ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE CIENCIAS

OPTIMIZACIÓN EN SISTEMAS DE TRANSPORTE PÚBLICO.

FORMULACIÓN LINEAL Y NO LINEAL PARA EL PROBLEMA DE PLANIFICACIÓN DE LÍNEAS Y FRECUENCIAS.

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERA MATEMÁTICA

YOMAIRA ELIZABETH CERÓN TIRIRA

yomaira.ceron@epn.edu.ec

DIRECTOR: PH.D. RAMIRO DANIEL TORRES GORDILLO

ramiro.torres@epn.edu.ec

DMQ, AGOSTO 2022

CERTIFICACIONES

Yo, YOMAIRA ELIZABETH CERÓN TIRIRA, declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Yomaira Elizabeth Cerón Tirira

Certifico que el presente trabajo de integración curricular fue desarrollado por Yomaira Elizabeth Cerón Tirira, bajo mi supervisión.

Ph.D. Ramiro Daniel Torres Gordillo

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante del mismo, es público y estará a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Yomaira Elizabeth Cerón Tirira

Ph.D. Ramiro Daniel Torres Gordillo

AGRADECIMIENTO

A mis padres, Manuel y Laura, por su amor incondicional y apoyo en toda mi vida estudiantil. A mi hermana Sandy, por su ejemplo de trabajo y valentía. A mis hermanos, Fabricio y Stalyn, por su cariño leal. A mis sobrinos, Sebastián y Matteo, por tantos momentos de alegría.

Un agradecimiento sincero al Dr. Ramiro Torres por haberme dado la oportunidad de trabajar en este proyecto. Gracias por su tiempo y paciencia, fue un verdadero honor haber aprendido y trabajado con uno de los mejores docentes que tiene la Facultad.

A mi amiga Gaby por apoyarme en cada paso que doy, a su familia, por haberme abierto las puertas de su hogar y brindarme su cariño sincero y desinteresado.

Un agradecimiento especial a todos mis amigos, compañeros y personas que tuve la oportunidad de conocer durante esta hermosa etapa de mi vida. Gracias por tantos momentos compartidos.

DEDICATORIA

A mi familia.

RESUMEN

El sistema de transporte público es el responsable del traslado de un gran número de personas en las ciudades y permitir de este modo el desarrollo de sus actividades diarias. La planificación de dichos sistemas es una tarea sumamente compleja y consiste de varios problemas que deben ser resueltos de manera secuencial: diseño de la red, planificación de líneas y frecuencias, diseño de itinerarios, asignación de flota y asignación de conductores. En el presente proyecto se pretende estudiar el problema de planificación de líneas y frecuencias con aplicación al sistema de transporte público de Quito. El problema consiste en encontrar un conjunto de líneas y sus respectivas frecuencias de manera que se pueda satisfacer una demanda de transporte conocida. Para el presente trabajo se considera la minimización de costos operativos y se presentan dos modelos de programación entera. El primero se formula como un modelo de programación no lineal entero y posteriormente es linealizado y se obtiene la segunda formulación expresada como un modelo de programación lineal entero con variables de decisión binarias. Para verificar el comportamiento de los modelos, se reportan resultados computacionales basados en instancias simuladas de diferentes tamaños. Finalmente, se presentan conclusiones sobre el presente trabajo.

Palabras clave: Planificación de líneas y frecuencias, programación lineal entera, transporte público.

ABSTRACT

The public transportation system is responsible for moving the mayority of people in the cities and in such a way it allows them to carry out the daily activities. The planning of such systems is an extremely complex task and it consists of several problems that must be solved sequentially: network design, line planning, timetable, vehicle scheduling and duty scheduling. This project aims to study the line planning problem with application to the Quito transport system. The problem consists of finding a set of lines and their frequencies in such a way that a transport demand is satisfied. For the present work, total operational cost is minimized and two integer programming models are considered. The first one is formulated as an integer nonlinear programming model and then it is linearized and the second formulation expressed as an integer linear programming model with binary decision variables is obtained. To verify the behavior of the models, computational results based on simulated instances of different sizes are reported. Finally, conclusions about the present work are presented.

Keywords: Line planning, integer linear programming, public transport.

Índice general

Capítulo 1

Introducción

La planificación estratégica de un sistema de transporte público es una tarea extremadamente compleja y ha sido el centro de varios trabajos en el campo de la optimización combinatoria. Dicho proceso en general es dividido en 3 fases [11]: diseño de la red, planificación de líneas y frecuencias y diseño del diagrama de marcha. Estas etapas dependen de la información que se reporta en la matriz origen-destino (matriz OD). Cada entrada de la matriz provee el número de pasajeros que desean trasladarse de una estación origen a una estación destino.

La primera etapa se refiere al diseño de la red de transporte, la cual consiste en identificar y seleccionar vías (calles, avenidas, etc) sobre las que podrían definirse las rutas de buses, tal que se permita transportar el máximo número de pasajeros. La segunda etapa está asociada al problema de planificación de líneas y frecuencias. El problema consiste en determinar un conjunto de líneas junto con sus respectivas frecuencias tal que la demanda de transporte sea satisfecha. Entre los objetivos del problemas están minimizar el número de transferencias que tienen que hacer los pasajeros o minimizar los costos operativos de las líneas. La última etapa consiste en diseñar el calendario de despacho tomando como entrada las líneas y frecuencias obtenidas en el paso anterior y se pretende minimizar el tiempo que tarda un pasajero en trasladarse de una estación a otra o minimizar el tiempo que un vehículo puede estar inactivo.

El presente trabajo se centra en el problema de planificación de lí-

neas y frecuencias, donde muchos artículos han sido reportados en la literatura en los últimos años. Así, Bussieck, Kreuzer y Zimmermann [9] desarrollan un modelo de programación entera que maximiza la calidad del servicio a través de la minimización del número total de cambios que un pasajero realiza para llegar a su destino, es decir, la maximización del número de viajeros directos. Para este proceso se considera como datos un conjunto de estaciones, las conexiones entre estaciones, el tiempo de viaje y la distancia para estos enlaces y se asume que los viajeros usan el camino más corto entre cada par de estaciones. Por otra parte, Linder [12] estudia la planificación de líneas y frecuencias enfocado en la minimización de costos para lo cual desarrolla un método tipo Branch and Bound. Bussieck, Lindner y Lübbecke [8] consideran los dos aspectos: servicio versus costo. Los autores consideran que los pasajeros requieren conexiones convenientes e idealmente directas y proponen un método heurístico basado en fijado de variables y nuevas cotas inferiores para el problema.

Claessens, van Dijk y Zwaneveld proponen estudiar el problema de planificación de líneas con aplicación al sistema de ferrocarriles de Holanda. Los autores proponen un modelo no lineal y otro lineal que minimiza los costos operativos totales y algoritmos exactos de solución tipo Branch and Bound. Goossens, van Hoesel y Kroon [8] desarrollan una formulación derivada de [10] e introducen un nuevo método de solución usando algoritmos tipo Branch and Cut, donde una variedad de desigualdades válidas y métodos de reducción del modelo son reportados.

El presente trabajo inicia con una revisión de las definiciones básicas que se van a utilizar en el desarrollo del proyecto. Posteriormente se presentan dos formulaciones de programación entera. La primera es un modelo no lineal mientras que la segunda es una versión linealizada de la primera. Finalmente, se implementan los dos modelos y se realizan experimentos computacionales con instancias simuladas.

Capítulo 2

Descripción del componente

En este trabajo se pretende analizar el problema de planificación de líneas y frecuencias considerando los costos operativos del sistema. La asignación de líneas implica la determinación de las estaciones origen y destino con sus frecuencias y el tipo de bus asociado. En este sentido se presentan dos modelos de programación entera que brindarán las soluciones al problema.

2.1. Objetivos

Objetivo General

Estudiar el problema de planificación de líneas y frecuencias utilizando los resultados obtenidos en [10] con el fin de aplicarlos al sistema metropolitano de Quito.

Objetivos Específicos

1. Revisión bibliográfica del problema de planificación de líneas y frecuencias.
2. Implementar un modelo de programación no lineal para el problema estudiado usando el solver de programación entera GUROBI.
3. Implementar un modelo de programación lineal para el problema estudiado usando el solver de programación entera GUROBI.

4. Conducir experimentos computacionales para verificar el comportamiento de los modelos.

2.2. Alcance

El presente proyecto tiene como objetivo estudiar el problema de planificación de líneas y frecuencias centrado en la minimización de los costos operativos. Para ello se utilizarán algunos resultados obtenidos previamente y se los adaptará al caso del sistema de transporte público de Quito.

Primero, se pretende implementar los modelos de programación lineal entera usando instancias manejables y posteriormente aumentar el tamaño de dichas instancias. Con ello no se espera obtener una nueva formulación del problema o un método de solución, sino únicamente replicar los resultados existentes a un caso particular.

2.3. Marco teórico

2.3.1. Teoría de Grafos

Definición 2.1. *Un grafo no dirigido es un conjunto finito* $V = \{v_1, \ldots, v_n\}$ *llamados vértices o nodos y un multiconjunto* $E = \{\{v_i, v_j\} : v_i, v_j \in V\}$ *de pares ordenados de* V*, conocidos como aristas.*

Un grafo se asocia por lo general a una representación gráfica que no es única. El conjunto de nodos del grafo se nota también como $V(G)$ y su conjunto de aristas $E(G)$. Además, se define $|V| = n$ como el número de nodos y $|E| = m$ como el numero de aristas de G.

- Dada una arista $e = \{i, j\} \in E$, los nodos i, j se conocen como extremos de la arista e.
- Dos aristas que tiene los mismos extremos se dicen paralelas.
- Una arista que conecta a un nodo consigo mismo se llama un lazo o bucle.

- Si G no tiene lazos ni aristas paralelas, entonces se denomina un grafo simple.
- Un grafo simple en el que cada par de nodos está conectado por una arista se denomina grafo completo.

Definición 2.2. *Sea $G = (V, E)$ no dirigido $W \subset V$ y $F \subset E$. Si $H = (W, F)$ es un grafo, entonces H es un subgrafo de G.*

Si además se cumple que F contiene a todas las aristas de G con ambos extremos en W se dice que $H = (W, F)$ es el grafo inducido por W y se nota $G[W]$. Si $W = V$ y $F \subset E$ entonces H se denomina el grafo generado por E y se nota $G[F]$.

Definición 2.3. *Un camino es un grafo no vacío $P = (V, E)$ de la forma:*

$$V = \{v_1, v_2, \dots v_k\},$$
$$E = \{\{v_i, v_{i+1}\}, \{v_{i+1}, v_{i+2}\}, \dots \{v_{k-1}, v_k\}\}$$

Los nodos v_i y v_k son unidos por P y son llamados nodos extremos y $v_{i+1}, \dots, v_{k-1}$ los nodos internos de P.

Definición 2.4. *Un ciclo es un camino en el que sus nodos extremos son iguales, es decir, $v_i = v_k$. Se lo llama también camino cerrado o circuito.*

Definición 2.5. *Un grafo no vacío $G = (V, E)$ es llamado conexo si para cualquier par de nodos u y v, existe al menos un camino entre u y v. En caso contrario se dice que no es conexo.*

Definición 2.6. *Un bosque es un grafo no dirigido sin ciclos o circuitos.*

Definición 2.7. *Un árbol es un bosque conexo, es decir, un grafo conexo sin ciclos.*

2.3.2. Programación Lineal

La programación lineal tiene como objetivo optimizar (maximizar el beneficio o minimizar el costo) una función lineal que está sujeta a restricciones lineales.

Definición 2.8. *Dados: $A \in M_{mxn}$, $b \in \mathbb{R}^m$ y $c \in \mathbb{R}^n$. Un problema de*

programación lineal se dice que está escrito en la forma canónica si se expresa como

$$\begin{cases} \text{Máx } c^T x \\ \text{s.a.} \\ \quad Ax \leq b \\ \quad x \geq 0 \end{cases}$$

Así, la programación lineal consiste en encontrar un vector x que maximiza $c^T x$ sujeto a $Ax \leq b$ y $x \geq 0$.

Existen algunos métodos para resolver problemas de programación lineal. Uno de ellos es el Método del Simplex [4], dicho algoritmo inicia en un vértice básico de la región factible y luego se mueve iterativamente a vértices adyacentes, mejorando la solución cada vez hasta encontrar la solución óptima. El primer algoritmo de tiempo polinomial fue propuesto por Khachiyan en 1979 [7], basado en el método del elipsoide, sin embargo, el tiempo de ejecución no es competitivo comparado con el método del Simplex. En 1984 se presenta el algoritmo de Karmarkar [6], que es una mejora del algoritmo de Khachiyan, también resuelve problemas lineales en tiempo polinomial y los autores reportan una implementación computacional competitiva respecto al Simplex.

2.3.3. Programación no Lineal

Es un conjunto de técnicas utilizadas para optimizar un problema donde la función objetivo o las restricciones son no lineales. La no linealidad del problema se presenta cuando el grado de las variables es distinto de uno o hay la presencia de un producto entre las variables a estudiar. Un problema de programación no lineal no siempre es factible.

Definición 2.9. *Sean $n, m, p, \in \mathbb{Z}^+$, $X \in \mathbb{R}^n$, f, g_i, h_j funciones evaluadas en X para todo $i \in \{1, \ldots, m\}$ y todo $j \in \{1, \ldots, p\}$ con al menos una de f, g_i, h_j no lineales. Un problema de programación no lineal se formula de la siguiente manera:*

$$\begin{cases} \text{Min } f(x) \\ \text{s.a.} \\ \quad g_i(x) \leq 0 \qquad \forall i \in \{1, \ldots, m\} \\ \quad h_j(x) = 0 \qquad \forall j \in \{1, \ldots, p\} \\ \quad x \in X \end{cases}$$

De acuerdo a la estructura de la función objetivo y restricciones, se pueden utilizar distintas técnicas de optimización. Así se tienen métodos de optimización convexa si la función objetivo es cóncava o convexa y el conjunto de restricciones es convexo. Si la función objetivo es cuadrática y las restricciones son lineales se aplica técnicas de programación cuadrática [1].

2.3.4. Programación Lineal Entera

Definición 2.10. *Se dice que un problema de programación lineal es de programación lineal entera si todas las variables pertenecen a* $\mathbb{Z}^+ = \{0, 1, 2, \ldots\}$*. La formulación de un problema de programación lineal entera tiene la siguiente forma:*

$$\begin{cases} \text{Máx } c^T x \\ \text{s.a.} \\ \quad Ax \leq b \\ \quad x \geq 0 \\ \quad x \in Z^+ \end{cases}$$

Un método de solución de problemas de programación lineal entera es el Algoritmo de Branch & Bound [3] que consiste en enumerar las soluciones candidatas y compararlas con los límites estimados, superior e inferior, de la solución óptima. Otro método de solución es el método de planos cortantes [2], cuya idea principal es mejorar iterativamente la región de búsqueda introduciendo desigualdades lineales. Por último, mencionamos el método Branch & Cut [3], el cual implica ejecutar el algoritmo Branch & Bound y usar planos de corte para ajustar las relajaciones de programación lineal.

Capítulo 3

Formulaciones enteras para el problema de planificación de líneas

En la presente sección se introducen dos modelos de programación entera para el problema de planificación de líneas y frecuencias. Cabe indicar que no se pretende diseñar una nueva formulación, únicamente se aspira adaptar y replicar los resultados obtenidos en [10] con el objetivo de aplicarlos a los corredores del sistema de transporte de Quito. Iniciamos haciendo un estudio más profundo de [10] para posteriormente adaptar las formulaciones para el caso de interés.

3.1. Descripción del problema

Claessens, van Dijk y Zwaneveld consideran el problema de planificación de las líneas y frecuencias aplicado al sistema ferroviario holandés. Los autores plantean minimizar los costos operativos totales del sistema asociado a la asignación de líneas. Un costo óptimo asociado a la asignación de líneas se puede conseguir si se conoce: la infraestructura entre estaciones, el número de pasajeros en cada vía, los costos asociados a los trenes y las limitaciones de servicio y capacidad.

El sistema ferroviario holandés cuenta con tres tipos de líneas: Intercity (IC) son trenes de larga distancia que conectan las ciudades más grandes de Holanda. Interregional (IR) operan entre regiones y también sirven a estaciones en ciudades medianas. El tipo de línea Aggloregional

(AR) opera dentro de una red específica y hacen escalas en casi todas las paradas. Las líneas de tipo IC se detienen solo en estaciones (IC), las líneas (IR) se detienen en las estaciones IC e IR y las líneas AR se detienen en todas las estaciones. La asignación de pasajeros en cada vía se la realiza mediante un sistema denominado *System Split*.

La estrategia principal de System Split se deriva de las matrices de origen - destino producidas por el modelo de movilidad nacional holandés. Las matrices OD se transforman de una manera bastante sencilla en una matriz de estaciones cuyas entradas en la casilla (p, q) indica el número esperado de pasajeros que viajan entre las estaciones p y q del sistema ferroviario en un determinado horizonte de tiempo. En un paso siguiente, System Split desagrega las entradas de la matriz en estimaciones del número esperado de pasajeros que viajan por cada tipo de tren en cada segmento de red. Si existen rutas alternativas o posibilidades de viajes entre dos estaciones, System Split asigna los pasajeros a las alternativas basándose en factores tales como la distancia de viaje, el tiempo de viaje y el número de transbordos requeridos para cada alternativa.

La longitud del tren se determina por el número de coches (vagones) que van a adherirse a la locomotora y estos dependen de la cantidad de pasajeros que se van a transportar. Es así que la longitud del tren (número de vagones) no se la va a considerar fija. El número de trenes tampoco es fijo, pero debe estar dentro ciertos límites para asegurar que todas las rutas sean cubiertas. Adicionalmente, se considera un total de tres rutas entre las estaciones origen y destino.

Los costos operativos que se incluyen en el problema son de tres tipos:

- **Costos fijos por tipo de vagón** : incluyen los costos fijos de mantenimiento y los costos de estacionamiento nocturno.
- **Costos variables por tipo de vagón** : cobro de boletos, costos de limpieza, costo de energía, costo de infraestructura y costo de mecánica.
- **Costos variables por tipo de tren** : estos costos incluyen el salario de conductor, una parte para los recolectores de boletos y costos de energía.

3.2. Formulación del Modelo

En la presente sección se presentan dos formulaciones para resolver el problema de planificación de líneas. El primero, un modelo no lineal entero que describe claramente el problema. Posteriormente, este modelo se transforma en un modelo de programación lineal entero para facilitar su solución.

3.2.1. Notación

Para la formulación de los modelos matemáticos se va a utilizar la siguiente notación.

Un grafo dirigido $G = (V, A)$ es un par ordenado compuesto por un conjunto V de nodos, que representan las estaciones, y A un conjunto de pares ordenados de nodos, conocidos como arcos y representan las conexiones directas entre estaciones.

La demanda de pasajeros está dada por la matriz origen-destino $T \in \mathbb{Z}^{|V|\times|V|}$, donde cada entrada t_{lk} representa el número de pasajeros que van a trasladarse de la estación l a la estación k.

Una línea es una conexión directa entre dos estaciones. Una línea se caracteriza por la estación de origen y destino, su frecuencia y las paradas intermedias por las que pasa el transporte. Un sistema de líneas es la colección de todas las líneas. Al conjunto de todas las líneas admisibles en el sistema de trasporte se lo notará como $\mathcal{L}$. En un sistema con horario periódico, la frecuencia es la cantidad de veces que la línea recorre una ruta. A cada línea $l \in \mathcal{L}$ se le asigna una frecuencia $f_l \in \mathbb{Z}^+$.

- **Índices**

  c número de coches,

  f frecuencia de una línea $(f = 1, \ldots, f^{\text{máx}})$, donde $f^{\text{máx}}$ es la frecuencia máxima que puede tomar una línea,

  i, j estaciones $(i, j = 1, \ldots, s^{\text{máx}})$, donde $s^{\text{máx}}$ es el número máximo de estaciones que tiene la red,

  k vía $(k = 1, \ldots, k^{\text{máx}})$, donde $k^{\text{máx}}$ es el número máximo de vías,

r ruta $(r = 1, \ldots, r^{\text{máx}})$, donde $r^{\text{máx}}$ es el número máximo de rutas, en este caso tres,

T conjunto de tipos de línea $(T = \{IC, IR, AR\},\ t \in T)$.

Los conjuntos de índices i, j, r, t y k indican que no todas las estaciones pueden utilizarse como estaciones de origen o destino de una línea.

- **Parámetros**

$c_t^{\text{máx}}$ número máximo de coches por tren de la linea tipo t,

$c_t^{\text{mín}}$ número mínimo de coches por tren de la línea tipo t,

$ccap_t$ capacidad del coche de la línea de tipo t,

$cfix_t$ costo fijo por hora de un coche de la linea de tipo t,

ckm_t costo por kilómetro de un coche con línea de tipo t,

cp_{ij}^{rt} tiempo mínimo de circulación de la estación i a la estación j que circula por la ruta r con la línea de tipo t,

d_{ij}^{rt} distancia entre las estaciones i y j usando la ruta r y un tren tipo t,

$f_k^{\text{máx}}$ frecuencia máxima de un tren sobre la vía k,

$f_k^{\text{mín}}$ frecuencia mínima de un tren sobre la vía k,

n_k número de pasajeros en la vía k,

$trkm_t$ costos por kilómetro de un tren con línea de tipo k,

δ_{ijk}^{rt} toma el valor de 1 si la línea de la estación i a la estación j con línea de tipo t se traslada por la vía k cubriendo la ruta r, 0 caso contrario.

Notemos que el conjunto de coches forma una composición o un tren. El número de composiciones para una línea depende del tiempo de viaje desde su origen hasta su destino, su frecuencia y su tiempo mínimo de respuesta y se puede calcular a partir del tiempo de circulación. El tiempo de circulación se divide para 60 minutos y se obtiene el tiempo mínimo de circulación (cp_{ij}^{rt}). A este tiempo mínimo se multiplica por la frecuencia de la línea, se redondea hacia arriba y se obtiene el número total de trenes necesarios por hora para operar una línea.

3.2.2. Formulación no lineal entera

El problema se puede formular en base a las siguientes variables de decisión:

- F_{ij}^{rt} : la frecuencia de una línea desde la estación i hasta la estación j, cubriendo la ruta r con un tren de tipo t.
- C_{ij}^{rt} : el número de coches de una línea desde la estación i hasta la estación j, cubriendo la ruta r con un tren de tipo t.

El modelo de programación no lineal entera se puede expresar de la siguiente manera.

$$\text{Min} \sum_{r=1}^{r^{\text{máx}}} \sum_{t \in T} \sum_{i=1}^{s^{\text{máx}}} \sum_{j=i+1}^{s^{\text{máx}}} cfix_t \cdot \lceil cp_{ij}^{rt} F_{ij}^{rt} \rceil C_{ij}^{rt} + d_{ij}^{rt}\, ckm_t F_{ij}^{rt}\, C_{ij}^{rt} + d_{ij}^{rt}\, trkm_t\, F_{ij}^{rt} \quad (3.1)$$

s.a.

$$\sum_{r=1}^{r^{\text{máx}}} \sum_{t \in T} \sum_{i=1}^{s^{\text{máx}}} \sum_{j=i+1}^{s^{\text{máx}}} \delta_{ijk}^{rt}\, ccap_t\, F_{ij}^{rt}\, C_{ij}^{rt} \geqslant n_k, \qquad \forall k, \quad (3.2)$$

$$f_k^{\text{mín}} \leqslant \sum_{r=1}^{r^{\text{máx}}} \sum_{t \in T} \sum_{i=1}^{s^{\text{máx}}} \sum_{j=i+1}^{s^{\text{máx}}} \delta_{ijk}^{rt}\, F_{ij}^{rt} \leqslant f_k^{\text{máx}}, \qquad \forall k, \quad (3.3)$$

$$c_t^{\text{mín}}\, F_{ij}^{rt} \leqslant F_{ij}^{rt}\, C_{ij}^{rt} \leqslant c_t^{\text{máx}}\, F_{ij}^{rt}, \qquad \forall i, j > i, r, t, \quad (3.4)$$

$$0 \leqslant F_{ij}^{rt} \leqslant f^{\text{máx}}, F_{ij}^{rt} \in \mathbb{Z}^+, \qquad \forall i, j > i, r, t, \quad (3.5)$$

$$C_{ijk}^{rt} \geqslant 0, C_{ijk}^{rt} \in \mathbb{Z}^+, \qquad \forall i, j > i, r, t \quad (3.6)$$

La función objetivo (3.1) agrupa los costos fijos por cada vagón que se van a utilizar, los costos variables por vagón y los costos variables por tren. Las restricciones (3.2) aseguran que se satisface la demanda de pasajeros, es decir, se asegura suficiente capacidad en el sistema de transporte para que todos los pasajeros sean transportados. Las restricciones (3.3) indican que en cada vía las frecuencias de las líneas cumplan con las cotas impuestas por las restricciones de servicio y capacidad. Las restricciones (3.4) restringen un número mínimo y máximo de vagones por tren. Las restricciones (3.5) aseguran que la frecuencia de una línea sea un número entero positivo y no supere su frecuencia máxima esta-

blecida. Las restricciones (3.6) aseguran que el número de vagones sea un número entero positivo.

En esta formulación existe presencia de factores no lineales, tanto en la función objetivo (3.1) como en las restricciones (3.2) y (3.4), se plantea realizar una linealización del modelo.

3.2.3. Formulación lineal entera

Se introducen las siguientes variables binarias:

$$X_{ij}^{rtfc} = \begin{cases} 1, & \text{si la línea que va de la estación } i \text{ a la estación } j \\ & \text{por la ruta } r \text{ y del tipo } t \text{ está incluida en el} \\ & \text{sistema con una frecuencia } f \text{ y } c \text{ coches} \\ 0, & \text{otro caso} \end{cases}$$

El modelo linealizado puede ser escrito de la siguiente forma:

$$\text{Min} \quad \sum_{r=1}^{r^{\text{máx}}} \sum_{t \in T} \sum_{f=1}^{f^{\text{máx}}} \sum_{c=c_t^{\text{mín}}}^{c_t^{\text{máx}}} \sum_{i=1}^{s^{\text{máx}}} \sum_{j=i+1}^{s^{\text{máx}}} \Big(cfix_t \cdot \lceil cp_{ij}^{rt}\, f \rceil\, cX_{ij}^{rtfc} + d_{ij}^{rt}\, ckm_t\, f\, c\, X_{ij}^{rtfc} + d_{ij}^{rt}\, trkm_t\, f\ X_{ij}^{rtfc} \Big) \tag{3.7}$$

s.a.

$$\sum_{r=1}^{r^{\text{máx}}} \sum_{t \in T} \sum_{f=1}^{f^{\text{máx}}} \sum_{c=c_t^{\text{mín}}}^{c_t^{\text{máx}}} \sum_{i=1}^{s^{\text{máx}}} \sum_{j=i+1}^{s^{\text{máx}}} \delta_{ijk}^{rt}\, ccap_t\, f\, c\, X_{ij}^{rtfc} \geqslant n_k, \qquad \forall k, \tag{3.8}$$

$$f_k^{\text{mín}} \leqslant \sum_{r=1}^{r^{\text{máx}}} \sum_{t \in T} \sum_{f=1}^{f^{\text{máx}}} \sum_{c=c_t^{\text{mín}}}^{c_t^{\text{máx}}} \sum_{i=1}^{s^{\text{máx}}} \sum_{j=i+1}^{s^{\text{máx}}} \delta_{ijk}^{rt}\, f\, X_{ij}^{rtfc} \leqslant f_k^{\text{máx}}, \quad \forall k, \tag{3.9}$$

$$\sum_{f=1}^{f^{\text{máx}}} \sum_{c=c_t^{\text{mín}}}^{c_t^{\text{máx}}} X_{ij}^{rtfc} \leqslant 1, \qquad \forall i, j > i, r, t, \tag{3.10}$$

$$X_{ij}^{rtfc} \in \{0, 1\}, \qquad \forall i, j > i, r, t, f, c \tag{3.11}$$

La función objetivo (3.7) muestra los costos del sistema de líneas. Las restricciones (3.8) garantizan que se satisface la demanda de pasajeros que se van a transportar. Las restricciones (3.9) indican que la frecuencia

de las líneas va a estar acotada inferior y superiormente. Las restricciones (3.10) señalan que la frecuencia y la longitud de tren para una línea son únicas. Las restricciones (3.11) aseguran que las variables X_{ij}^{rtfc} son variables de decisión binarias.

3.3. Complejidad

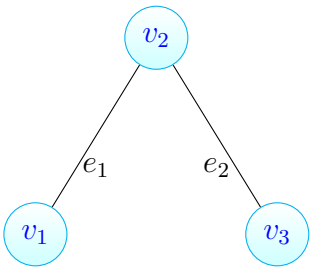
En [10] se demuestra que el problema de encontrar un sistema de líneas factible es NP - completo cuando se considera el número de trenes fijo. Para ello se considera una versión simplificada del problema que se describe a continuación.

Considere una red ferroviaria que consta de $S = \{1, \ldots, s^{\text{máx}}\}$ estaciones y $k \in \mathcal{K}$ vías entre un par de estaciones. El número de pasajeros en la vía k está dado por n_k. Una línea $l \in \mathcal{L}$ se caracteriza por su estación de origen y destino. La frecuencia máxima de la línea l se denota por $f_l^{\text{máx}}$ y la capacidad de dicha línea es ccap_l. El costo de la línea l por frecuencia se denota por c_l. Una solución factible es un conjunto de líneas $M \subset \mathcal{L}$ tales que todos los pasajeros pueden ser transportados.

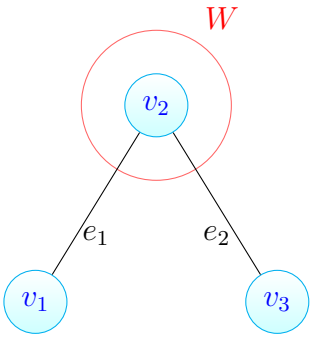
Teorema 3.1. *El problema simplificado es NP - duro.*

Demostración. La idea básica para la demostración es que el problema puede ser transformado desde el problema del cubrimiento de vértices. Se sabe que el problema de cubrimiento de vértices es NP-duro, entonces si logramos hallar una equivalencia entre una instancia de dicho problema y una instancia del problema simplificado demostraríamos que nuestro problema es NP-duro.

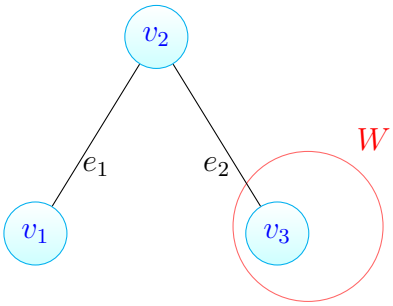
El problema de cubrimiento de vértice dice que dado un grafo $G = (V, E)$, se debe seleccionar un subconjunto $W \subset V$ de nodos tal que toda arista en el grafo debe tener al menos un nodo en el conjunto W. Por ejemplo, dado el siguiente grafo, podríamos seleccionar el conjunto W de varias formas:



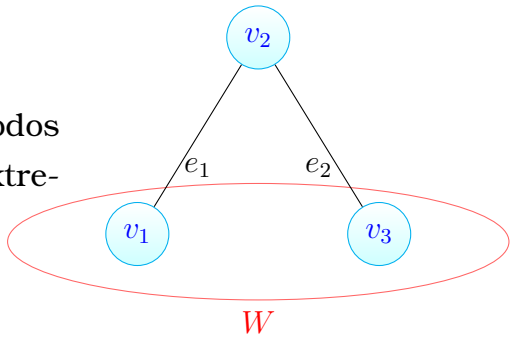
- Si seleccionamos el nodo v_2, entonces $W = \{v_2\}$, así la arista e_1 y e_2 tienen algún nodo extremo dentro del conjunto W.



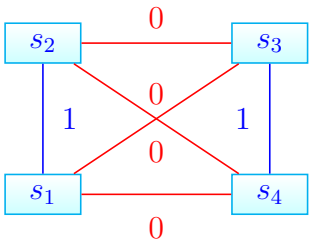
- Si en cambio seleccionamos el nodo v_3, vemos que la arista e_1 no tiene ningún extremo en W.



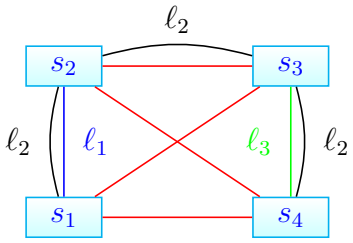
- Así también se podría tomar los nodos v_1 y v_3, y las aristas e_1 y e_2 tienen extremos en W.



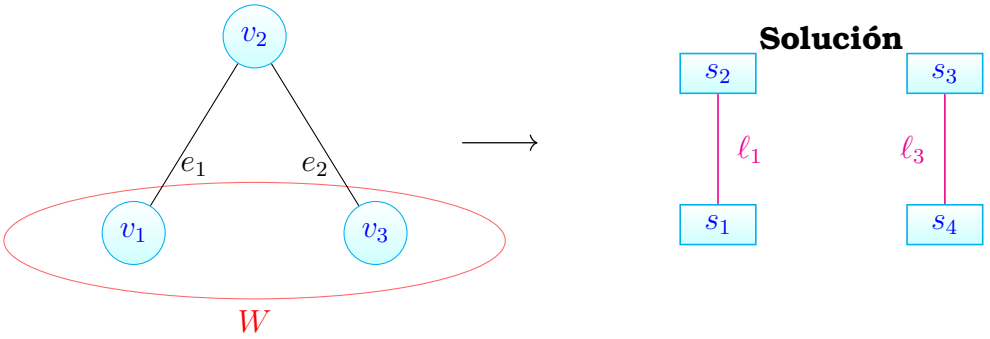
Entonces vamos a transformar una instancia de este problema dada por un grafo $G = (V, E)$ en una instancia del problema de planificación de líneas simplificado. Para ilustrar esta transformación consideremos el grafo anterior, donde para cada arista vamos a construir dos estaciones y conectarlas por una arista. Para e_1 se construyen las estaciones s_1 y s_2 y para la arista e_2 se construyen otro par de nodos distinto s_3 y s_4, los que son unidos por una arista como se muestra en la figura. Cada arista tiene una demanda igual a 1. La red se completa con aristas entre cada par de nodos con una demanda igual a cero.



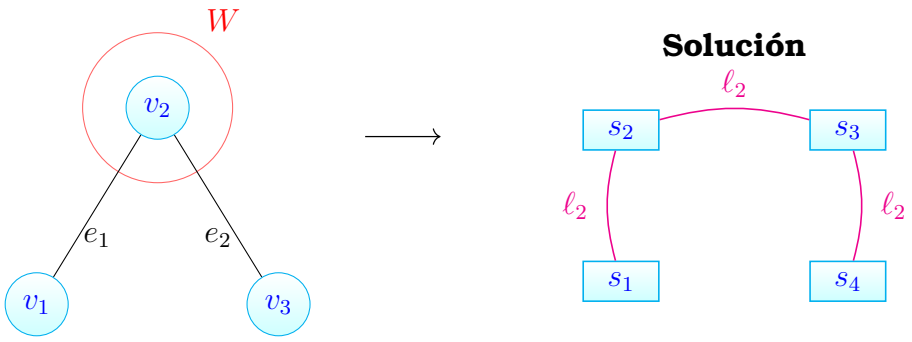
Para cada nodo vamos a generar una línea, donde si el nodo es adyacente con alguna arista en G, entonces la línea usa la arista asociada en la red de transportación. Así, el nodo v_1, como tiene intersección con la arista e_1, generamos la línea l_1 que va de s_1 a s_2 usando únicamente la conexión $\{s_1, s_2\}$. El nodo v_3 se interseca con e_2 por tanto construimos la línea l_3 que usa únicamente la conexión $\{s_3, s_4\}$. Por otro lado, el nodo v_2 es adyacente a las aristas e_1, $e_2 \in E$ por lo que se genera una tercera línea ℓ_2 usando las aristas $\{s_1, s_2\}$, $\{s_2, s_3\}$, $\{s_3, s_4\}$. El costo de cada línea construída es igual a 1.



Ahora, vamos a minimizar el costo total de la solución. Entonces si seleccionamos la línea ℓ_1 también debemos seleccionar la línea ℓ_3 para cubrir la demanda de $\{s_1, s_2\}$ y de $\{s_3, s_4\}$ que se relaciona con tomar el conjunto $W = \{v_1, v_2\}$.



La otra opción es tomar la linea ℓ_2, puesto que con esta se cubren las demandas de $\{s_1, s_2\}$ y $\{s_3, s_4\}$. Obviamente la solución de menor costo es seleccionar únicamente la línea ℓ_2 que genera una solución del problema de cubrimiento de nodos tomando el conjunto $W = \{v_2\}$.



De esa manera tenemos una transformación polinomial de una instancia del problema del cubrimiento de nodos a una instancia del problema de planificación de líneas, probando de este modo que el problema estudiado es NP-duro. □

3.4. Aplicación al Sistema de Transporte de Quito

En esta sección, vamos a adaptar las formulaciones de [10] para el sistema Trolebús del distrito metropolitano de Quito. Para ello, consideramos el grafo $G = (V, E)$ tal que:

$$V = \{1, \ldots, s^{\text{máx}}\}$$
$$E = \left\{\{i, i+1\} : i = 1, \ldots, s^{\text{máx}} - 1\right\}$$

Dado que el sistema de Trolebús posee una única vía, no consideramos el índice k. Por otra parte, dada la infraestructura del sistema, consideramos una única ruta entre cada par de estaciones, así el índice r tampoco es considerado. El índice t no es incluído en la formulación ya que asumiremos que existe un solo tipo de sistema de transporte. El concepto de coche le asociamos al tipo de vehículo, así $c^{\text{mín}} = 1$ corresponde a un bus normal, $c = 2$ un vehículo articulado y $c^{\text{máx}} = 3$ indica que se va a utilizar un vehículo biarticulado. Por tanto, los parámetros utilizados en esta formulación son:

Parámetros

$c^{\text{máx}}$	número máximo de coches,
$ccap$	capacidad del coche,
$cfix$	costo fijo por hora de un coche,

ckm costo por kilómetro de un coche,

cp_{ij} tiempo mínimo de circulación de la estación i a la estación j,

d_{ij} distancia entre las estaciones i y j,

$f^{\text{máx}}$ frecuencia máxima de una línea,

n_{lk} número de pasajeros entre cada par de estaciones,

$trkm$ costos por kilómetro de un vehículo.

3.4.1. Formulación no lineal

La formulación no lineal se la realiza en base a las siguientes variables:

- F_{ij} : frecuencia de una línea desde la estación i hasta la estación j.
- C_{ij} : el número de coches de una línea que circula desde la estación i hasta la estación j.

$$\text{Min} \sum_{i=1}^{s^{\text{máx}}} \sum_{j=i+1}^{s^{\text{máx}}} cfix \cdot \lceil cp_{ij} F_{ij} \rceil C_{ij} + d_{ij}\, ckm F_{ij}\, C_{ij} + d_{ij}\, trkm\, F_{ij} \tag{3.12}$$

s.a.

$$\sum_{i=1}^{s^{\text{máx}}} \sum_{j=i+1}^{s^{\text{máx}}} ccap\, F_{ij}\, C_{ij} \geqslant n_{lk}, \qquad \forall \{l,k\} \in E, \tag{3.13}$$

$$0 \leqslant C_{ij} \leqslant c^{\text{máx}}, C_{ij} \in \mathbb{Z}^{+}, \qquad \forall i, j > i, \tag{3.14}$$

$$0 \leqslant F_{ij} \leqslant f^{\text{máx}}, F_{ij} \in \mathbb{Z}^{+}, \qquad \forall i, j > i \tag{3.15}$$

La función objetivo (3.12) agrupa los costos del sistema de transporte. Las restricciones (3.13) aseguran que la demanda sea satisfecha. Las restricciones (3.14) garantizan que el número de coches tome un número entero positivo, mientras que las restricciones (3.15) indican que la frecuencia de una línea sea un número entero positivo.

3.4.2. Formulación lineal

Al modelo no lineal entero anterior lo transformamos en un modelo de programación lineal entera introduciendo la siguiente variable de decisión

$$X_{ij}^{fc} = \begin{cases} 1, & \text{si la línea que va de la estación } i \text{ a la estación } j \text{ está incluida} \\ & \text{en la solución con frecuencia } f \text{ y } c \text{ coches,} \\ 0, & \text{otro caso.} \end{cases}$$

Entonces tenemos la siguiente formulación lineal.

$$\text{Min} \sum_{f=1}^{f^{\text{máx}}} \sum_{c=1}^{c^{\text{máx}}} \sum_{i=1}^{s^{\text{máx}}} \sum_{j=i+1}^{s^{\text{máx}}} \Big(cfix \cdot \lceil cp_{ij} \cdot f \rceil \cdot c\, X_{ij}^{fc} + d_{ij} \cdot ckm \cdot f \cdot c\, X_{ij}^{fc} + d_{ij}\, trkm \cdot f \cdot X_{ij}^{fc} \Big) \tag{3.16}$$

s.a.

$$\sum_{f=1}^{f^{\text{máx}}} \sum_{c=1}^{c^{\text{máx}}} \sum_{i=1}^{s^{\text{máx}}} \sum_{j=i+1}^{s^{\text{máx}}} ccap\, f\, c\, X_{ij}^{fc} \geqslant n_{lk}, \qquad \forall \{l,k\} \in E, \tag{3.17}$$

$$\sum_{f=1}^{f^{\text{máx}}} \sum_{c=1}^{c^{\text{máx}}} X_{ij}^{fc} \leqslant 1, \qquad \forall i,j > i, \tag{3.18}$$

$$X_{ij}^{fc} \in \{0,1\}, \qquad \forall\, i,j > i, f, c \tag{3.19}$$

La función objetivo (3.16) indica la minimización de los costos totales del sistema, las restricciones (3.17) aseguran que se cubra la demanda de pasajeros en cada arco que forma parte de la red de transporte. Las restricciones (3.18) garantizan que la frecuencia y el número de coches por línea van sean únicos y las restricciones (3.19) señalan que las variables X_{ij}^{fc} son variables de decisión binarias.

Capítulo 4

Resultados Computacionales

En el presente capítulo se reportan los resultados de los experimentos computacionales para el caso particular de la topología del sistema Trolebús de la ciudad de Quito. Adicionalmente, se incluyen los resultados que se obtuvieron al trabajar con redes tipo árbol.

Todas los experimentos computacionales fueron realizados utilizando un computador con procesador AMD Ryzen 7 4700U with Radeon Graphics 2.00 GHz con 8 GB de memoria RAM en Windows 10. Además, se utilizó la versión 9.5.1 del solver de optimización Gurobi [5] a través del navegador Anaconda. Los modelos se implementaron en la aplicación Jupyter Notebook con el lenguaje de programación Python.

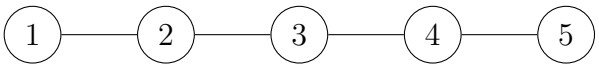
Para el desarrollo de este trabajo se tomaron 44 instancias que fueron simuladas de acuerdo a la operación real por hora del Sistema Trolebús. Estas instancias incluyen: el número de estaciones (n) y una matriz de n filas por $n+2$ columnas. En la primera columna se tiene el nombre de cada estación, en la segunda se reporta un indicador que toma el valor de 2 si la estación es terminal, 1 si la estación es de giro y 0 si no lo es, en la tercera columna se incluye la distancia del terminal inicial a las demás estaciones y finalmente se presenta la Matriz OD, donde cada entrada corresponde al número de pasajeros que van a ser transportados de una estación origen hacia una estación destino.

4.1. Construcción de la instancia

A continuación se ilustra la construcción de una instancia del problema de planificación de líneas a partir de la matriz origen-destino. Para esto consideramos una instancia de 5 estaciones. Así tenemos

$$V = \{1, 2, 3, 4, 5\}$$
$$E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}\}$$

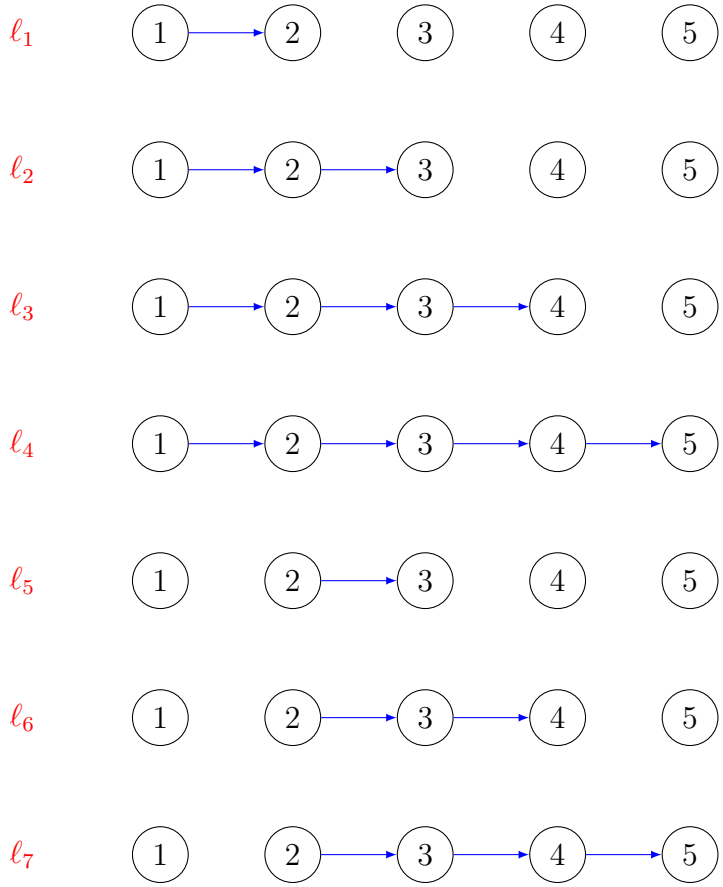
Por tanto el grafo es de la siguiente forma

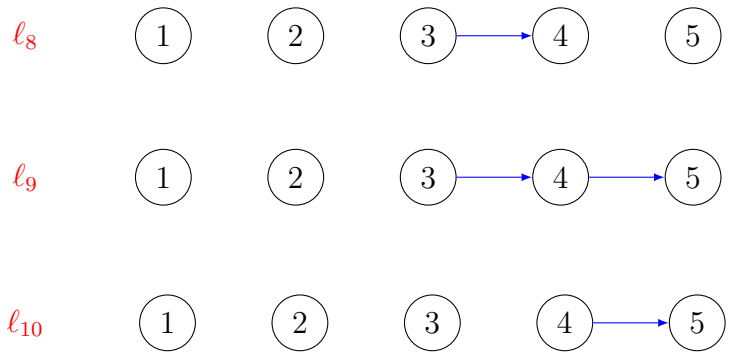


De acuerdo al modelo, las líneas que se van a generar son

$$L = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}$$

Gráficamente se ven así:





La matriz OD, está dada por:

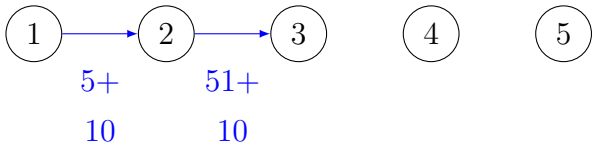
$$T = \begin{pmatrix} 0 & 5 & 10 & 21 & 23 \\ 4 & 0 & 51 & 41 & 21 \\ 8 & 12 & 0 & 12 & 32 \\ 17 & 13 & 9 & 0 & 21 \\ 8 & 16 & 30 & 20 & 0 \end{pmatrix}$$

Notemos que la demanda sobre el arco $\{l, k\}$ es distinta a la demanda de pasajeros sobre el arco $\{k, l\}$, por tanto la demanda final sobre el arco $\{l, k\}$ se calcula tomando el máximo de las demandas en cada dirección, es decir, $n_{lk} = \text{máx}\{n_{lk}, n_{kl}\}$. Esto debido a que las líneas son cerradas, es decir, una línea que inicia en i y gira en j, visita a todas las estaciones intermedias en dirección de i a j y luego en dirección j a i.

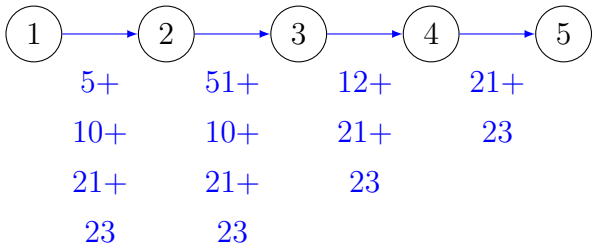
Así, vamos a calcular la demanda sobre cada uno de los arcos de la red. Entonces, la cantidad de personas que deseen transportarse desde la estación 1 a la estación 2 está dada por la entrada (1,2) de la matriz OD. Ya que existe un solo camino entre cualquier par de nodos, la demanda parcial sobre la arista $\{1, 2\}$ es $n_{12} = 5$.

De forma similar, si consideramos el viaje de la estación 2 a la estación 3, se tiene que existen 51 pasajeros que desean transportarse entre estas dos estaciones. Esto implica que la demanda parcial es $n_{23} = 51$. Por otro lado, si analizamos el viaje de la estación 1 a la estación 3, para llegar a su destino se debe atravesar los arcos $\{1, 2\}, \{2, 3\}$ cuyas demandas

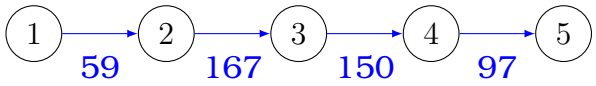
previas son $n_{12} = 5$ y $n_{23} = 51$, respectivamente. Consideramos la demanda de dicho viaje igual a $t_{13} = 10$, así la demanda acumulada puede ser calculada como:



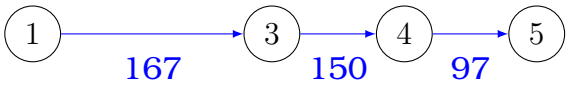
Replicando el proceso anterior para los viajes (1,4) y (1,5), se tiene



Repitiendo iterativamente el proceso anterior para todos los viajes, las demandas resultantes son



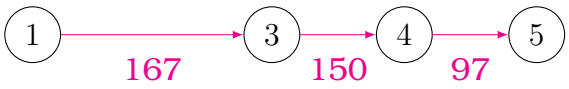
Ahora bien, la segunda columna de la instancia nos da un indicador por cada estación. Si el indicador de giro es igual a 0, eliminamos dicha estación y la demanda final es el máximo de las demandas de los arcos asociados. Para el ejemplo, consideramos cero a la estación 2. Por tanto el grafo resultante es:



Una vez que hemos agregado la demanda, procedemos a implementar el modelo no lineal. Para ello, dado que tenemos términos discontinuos en la función objetivo (3.12) introducimos un conjunto de variables auxiliares enteras $a_{i,j}$, $i, j \in V$ e incluimos la desigualdad $a_{ij} \geq cp_{ij} \cdot F_{ij}$ que resuelven los inconvenientes del término $\lceil cp_{ij}\, F_{i,j} \rceil$, asegurando que el producto $cp_{ij}\, F_{i,j}$ tome un número entero.

Una vez implementado el modelo no lineal y tomando $fmax = 20, cmax = 3$, $ccap = 100$, $cfix = 150$, $ckm = 0$, $trkm = 1.5$, obtenemos una solución asociada al ejemplo anterior: $F_{15} = 1, C_{15} = 2$. Esto implica que el modelo sugiere implementar una línea desde la estación 1 hasta la estación 5 con frecuencia igual a 1 usando el tipo de bus asociado a un articulado cuya capacidad es de 200 pasajeros.

Solución



$F_{15} = 1, C_{15} = 2$

4.2. Redes lineales y árboles

En la presente sección se reportan experimentos computacionales realizados con el modelo no lineal y lineal sobre redes de tipo camino y tipo árbol. Se dispone de 44 instancias de distintos tamaños $|V| \in \{30, 40, 50, 60, 70, 80, 90, 100\}$. Se ha fijado un tiempo máximo de ejecución igual a 3600 segundos. Los cuadros 4.1 y 4.2 presentan los resultados obtenidos para redes lineales, mientras que los cuadros 4.3 y 4.4 reportan los experimentos cuando la red tiene estructura de árbol.

Se asignaron los siguientes valores a cada uno de los parámetros del modelo: $cmax = 3$, $ccap = 100$, $cfix = 150$, $ckm = 0$, $trkm = 1.5$, $fmax = 20$ si el número de estaciones es igual a 30, 40 o 50 y $fmax = 30$ si corresponde a las instancias con 60, 70, 80, 90 o 100 estaciones.

Cuadro 4.1: Grafos lineales

	Modelo lineal						**Modelo no lineal**					
Instancia	F. Obj.	# Vars.	# Restr.	# Nodos	% Gap	Tiempo (seg)	F.Obj.	# Vars.	# Restr.	# Nodos	% Gap	Tiempo (seg)
M_aleatoria_30_1	2488.90	10260	189	1	0.00	1.55	2488.90	513	171	43996	0.00	169.38
M_aleatoria_30_2	2463.57	10260	189	1	0.00	0.70	2463.57	513	171	40725	0.00	174.02
M_aleatoria_30_3	2487.26	10260	189	1	0.00	0.97	2487.26	513	171	122084	0.00	313.50
M_aleatoria_30_4	2535.47	10260	189	1	0.00	0.64	2535.47	513	171	37127	0.00	127.14
M_aleatoria_30_5	2593.20	10260	189	1	0.00	0.65	2593.20	513	171	819604	2.33	3600.52
M_aleatoria_40_1	3924.05	18000	324	6102	0.00	24.86	3944.93	900	300	387701	7.33	3600.07
M_aleatoria_40_2	4296.72	18000	324	2119	0.00	28.06	4296.72	900	300	414991	8.37	3600.09
M_aleatoria_40_3	3860.00	18000	324	1	0.00	5.37	3860.00	900	300	924306	7.55	3600.07
M_aleatoria_40_4	3956.93	18000	324	945	0.00	5.88	3973.31	900	300	512034	6.51	3600.05
M_aleatoria_40_5	4009.99	18000	324	330	0.00	10.53	4009.99	900	300	738200	7.77	3600.06
M_aleatoria_50_1	5609.28	27900	495	2868	0.00	36.78	5630.69	1395	465	828558	8.87	3600.07
M_aleatoria_50_2	5752.46	27900	495	508	0.0 0	23.43	5752.46	1395	465	730351	6.71	3600.09
M_aleatoria_50_3	5622.18	27900	495	6399	0.0 0	56.24	5655.13	1395	465	731541	8.93	3600.08
M_aleatoria_50_4	5585.62	27900	495	9727	0.00	52.78	5854.68	1395	465	763105	53.40	3600.39
M_aleatoria_50_5	5610.59	27900	495	6566	0.0 0	37.30	5930.25	1395	465	991597	13.78	3600.38
M_aleatoria_60_1	6917.26	59940	702	1162601	1.03	3600.56	7196.14	1998	666	692780	52.42	3600.28
M_aleatoria_60_2	7377.69	59940	702	341790	3.30	3600.68	7817.44	1998	666	620048	54.07	3600.14
M_aleatoria_60_3	7753.42	59940	702	18173	0.00	323.98	8119.94	1998	666	570468	10.91	3600.22
M_aleatoria_60_4	7397.24	59940	702	335946	4.02	3600.89	7404.92	1998	666	563832	8.78	3600.34
M_aleatoria_60_5	7425.95	59940	702	439182	2.89	3600.83	7435.79	1998	666	600331	51.05	3600.27
M_aleatoria_70_1	9677.30	81270	945	534550	0.00	1726.53	10096.05	2709	903	421684	9.51	3600.83
M_aleatoria_70_2	9567.23	81270	945	367805	1.27	3600.60	9793.10	2709	903	430826	50.50	3600.79

Cuadro 4.2: Grafos lineales

	Modelo lineal						**Modelo no lineal**					
Instancia	F. Obj.	# Vars.	# Restr.	# Nodos	% Gap	Tiempo (seg)	F.Obj.	# Vars.	# Restr.	# Nodos	% Gap	Tiempo (seg)
M_aleatoria_70_3	10340.26	81270	945	247844	4.17	3600.80	10404.96	2709	903	466219	8.53	3600.33
M_aleatoria_70_4	9669.89	81270	945	103063	0.00	453.80	10101.66	2709	903	471098	9.09	3600.38
M_aleatoria_70_5	10262.16	81270	945	335066	4.02	3601.35	10391.27	2709	903	394798	8.56	3600.34
M_aleatoria_80_1	12395.82	105840	1224	200824	0.03	3600.82	13006.11	3528	1176	376579	10.33	3600.20
M_aleatoria_80_2	13240.37	105840	1224	149043	3.35	3601.04	13291.10	3528	1176	379605	7.71	3600.36
M_aleatoria_80_3	12597.17	105840	1224	249405	1.10	3600.33	13181.83	3528	1176	319072	7.98	3600.50
M_aleatoria_80_4	13109.17	105840	1224	226613	4.64	3600.71	13157.72	3528	1176	339766	50.10	3600.84
M_aleatoria_80_5	13038.85	105840	1224	166495	4.64	3601.16	13702.41	3528	1176	349531	11.90	3600.20
M_aleatoria_90_1	15425.86	133650	1539	68447	1.50	3600.66	16309.65	4455	1485	212251	50.81	3600.32
M_aleatoria_90_2	15311.79	133650	1539	147642	2.58	3600.85	15930.02	4455	1485	224181	8.65	3600.54
M_aleatoria_90_3	15993.71	133650	1539	95778	4.17	3601.56	16995.73	4455	1485	271385	52.48	3600.35
M_aleatoria_90_4	15373.34	133650	1539	126905	2.64	3601.49	15946.53	4455	1485	227041	60.63	3600.26
M_aleatoria_90_5	16242.52	133650	1539	105747	3.47	3601.56	17386.56	4455	1485	243732	52.27	3600.81
M_aleatoria_100_1	19128.09	164700	1890	63239	3.08	3600.95	19768.22	5490	1830	88006	7.67	3600.89
M_aleatoria_100_2	20062.80	164700	1890	57698	3.44	3601.24	21134.21	5490	1830	175815	51.24	3600.39
M_aleatoria_100_3	18203.60	164700	1890	66877	2.61	3601.72	19597.73	5490	1830	180078	52.06	3600.69
M_aleatoria_100_4	19105.01	170190	1952	35632	3.29	3601.12	20540.29	5673	1891	167552	52.20	3603.41
M_aleatoria_100_5	20180.58	170190	1952	55521	2.57	3603.83	21440.76	5673	1891	127418	9.66	3600.72
M_aleatoria_100_7	20136.93	164700	1890	51595	3.13	3601.61	20755.02	5490	1830	85811	7.30	3600.44
M_aleatoria_100_8	19316.67	170190	1952	32391	2.17	3601.32	20148.74	5673	1891	113440	7.75	3600.75
M_aleatoria_100_9	19296.26	164700	1890	35497	2.23	3601.36	20056.00	5490	1830	186527	7.50	3600.37
M_aleatoria_100_10	20110.13	164700	1890	57912	3.09	3601.00	21259.35	5490	1830	170070	51.17	3600.33

Para el caso en el que el sistema es un camino, podemos ver que para las instancias M_aleatoria_30_1, M_aleatoria_30_2, M_aleatoria_30_3 y M_aleatoria_30_4, que corresponden a instancias de 30 estaciones, tanto el modelo lineal como el no lineal encuentran la solución óptima, pues el gap reportado es del 0 % y los valores de la función objetivo son iguales. Además se puede ver que el tiempo de ejecución en el modelo no lineal es superior al tiempo del modelo lineal. En la instancia M_aleatoria_30_5 el modelo no lineal se ejecutó en el tiempo máximo asignado y muestra un gap del 2.33 % lo que significa que el solver no encontró la solución óptima. Sin embargo, la función objetivo hallada coincide con la hallada por el modelo lineal.

Para las instancias M_aleatoria_40_2 y M_aleatoria_40_3 se puede notar que se alcanza el valor óptimo, aunque en el modelo no lineal la brecha continúa con un valor distinto de cero. Vemos también que aunque en el modelo no lineal el número de variables y restricciones es menor que en el modelo lineal, el número de nodos generados por el algoritmo Branch and Bound en el modelo no lineal es superior al número de nodos en el modelo lineal. Para las instancias M_aleatoria_40_1, M_aleatoria_40_3 y M_aleatoria_40_5 vemos que se alcanza el tiempo máximo de cálculo igual a 3600 segundos con el modelo no lineal y el gap está sobre el 1 %. Por otro lado, el modelo lineal encuentra las soluciones con un gap del 0 % y en un período de tiempo muy corto.

Vemos que para las instancias de tamaño 50 estaciones el modelo lineal encuentra el valor óptimo en un tiempo promedio igual a 41.31 segundos. El comportamiento del modelo no lineal es completamente distinto ya que se usa los 3600 segundos y se tiene un gap promedio de 18.34 %. Las únicas instancias en las que se alcanza el valor óptimo son M_aleatoria_50_2 pese a que el gap en el modelo no lineal es del 6.71 %.

Algo muy parecido sucede para las instancias de 60, 70, 80 y 90 estaciones, pues en el modelo lineal se reportan valores de gap cercanos a cero y se podría considerar que la solución hallada está muy cercana al óptimo respetando el tiempo límite de ejecución que hemos impuesto. Como era de esperarse, el modelo no lineal nos da valores de gap más elevados en un tiempo de ejecución de 3600 segundos, por lo que podemos concluir que el valor objetivo no es el mejor y por ende la solución no es la óptima.

Finalmente, para las instancias de 100 estaciones vemos que en ningún caso se llega a un valor de 0 % de gap. Sin embargo, los valores que nos reporta el modelo lineal son muy bajos por lo que la solución se puede considerar aceptable. Esto no sucede para el modelo no lineal, pues en tres de las 9 instancias obtenemos un porcentaje de gap superior al 50 %, lo que implica que el valor objetivo encontrado esta bastante distante del valor óptimo.

En los cuadros 4.3 y 4.4 se pueden encontrar las soluciones cuando la red tiene estructura de árbol. Es fácil notar que las instancias se vuelven mucho más simples para el modelo lineal, ya que en la gran mayoría de ellas se alcanza la solución óptima. Por otro lado, el comportamiento del modelo no lineal es totalmente opuesto al anterior, pues en todas las instancias no se reporta la solución óptima e incluso se puede observar valores altos en la brecha de optimalidad (gap).

Cuadro 4.3: Grafos tipo árbol

	Modelo lineal						**Modelo no lineal**					
Instancia	F. Obj.	# Vars.	# Restr.	# Nodos	% Gap	Tiempo (seg)	F.Obj.	# Vars.	# Restr.	# Nodos	% Gap	Tiempo (seg)
M_aleatoria_30_1	2687.50	52200	899	1	0.00	0.32	2687.50	2610	870	818427	4.07	3600.18
M_aleatoria_30_2	2552.07	52200	899	1	0.01	0.38	2572.09	2610	870	870486	8.44	3600.33
M_aleatoria_30_3	2695.17	52200	899	1	0.00	0.35	2765.81	2610	870	709012	23.81	3600.15
M_aleatoria_30_4	2664.56	52200	899	1	0.00	0.28	2664.56	2610	870	810055	2.04	3600.24
M_aleatoria_30_5	2708.91	52200	899	1	0.00	0.35	2713.83	2610	870	644145	3.23	3600.12
M_aleatoria_40_1	3630.76	93600	1599	1	0.00	0.77	3892.23	4680	1560	606192	43.13	3600.20
M_aleatoria_40_2	3772.63	93600	1599	1	0.00	0.71	3842.54	4680	1560	362026	28.88	3600.32
M_aleatoria_40_3	3522.58	93600	1599	1	0.00	0.99	3821.68	4680	1560	317239	43.67	3600.22
M_aleatoria_40_4	3761.08	93600	1599	1	0.00	0.73	3797.32	4680	1560	368508	42.43	3600.26
M_aleatoria_40_5	3614.99	93600	1599	1	0.00	0.62	3748.36	4680	1560	417266	37.72	3600.38
M_aleatoria_50_1	4234.45	147000	2499	1	0.00	1.06	4377.07	7350	2450	146633	42.71	3600.49
M_aleatoria_50_2	4249.95	147000	2499	1	0.00	2.02	4596.06	7350	2450	157842	47.91	3600.25
M_aleatoria_50_3	4366.59	147000	2499	1	0.00	1.70	4547.30	7350	2450	194612	39.47	3600.50
M_aleatoria_50_4	4267.03	147000	2499	1	0.00	3.15	4585.75	7350	2450	203972	41.97	3600.16
M_aleatoria_50_5	4236.91	147000	2499	1	0.00	2.82	4421.98	7350	2450	180541	42.51	3600.17
M_aleatoria_60_1	4689.31	318600	3599	1	0.00	2.03	4966.34	10620	3540	101044	53.61	3600.55
M_aleatoria_60_2	4714.71	318600	3599	1	0.00	3.26	4960.36	10620	3540	123386	57.74	3600.34
M_aleatoria_60_3	4999.74	318600	3599	1	0.00	4.53	5667.98	10620	3540	104663	50.41	3600.37
M_aleatoria_60_4	4708.04	318600	3599	1	0.00	2.09	5365.05	10620	3540	111453	63.48	3600.34
M_aleatoria_60_5	4696.16	318600	3599	1	0.00	1.28	4882.77	10620	3540	121985	58.15	3600.64
M_aleatoria_70_1	5490.01	434700	4899	1	0.01	2.02	6060.44	14490	4830	63964	64.89	3600.10
M_aleatoria_70_2	5479.54	434700	4899	1	0.00	3.27	5479.54	14490	4830	88414	55.41	3600.52

Cuadro 4.4: Grafos tipo árbol

	Modelo lineal						**Modelo no lineal**					
Instancia	F. Obj.	# Vars.	# Restr.	# Nodos	% Gap	Tiempo (seg)	F.Obj.	# Vars.	# Restr.	# Nodos	% Gap	Tiempo (seg)
M_aleatoria_70_3	5472.91	434700	4899	1	0.00	2.76	5698.45	14490	4830	108802	61.19	3600.43
M_aleatoria_70_4	5502.36	434700	4899	1	0.00	4.61	5845.14	14490	4830	73939	63.55	3600.20
M_aleatoria_70_5	5489.60	434700	4899	1	0.01	4.79	5505.57	14490	4830	65280	55.39	3600.36
M_aleatoria_80_1	6229.04	568800	6399	1	0.00	5.53	6803.17	18960	6320	83944	75.28	3600.58
M_aleatoria_80_2	6236.01	568800	6399	1	0.01	5.04	6659.86	18960	6320	62242	68.78	3600.42
M_aleatoria_80_3	6221.35	568800	6399	1	0.00	3.58	6658.97	18960	6320	78039	73.78	3600.48
M_aleatoria_80_4	6728.01	568800	6399	1	0.01	6.13	7141.22	18960	6320	16308	49.81	3600.31
M_aleatoria_80_5	6552.65	568800	6399	1	0.00	4.94	6895.13	18960	6320	17387	52.73	3600.15
M_aleatoria_90_1	7029.21	720900	8099	1	0.00	4.44	7322.71	24030	8010	23352	64.17	3600.22
M_aleatoria_90_2	6976.95	720900	8099	1	0.01	4.04	6977.49	24030	8010	47991	74.09	3600.73
M_aleatoria_90_3	7020.37	720900	8099	1	0.00	11.17	7394.71	24030	8010	43288	68.78	3600.59
M_aleatoria_90_4	6958.31	720900	8099	1	0.00	6.99	7100.35	24030	8010	54354	75.10	3600.73
M_aleatoria_90_5	6994.17	720900	8099	1	0.00	3.87	7135.18	24030	8010	40040	72.81	3600.55
M_aleatoria_100_1	7734.34	891000	9999	1	0.01	5.09	7846.31	29700	9900	38566	70.92	3600.60
M_aleatoria_100_2	7766.78	891000	9999	1	0.01	13.20	8058.86	29700	9900	39807	72.56	3600.28
M_aleatoria_100_3	7799.02	891000	9999	1	0.00	14.16	7800.30	29700	9900	34954	69.23	3600.32
M_aleatoria_100_4	7720.48	891000	9999	1	0.00	13.27	8115.24	29700	9900	43631	73.68	3600.26
M_aleatoria_100_5	7727.02	891000	9999	1	0.01	12.03	8385.22	29700	9900	16140	72.77	3600.16
M_aleatoria_100_7	7748.29	891000	9999	1	0.00	13.12	7896.84	29700	9900	52256	72.75	3600.26
M_aleatoria_100_8	7731.20	891000	9999	1	0.00	13.85	7872.52	29700	9900	37905	74.13	3600.22
M_aleatoria_100_9	7733.65	891000	9999	1	0.00	5.23	7733.65	29700	9900	68003	75.24	3600.19
M_aleatoria_100_10	7755.19	891000	9999	1	0.00	17.02	7898.56	29700	9900	58342	74.28	3600.76

Conclusiones y Recomendaciones

- El tema de estudio del presente trabajo es la planificación de líneas y frecuencias. Para ello se realizó la adaptación de las formulaciones presentadas en [10] para el sistema de transporte de ciudad de Quito. Se presentaron dos modelos de programación entera, un modelo lineal y un modelo no lineal.
- Se realizó un análisis de la complejidad del problema de planificación de líneas y frecuencias, obteniendo que dicho problema es NP-duro.
- Los modelos tanto lineal como no lineal fueron implementados sobre grafos lineales y grafos tipo árbol. Para los grafos lineales, el modelo lineal encuentra la solución con una brecha de optimalidad mínima, esta brecha toma valores más altos con el modelo no lineal. Un comportamiento similar se observó con los grafos tipo árboles. Esto permite concluir que el modelo lineal es más eficiente que el modelo no lineal.
- Para los grafos árboles el modelo lineal encuentra la solución óptima, mientras que para el modelo no lineal, en la mayoría de instancias propuestas, el gap toma valores superiores al 50 %. Con esto se concluye que la eficiencia del modelo depende de la estructura del grafo.

Referencias

[1] Jim Orlin. Nonlinear Programming [Notas de curso]. https://web.mit.edu/15.053/www/AMP-Chapter-13.pdf, 2019.

[2] Sebnem Yilmaz Balaman. *Decision-Making for Biomass-Based Production Chains: The Basic Concepts and Methodologies*. Academic Press, 2018.

[3] Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli, et al. *Integer programming*, volume 271. Springer, 2014.

[4] George B. Dantzig. Origins of the simplex method. Technical Report SOL-87-5, Department of Operations Research - SOL Stanford University, Stanford, CA 94305, may 1987.

[5] Gurobi Optimization. Gurobi Optimizer Technical Features and Details. https://www.gurobi.com, 2022.

[6] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, nov 1984.

[7] Peter Gács, Laszlo Lovász,. Khachiyan's algorithm for linear programming. *Mathematical Programming Study*, 14:61–68, 1981.

[8] Marco E. Lübbecke, Michael R. Bussieck, Thomas Lindner. A fast algorithm for near cost optimal line plans. *Mathematical Methods of Operations Research*, 59:205–220, nov 2004.

[9] Uwe T. Zimmermann, Michael R. Bussieck, Peter Kreuzer. Optimal lines for railway systems. *European Journal of Operational Research*, 96(1):54–63, jan 1997.

[10] P.J.Zwaneveld, M.T.Claessens, N.M.van Dijk. Cost optimal allocation of rail passenger lines. *European Journal of Operational Research*, 110(3):474–489, nov 1998.

[11] Marc E. Pfetsch, Ralf Borndörfer, Martin Grötschel. Models for line planning in public transport. *Springer*, 600 of Lecture Notes in Economics and Mathematical Systems book series:363–378, jan 2008.

[12] U. T. Zimmermann and T. Linder. Train schedule optimization in public rail transport. *Technical report*, jun 2000.

Anexos

Código 1: Paquetes

```
import pandas as pd
import numpy as np
from collections import OrderedDict
from gurobipy import *
import math as op
import csv
```

Código 2: Modelo no lineal

```
Z=[]
def Modelo_no_lineal(instancia):
    #lectura de datos
    datos1=pd.read_csv("C:/Users/yomai/Desktop/Proyecto/FINAL/Matriz_OD/"+instancia, sep=" ", skiprows=1, header=None)
    #datos1=pd.read_csv("C:/Users/yomai/Desktop/Proyecto/input.txt", sep=" ", skiprows=1, header=None)
    datos = datos1.dropna(1)

    #numero de estaciones
    n_estaciones = int(pd.read_csv("C:/Users/yomai/Desktop/Proyecto/FINAL/Matriz_OD/"+instancia,sep=" ").columns[0])
    #n_estaciones = int(pd.read_csv("C:/Users/yomai/Desktop/Proyecto/input.txt",sep=" ").columns[0])

    # nombres de las estaciones
    V1=datos[0]
    E1=[]
    for i in range(0,n_estaciones-1):
        E1.append((V1[i],V1[i+1]))
    ind=datos[1]
    distancia1 = list(datos[2])
    t_circulacion1=list(datos[2]*0.17)
    f_max1=[]
    f_max1.append(0)
    for i in range(1,n_estaciones):
        f_max1.append(f_max1[i-1]+10)
    dem1=datos.loc[0:,3:]
    dem=dem1.to_numpy()
```

```
26
27      # cargar de la demanda en la red
28      M=[]
29      w=[]
30      n1={}
31      for i in range(0,n_estaciones-1):
32          a=[]
33          for j in range(i+1,n_estaciones):
34              for k in range(0,n_estaciones-1):
35                  if i-k>=0:
36                      a.append(max(dem[i-k][j],dem[j][i-k]))
37          M.append(a)
38          n1[V1[i],V1[i+1]]=sum(M[i])
39      # Eliminar estaciones de acuerdo al indicador
40      def eliminar_estaciones(V1,ind):
41          indicadores=dict(zip(V1,ind))
42          indicadores2=dict(filter(lambda x: x[1]!=0, indicadores.items()))
43          nuevosnodos=list(indicadores2.keys())
44          nuevosindicadores=list(indicadores2.values())
45          E=[]
46          for i in list(range(0,len(nuevosnodos)-1)):
47              E.append((nuevosnodos[i],nuevosnodos[i+1]))
48          aux=[]
49          for i in range(0,len(E)):
50              aux.append(E[i][0])
51              aux.append(E[i][1])
52          V=[]
53          for i in aux:
54              if i not in V:
55                  V.append(i)
56          L=[]
57          for i in range(0,n_estaciones-1):
58              A={}
59              for j in range(i+1,n_estaciones):
60                  if (V1[i],V1[j]) in E1:
61                      A[V1[i],V1[j]]=n1[V1[i],V1[j]]
62                  else:
63                      A[V1[i],V1[j]]=n1[V1[j-1],V1[j]]
64              L.append(A)
65          L1_aux=[]
66          L2_aux=[]
67          for i in range(0,len(L)):
68              for j in range(0,len(L[i])):
69                  for k in E:
70                      if list(L[i].keys())[j]==k:
71                          L1_aux.append(k)
72                          L2_aux.append(max(list(L[i].values())[:j+1]))
73          n=dict(zip(L1_aux,L2_aux))
74          return V,E,n
75      V,E,n=eliminar_estaciones(V1,ind)
76
77      dist=[]
78      t_=[]
79      f_=[]
80      for i in range(0,n_estaciones):
81          if ind[i]!=0:
```

```
            dist.append(distancia1[i])
            t_.append(t_circulacion1[i])
            f_.append(f_max1[i])
    distancia1=dist
    t_circulacion1=t_
    f_max1=f_

    distancia = {}
    for i in range(0,len(distancia1)):
        for j in range(i+1, len(distancia1)):
            distancia[V[i],V[j]]=distancia1[j]-distancia1[i]

    cp = {}
    for i in range(0,len(t_circulacion1)):
        for j in range(i+1, len(t_circulacion1)):
            cp[(V[i],V[j])]=(t_circulacion1[j]-t_circulacion1[i])/60

    # valores de los parametros
    if n_estaciones==30 or n_estaciones==40 or n_estaciones==50:
        fmax=20
    else:
        fmax=30
    cmin=1
    cmax=3
    cfix=150
    ckm=0
    trkm=1.5
    ccap=100

    #---------------------------MODELO NO LINEAL-------------------------------
    m=Model('Modelo_no_lineal')
    F={}
    C={}
    aux1={}
    aux2={}
    for i in range(0,len(V)-1):
        for j in range(i+1,len(V)):
            F[V[i],V[j]]=m.addVar(lb=0, ub=fmax, vtype=GRB.INTEGER,name = "F_{}_{}".
    format(i,j))
            C[V[i],V[j]]=m.addVar(lb=0, ub=cmax, vtype=GRB.INTEGER,name = "C_{}_{}".
    format(i,j))
            aux1[V[i],V[j]]=m.addVar(vtype=GRB.INTEGER,name = "aux1_{}_{}".format(i,j))

    m.addConstrs((aux1[V[i],V[j]] >= cp[V[i],V[j]]*F[V[i],V[j]] for i in range(0,len(V)
    -1) for j in range(i+1,len(V))),"linealizacion")

    costo_fijo = quicksum(cfix*aux1[V[i],V[j]]*C[V[i],V[j]] for i in range(0,len(V)-1)
    for j in range(i+1,len(V)))
    costo_coche = quicksum(distancia[V[i],V[j]]*ckm*F[V[i],V[j]]*C[V[i],V[j]] for i in
    range(0,len(V)-1) for j in range(i+1,len(V)))
    costo_com = quicksum(distancia[V[i],V[j]]*trkm*F[V[i],V[j]] for i in range(0,len(V)
    -1) for j in range(i+1,len(V)))
    m.setObjective(costo_fijo+costo_coche+costo_com)

    m.addConstrs((quicksum(ccap*F[V[i],V[j]]*C[V[i],V[j]] for i in range(0,len(V)-1) for
     j in range(i+1,len(V)) if i<=l and l+1<=j)>=n[V[l],V[l+1]] for l in range(0,len(V)
```

```
    -1)),"Restriccion2")
    m.update()

    #fijar parametros
    m.Params.LogToConsole = 0
    m.Params.LogFile= str(instancia[:-4])+"(no_lineal_1)"
    m.Params.TimeLimit = 3600
    m.params.NonConvex=2
    m.optimize()

    # informacion a extraer
    Instancia=instancia[:-4]
    V_objetivo=round(m.ObjVal,2)
    N_variables=m.NumVars
    N_restricciones=m.NumConstrs
    GAP=round(m.MIPGap,2)
    Tiempo=round(m.Runtime,2)
    N_nodos=m.NodeCount
    aux1=[Instancia,V_objetivo,N_variables,N_restricciones,N_nodos, GAP,Tiempo]
    W1=[]
    for i in range(0,len(aux1)):
        W1.append(aux1[i])
    Z.append(W1)

    #------------------------------ SOLUCION -----------------------------------
    print('El valor optimo encontrado es:',m.objVal)
    # Recuperar los valores de las variables
    vF=m.getAttr('x',F)
    vC=m.getAttr('x',C)
    for i in range(0,len(V)-1):
        for j in range(i+1,len(V)):
            if vF[V[i],V[j]] > 0.1 and vC[V[i],V[j]] > 0.1:
                print('({},{}), F:{}, C:{}'.format(V[i],V[j],vF[V[i],V[j]],vC[V[i],V[j
    ]]))
```

```
datos_entrada = os.listdir("C:/Users/yomai/Desktop/Proyecto/FINAL/Matriz_OD/")
for i in range(0,len(datos_entrada)):
    Modelo_no_lineal(datos_entrada[i])
with open('Resultados_no_lineal.csv', 'w', newline='') as f:
    for i in range(0,len(Z)):
        writer = csv.writer(f)
        writer.writerow(Z[i])
```

Código 3: Modelo lineal

```
W=[]
def Modelo_lineal(instancia):
    #lectura de datos
    datos1=pd.read_csv("C:/Users/yomai/Desktop/Proyecto/FINAL/Matriz_OD/"+instancia, sep
    =" ", skiprows=1, header=None)
    datos = datos1.dropna(1)

    #numero de estaciones
    n_estaciones = int(pd.read_csv("C:/Users/yomai/Desktop/Proyecto/FINAL/Matriz_OD/"+
    instancia,sep=" ").columns[0])
```

```
9
10      # nombres de las estaciones
11      V1=datos[0]
12      E1=[]
13      for i in range(0,n_estaciones-1):
14          E1.append((V1[i],V1[i+1]))
15      ind=datos[1]
16      distancia1 = list(datos[2])
17      t_circulacion1=list(datos[2]*0.17)
18      f_max1=[]
19      f_max1.append(0)
20      for i in range(1,n_estaciones):
21          f_max1.append(f_max1[i-1]+10)
22      dem1=datos.loc[0:,3:]
23      dem=dem1.to_numpy()
24      # Carga de la demanda en la red
25      M=[]
26      w=[]
27      n1={}
28      for i in range(0,n_estaciones-1):
29          a=[]
30          for j in range(i+1,n_estaciones):
31              for k in range(0,n_estaciones-1):
32                  if i-k>=0:
33                      a.append(max(dem[i-k][j],dem[j][i-k]))
34          M.append(a)
35          n1[V1[i],V1[i+1]]=sum(M[i])
36
37      # elinimar estaciones de acuerdo al indicador
38      def eliminar_estaciones(V1,ind):
39          indicadores=dict(zip(V1,ind))
40          indicadores2=dict(filter(lambda x: x[1]!=0, indicadores.items()))
41          nuevosnodos=list(indicadores2.keys())
42          nuevosindicadores=list(indicadores2.values())
43          E=[]
44          for i in list(range(0,len(nuevosnodos)-1)):
45              E.append((nuevosnodos[i],nuevosnodos[i+1]))
46          aux=[]
47          for i in range(0,len(E)):
48              aux.append(E[i][0])
49              aux.append(E[i][1])
50          V=[]
51          for i in aux:
52              if i not in V:
53                  V.append(i)
54          L=[]
55          for i in range(0,n_estaciones-1):
56              A={}
57              for j in range(i+1,n_estaciones):
58                  if (V1[i],V1[j]) in E1:
59                      A[V1[i],V1[j]]=n1[V1[i],V1[j]]
60                  else:
61                      A[V1[i],V1[j]]=n1[V1[j-1],V1[j]]
62              L.append(A)
63          L1_aux=[]
64          L2_aux=[]
```

```
        for i in range(0,len(L)):
            for j in range(0,len(L[i])):
                for k in E:
                    if list(L[i].keys())[j]==k:
                        L1_aux.append(k)
                        L2_aux.append(max(list(L[i].values())[:j+1]))
        n=dict(zip(L1_aux,L2_aux))
        return V,E,n

    V,E,n=eliminar_estaciones(V1,ind)

    dist=[]
    t_=[]
    f_=[]
    for i in range(0,n_estaciones):
        if ind[i]!=0:
            dist.append(distancia1[i])
            t_.append(t_circulacion1[i])
            f_.append(f_max1[i])
    distancia1=dist
    t_circulacion1=t_
    f_max1=f_

    distancia = {}
    for i in range(0,len(distancia1)):
        for j in range(i+1, len(distancia1)):
            distancia[V[i],V[j]]=distancia1[j]-distancia1[i]

    cp = {}
    for i in range(0,len(t_circulacion1)):
        for j in range(i+1, len(t_circulacion1)):
            cp[(V[i],V[j])]=(t_circulacion1[j]-t_circulacion1[i])/60

    # valores de los parametros
    if n_estaciones==30 or n_estaciones==40 or n_estaciones==50:
        fmax=20
    else:
        fmax=30
    cmin=1
    cmax=3
    cfix=150
    ckm=0
    trkm=1.5
    ccap=100
    F=tuplelist(range(1,fmax+1))
    C=tuplelist(range(1,cmax+1))

    #----------------------------MODELO LINEAL ---------------------------------
    m=Model('Modelo_lineal')
    X={}
    for f in F:
        for c in C:
            for i in range(0,len(V)-1):
                for j in range(i+1,len(V)):
                    X[V[i],V[j],f,c]=m.addVar(vtype=GRB.BINARY,name = "X_{}_{}_{}_{}".
    format(V[i],V[j],f,c))
```

```
120
121     # funcion objetivo
122     costo_fijo = quicksum(cfix*op.ceil(cp[V[i],V[j]]*f)*c*X[V[i],V[j],f,c] for f in F
        for c in C for i in range(0,len(V)-1) for j in range(i+1,len(V)))
123     costo_coche = quicksum(distancia[V[i],V[j]]*ckm*f*c*X[V[i],V[j],f,c] for f in F for
        c in C for i in range(0,len(V)-1) for j in range(i+1,len(V)) )
124     costo_comp = quicksum(distancia[V[i],V[j]]*trkm*f*X[V[i],V[j],f,c] for f in F for c
        in C for i in range(0,len(V)-1) for j in range(i+1,len(V)))
125     m.setObjective(costo_fijo+costo_coche+costo_comp, GRB.MINIMIZE)
126
127     #restricciones
128     m.addConstrs((quicksum(ccap*f*c*X[V[i],V[j],f,c] for f in F for c in C  for i in
        range(0,len(V)-1) for j in range(i+1,len(V)) if i<=l and l+1<=j)>=n[V[l],V[l+1]] for
         l in range(0,len(V)-1)),"Restriccion8")
129     m.addConstrs((quicksum(X[V[i],V[j],f,c] for f in F for c in C)<=1 for i in range(0,
        len(V)-1) for j in range(i+1,len(V))),"restriccion10")
130
131     #fijar parametros
132     m.Params.LogToConsole =0
133     m.Params.LogFile= str(instancia[:-4])+"(lineal_1)"
134     m.Params.TimeLimit = 3600
135     m.optimize()
136
137     # informacion a extraer
138     Instancia=instancia[:-4]
139     V_objetivo=round(m.ObjVal,2)
140     N_variables=m.NumVars
141     N_restricciones=m.NumConstrs
142     GAP=round(m.MIPGap*100,2)
143     Tiempo=round(m.Runtime,2)
144     N_nodos=m.NodeCount
145     aux1=[Instancia,V_objetivo,N_variables,N_restricciones,N_nodos, GAP,Tiempo]
146     W1=[]
147     for i in range(0,len(aux1)):
148         W1.append(aux1[i])
149     W.append(W1)
150
151     # ------------------------------SOLUCION ---------------------------------------
152     print('El valor optimo encontrado es:',m.objVal)
153     # Recuperar los valores de las variables
154     vX=m.getAttr('x',X)
155     for f in F:
156         for c in C:
157             for i in range(0,len(V)-1):
158                 for j in range(i+1,len(V)):
159                     if vX[V[i],V[j],f,c] > 0.1 :
160                         print('({},{}), f:{}, c:{}'.format(V[i],V[j],f,c))
```

```
1 datos_entrada = os.listdir("C:/Users/yomai/Desktop/Proyecto/FINAL/Matriz_OD/")
2 for i in range(0,len(datos_entrada)):
3     Modelo_lineal(datos_entrada[i])
4 with open('Resultados_lineal.csv', 'w', newline='') as f:
5     for i in range(0,len(W)):
6         writer = csv.writer(f)
7         writer.writerow(W[i])
```