacidad del taller cuando hay más autobuses que franjas. En la parte 2.2.2, las restricciones de disponibilidad y capacidad por (franja,taller) se vuelven críticas, ya que limitan la posibilidad de separar autobuses con muchos pasajeros comunes. En el caso de inviabilidad (Prueba2-4), la suma de estas restricciones impide encontrar solución, lo que demuestra que el modelo las respeta estrictamente.

### **Análisis de la complejidad del problema**

El modelo de la parte 2.2.1 es más simple, con una única combinación autobús-franja, mientras que el de la parte 2.2.2 incorpora un tercer índice (taller) y variables adicionales de coincidencia entre pares de autobuses, aumentando significativamente el número de columnas y restricciones.

En las pruebas, el número de variables pasó de 9 a 18 y las restricciones de 5 a más de 40, lo que muestra cómo la complejidad crece al añadir talleres y relaciones entre pares. A pesar de ello, el solver GLPK resolvió todos los casos en tiempos muy cortos y con una verificación completa de las condiciones de optimalidad e inviabilidad.

### **Ventajas y desventajas de LibreOffice y GLPK**

LibreOffice resulta útil para realizar comprobaciones rápidas o visualizar de forma sencilla los resultados de un problema pequeño. La herramienta Solver permite resolver modelos básicos con facilidad y ofrece una interfaz visual clara, lo que ayuda a comprender la lógica del problema sin necesidad de código. Sin embargo, sus capacidades son limitadas para modelos grandes o con variables binarias, como los utilizados en las partes 2.2.1 y 2.2.2. Además, la introducción de datos y parámetros de forma manual hace que las modificaciones sean más lentas y propensas a errores, lo que dificulta el trabajo con estructuras complejas o con múltiples conjuntos de datos.

GLPK, en cambio, está diseñado para abordar problemas de optimización de mayor escala y complejidad. Su separación entre archivos .mod y .dat facilita realizar múltiples pruebas cambiando solo los datos, manteniendo fija la estructura del modelo. Además, ofrece diagnósticos precisos del estado de la solución, detectando fácilmente inviabilidades o soluciones óptimas enteras. Por otro lado, la interpretación de resultados puede requerir herramientas adicionales o scripts, como *gen-2.py*, para automatizar la lectura y presentación de los resultados, lo que añade un pequeño grado de complejidad técnica respecto al uso directo de una hoja de cálculo.

## 3.2 Casos de pruebas

### 3.2.1 Pruebas para parte-2-1.mod

#### Prueba1-1: Selección bajo capacidad

Con 2 franjas y 3 autobuses,  $k_d = 1$  y  $k_p = 2$ , el modelo prioriza a quienes aportan mayor ahorro por pasajeros frente al coste por distancia. El resultado asigna a los dos autobuses con mayo “beneficio” y deja sin asignar al menos rentable. Se verifica la restricción “exactamente una franja o sin asignar” y la capacidad 1 por franja. El número de filas y columnas crece moderadamente y el valor óptimo es de 50.

#### Prueba1-2: Empate en beneficio

Con 1 franja y 3 autobuses, hay dos candidatos con beneficio máximo idéntico. El modelo asigna cualquiera de los empatados sin degradar el objetivo y mantiene unicidad por franja. Se comprueba el tratamiento correcto de empates y la estabilidad del objetivo. Complejidad baja (una franja) y valor óptimo de 104.

#### Prueba1-3: Conviene no asignar

Aunque hay 3 franjas y 2 autobuses, con  $k_d$  alto y  $k_p$  bajo, atender es más caro que no atender. El modelo deja ambos autobuses sin asignar cumpliendo la igualdad por bus y la capacidad por franja. Se valida que no fuerza asignaciones ineficientes. Menor ocupación de variables de asignación y valor óptimo de 20.

#### Prueba1-4: Capacidad holgada

Con 3 franjas y 3 autobuses, y parámetros que hacen rentable atender, el modelo asigna los tres, uno por franja. Se verifica el comportamiento en régimen no restrictivo y la correcta satisfacción de todas las restricciones. El tamaño del problema aumenta linealmente con  $m \cdot n$  y el valor óptimo es de 30.

### 3.2.2 Pruebas para parte-2-2.mod

#### Prueba2-1: Separar el par “caro”

Con 2 franjas, 2 talleres y 3 autobuses, la matriz de pasajeros comunes penaliza especialmente al par (1,2). El modelo los separa en franjas distintas y hace coincidir al tercero con el menos costoso, minimizando coincidencias. Se verifican las restricciones de asignación única, capacidad por franja-taller y disponibilidad. Complejidad mayor por variables *Coinciden* y objetivo 1.

#### Prueba2-2: Una sola franja disponible

Con 1 franja y 3 talleres, los 3 autobuses deben coincidir en la misma franja (en talleres distintos). El objetivo suma todas las parejas, actuando como cota superior natural del modelo. Se valida el enlace de *Coinciden* y la capacidad por franja-taller. El número de filas/columnas crece por las variables de coincidencia y el objetivo es 60.

#### Prueba2-3: Cero solapamiento posible

Con 3 franjas y 1 taller para 3 autobuses, la capacidad por franja evita cualquier coincidencia. El modelo coloca uno por franja y el objetivo se reduce a cero. Se verifica la linealización de *Coinciden* (se activa a 0) y el cumplimiento estricto de disponibilidad. El tamaño aumenta por pares  $a < b, f$  y el objetivo es 0.

#### Prueba2-4: Inviabile por falta de huecos

Con 1 franja y 2 talleres para 3 autobuses, hay solo 2 huecos totales y la asignación obligatoria vuelve el problema inviable. El solver reporta infeasibilidad, validando el control

de disponibilidad y capacidad. Se comprueba el manejo correcto de errores en la ejecución y la ausencia de soluciones.

## Conclusiones de la práctica

Esta práctica nos ha servido para aplicar técnicas de optimización en la planificación de autobuses y talleres. Empezamos con un modelo sencillo para minimizar los costes de atención y fuimos aumentando la dificultad hasta incluir varias franjas y talleres, buscando reducir las coincidencias entre autobuses con pasajeros comunes. Cada parte planteó nuevos retos, pero conseguimos formular los modelos y obtener soluciones coherentes.

Durante el desarrollo vimos que usar GLPK y separar el modelo de los datos hacía mucho más fácil probar distintos casos y modificar parámetros sin cambiar el código. Además, automatizar la lectura de resultados con Python nos ayudó a interpretar mejor las soluciones y detectar inviabilidades, aunque al principio fue más costoso entender cómo leer o “parsear” los resultados obtenidos del solver calculado en el fichero .py. Una vez resuelto, nos pareció una forma muy óptima de resolver y modelar problemas.

En general, la práctica nos permitió entender mejor cómo funcionan los modelos de optimización y cómo influyen las restricciones en el resultado final. Comprobamos que GLPK es una herramienta potente y flexible, ideal para experimentar con diferentes escenarios y analizar su impacto en la planificación.

## Práctica 1- Programación lineal

---



Heurística y optimización- Curso 25/26

---