ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE CIENCIAS

OPTIMIZACIÓN EN SISTEMAS DE TRANSPORTE PÚBLICO.

MODELO INTEGRADO PARA EL PROBLEMA DE PLANIFICACIÓN DE LÍNEAS Y ENRUTAMIENTO DE PASAJEROS

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

MADELAINE STEFANY OBANDO JARAMILLO

madelaine.obando@epn.edu.ec

DIRECTOR: RAMIRO DANIEL TORRES GORDILLO

ramiro.torres@epn.edu.ec

DMQ, AGOSTO 2022

CERTIFICACIONES

Yo, MADELAINE STEFANY OBANDO JARAMILLO, 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.

Madelaine Stefany Obando Jaramillo

Certifico que el presente trabajo de integración curricular fue desarrollado por Madelaine Stefany Obando Jaramillo, bajo mi supervisión.

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(los) producto(s) resultante(s) del mismo, es(son) público(s) y estará(n) 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.

Madelaine Stefany Obando Jaramillo

Ramiro Daniel Torres Gordillo

RESUMEN

La planificación estratégica de un sistema de transporte público consiste de tres importantes fases: Diseño de red, planificación de líneas y frecuencias y el diseño del diagrama de marcha. El presente trabajo se centrará en estudiar el problema de planificación de líneas y frecuencias (LPP) en el sistema de transporte público de la ciudad de Quito. LPP consiste en encontrar un conjunto de líneas con sus respectivas frecuencias de tal forma que la demanda de transporte quede satisfecha. Generalmente el problema considera optimizar 2 grandes objetivos: el primero es maximizar la comodidad de los pasajeros a través de viajes directos y el segundo objetivo pretende minimizar los costos de operación. En ciertos estudios previos, la asignación de demanda se la realiza en un paso previo y el diseño de las líneas y frecuencias depende fuertemente de la forma en la que realizó dicha tarea. En el presente estudio se pretende estudiar el problema en la que el proceso de asignación de pasajeros y diseño de líneas y frecuencias se genere de forma integrada. Un modelo de programación lineal entera que minimiza el número de transferencias de los pasajeros y el costo total del operador es presentado. Además, una versión simplificada en la que no se permiten transferencias para los pasajeros es estudiada.

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

ABSTRACT

The strategic planning of a public transport system consists of three important phases: network design, line planning and timetable. The present work is focused on studying the problem of line planning problem (LPP) in the public transport system of Quito. LPP consists of finding a set of lines with their respective frequencies in such a way that the transport demand is satisfied. Usually this problem considers to optimize 2 objectives: the first one maximizes passenger comfort through direct trips and the second one aims to minimize the operational costs. In previous studies, the passenger routing is performed in a previous step and the design of the lines and frequencies depends strongly on the way in which such a task was performed. In the present study the integrated problem of line planning and passengers routing is consedered. An integer linear programming model that minimizes the number of transfers and the total cost for the operator is presented. Also, a simplified version in which passenger transfers are not allowed is studied.

Keywords: Line planning problem,passenger routing, integer linear programming, public transport

Índice general

Índice de figuras

Capítulo 1

Introducción

El proceso de planificación de un sistema de transporte público consiste en algunos problemas de optimización que deben ser resueltos en forma secuencial: diseño de la red, planificación de líneas y frecuencias, diseño del diagrama de marcha, asignación de flota y asignación de conductores. Los tres primeros se enmarcan en la fase conocida como planificación estratégica, mientras que los problemas restantes son incluidos en el proceso de planificación operacional.

Para Bussieck la planificación estratégica "se centra en la adquisición de recursos necesarios para poder operar durante un periodo futuro"Bussieck (1997). El diseño de la red consiste en determinar las rutas y calles por las cuales se podría mover el bus para generar las líneas de la red de transporte, la planificación de líneas y frecuencias se encarga de organizar cuáles líneas serán usadas por los buses. Finalmente, el diagrama de marcha definirá el horario de despacho de los buses de cada uno de los terminales.

En el presente trabajo nos enfocaremos en el problema integrado de planificación de líneas y frecuencias y enrutamiento de pasajeros. Dicho problema consiste en encontrar un conjunto de líneas con sus respectivas frecuencias y enrutar a los pasajeros de manera que la demanda sea satisfecha. Se plantea minimizar los costos de operación y maximizar la comodidad de los pasajeros.

El problema de planificación de líneas (LPP) ha sido modelado de diferentes maneras buscando optimizar varias funciones objetivo. En 1997, Bussieck (1997) formuló un modelo de programación entera mixta enfocado en la maximización de viajes directos. En 1998, Claessens et al. (1998) propusieron una formulación no lineal con el objetivo de minimizar los costos de operación. En 2004, Goossens and Kroon (2004) reportaron un algoritmo de solución tipo de Branch & Cut. Además, los autores reportaron técnicas de pre-procesamiento y fortalecimiento del modelo lineal formulado. En 2014, Ratajczak and Kasprzak (2014) diseñaron un algoritmo híbrido que es comparado con un método de recocido simulado y búsqueda aleatoria. En 2015, Borndörfer and Karbstein (2015) introdujeron un modelo de programación lineal entero mixto que distingue los viajes directos y los viajes no directos con al menos una transferencia.

El problema LPP integrado busca encontrar un conjunto de líneas y frecuencias junto con la asignación de pasajeros a las líneas incluidas en la solución, es decir, se seleccionan las líneas y se conoce las posibles rutas por las que podrían trasladarse cada pasajero dentro de la red. La función objetivo minimiza los costos de operación y se incluye un nuevo factor que minimiza las transferencias de los pasajeros, con lo cual se pretende incrementar la calidad del servicio.

El presente trabajo de integración curricular se encuentra organizado de la siguiente manera: en el Capítulo 2 se encuentran las definiciones y conceptos importantes para la comprensión del problema, en el Capítulo 3 se presenta el modelo integrado para el problema de planificación de líneas y frecuencias y enrutamiento de pasajeros, mientras que en el Capítulo 4 se muestran los resultados computacionales. Finalmente, el Capítulo 5 reporta las conclusiones y recomendaciones del trabajo.

Capítulo 2

Descripción del componente desarrollado

El presente proyecto pretende explorar en la literatura diferentes modelos matemáticos de optimización discreta y métodos de solución para el problema de planificación de líneas y enrutamiento de pasajeros, con el propósito de replicar y extender los resultados reportados en Torres (2021).

2.1. Objetivos

2.1.1. Objetivo general

Estudiar el problema integrado de planificación de líneas y enrutamiento de pasajeros. Replicar los resultados del artículo "Line planning and passenger routing problem with application to the Quito transportation system" Torres (2021) usando instancias de gran tamaño.

2.1.2. Objetivos específicos

1. Realizar una revisión bibliográfica de modelos integrados de planificación de líneas y frecuencias y enrutamiento de pasajeros.
2. Implementar un modelo integrado de planificación de lineas y frecuencias y enrutamiento de pasajeros usando el solver de progra-

mación lineal entera GUROBI.

3. Implementar un modelo simplificado de planificación de líneas y frecuencias y enrutamiento de pasajeros considerando solo viajes directos usando el solver de programación lineal entera GUROBI.

4. Conducir experimentos computacionales.

2.2. Alcance

En el presente proyecto no se pretende aportar con nuevas formulaciones para el problema estudiado, sino únicamente se pretende experimentar con los modelos propuestos previamente en la literatura y comparar los resultados obtenidos. Además, al trabajar con nuevas versiones del solver de programación lineal entera GUROBI, se pretende resolver instancias de mayor tamaño a las propuestas originalmente en el trabajo aprovechando la eficiencia de dichas nuevas versiones y los recursos computacionales actualmente disponibles.

Así, los modelos integrados considerando exclusivamente viajes directos y el modelo general considerando un número máximo de transferencias son implementados para posteriormente realizar pruebas computacionales con instancias de diferentes tamaños. Finalmente, se pretende comparar el comportamiento de dichas versiones respecto a las soluciones obtenidas y las diferencias en los costos totales.

2.3. Marco teórico

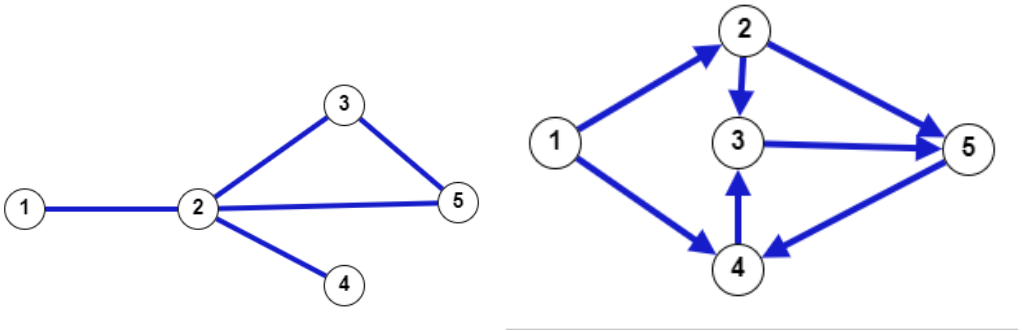
Para enfrentar matemáticamente el problema integrado de planificación de líneas y enrutamiento de pasajeros, es necesario conocer algunas definiciones y notaciones importantes. La presente sección esta basada en el trabajo desarrollado por Ravindra et al. (1993).

2.3.1. Teoría de Grafos

Es una de las ramas de la matemática cuyo origen se remota al siglo XVIII, siendo el matemático Leonhard Euler quien introdujo los fundamentos de la misma. Esta teoría busca representar de forma visual conjuntos de datos abstractos basados en nodos y aristas o arcos que representan la relación que existe entre los diferentes nodos. Una de las diversas aplicaciones de la teoría de grafos se encuentra en los sistemas de transporte. A continuación se enuncian algunas definiciones relevantes.

Un grafo no dirigido G al cual también llamaremos grafo (Figura 2.1a), es un par ordenado $G = (V, E)$, compuesto por un conjunto finito no vacío de vértices o nodos $V = \{v_1, v_2, ..., v_n\}$ y un conjunto de pares desordenados conocido como aristas $E = \{e_k = \{v_i, v_j\} : v_i, v_j \in V\}$. En un grafo no dirigido se tiene que $\{v_i, v_j\} = \{v_j, v_i\}$. Un camino es una secuencia de nodos $v_1, v_2, ..., v_n$ donde $\{v_i, v_{i+1}\} \in E$ para $i = 1, 2, ..., n-1$. El nodo v_1 se denomina origen y v_n es el nodo destino. Un circuito es un camino $\{v_1, v_2, ..., v_n\}$ en el que sus nodos extremos son iguales, es decir, $v_1 = v_n$.

Un grafo dirigido $D = (V, A)$ también conocido como dígrafo (Figura 2.1b), consiste en un conjunto finito de nodos V no vacío, y un conjunto de pares ordenados de nodos al que denominaremos conjunto de arcos $A = \{a_k = (v_i, v_j) : v_i, v_j \in V\}$. En un dígrafo se tiene que $(v_i, v_j) \neq (v_j, v_i)$. Las definiciones de camino y circuito para un dígrafo se interpretan de forma similar a las del caso no dirigido.



(a) Grafo no dirigido (b) Grafo Dirigido

Figura 2.1: Grafo y dígrafo

2.3.2. Programación Lineal

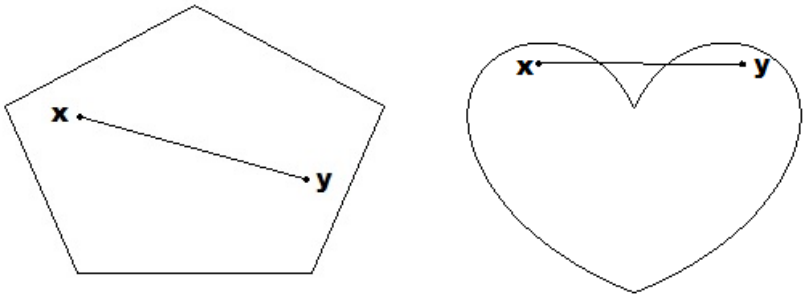
Es una técnica de optimización matemática que nació durante la Segunda Guerra Mundial (1939-1945). Fue desarrollada por el físico y matemático George Bernard Dantzig, quien utilizó la programación lineal para resolver diversos problemas de operación militar, con el objetivo de reducir los gastos para el ejército e incrementar las pérdidas para sus enemigos.

La programación lineal busca maximizar o minimizar una función lineal, llamada función objetivo sujeta a restricciones lineales. En esta sección presentamos algunos conceptos básicos de Programación Lineal y Programación Lineal Entera.

Definición 1. *Un conjunto $F \subset \mathbb{R}^n$ es convexo si para todo par de puntos $x, y \in F$ se tiene:*

$$\lambda x + (1 - \lambda)y \in F, \quad \lambda \in [0, 1]$$

La Figura (2.2a) muestra un ejemplo de un conjunto convexo mientras que en la Figura (2.2b) se puede apreciar un conjunto que no es convexo.



(a) Convexo (b) No Convexo

Figura 2.2: Ejemplos de convexidad

Definición 2. *Un problema de programación lineal (LP) consiste en optimizar (minimizar o maximizar) una función lineal de n variables de decisión $x_1, x_2, ..., x_n$ sujetas a un conjunto de restricciones expresadas mediante un conjunto de ecuaciones y desigualdades lineales. De manera formal, un LP puede ser expresado como se muestra en (2.1) y (2.2).*

$$\begin{aligned} & min \quad c^T x \\ & Sujeto \quad a: \\ & Ax \geq b \\ & x \geq 0 \end{aligned} \tag{2.1}$$

$$\begin{aligned} & max \quad c^T x \\ & Sujeto \quad a: \\ & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{2.2}$$

donde $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{n \times m}$ y $x \in \mathbb{R}^n$ con x diferente del vector nulo. Llamamos solución factible al vector x si y solo si cumple con todas las restricciones. Denominamos región factible al conjunto formado por todos los puntos $S = \{x \in \mathbb{R} : Ax \geq b\}$ para el caso (2.1) y $S = \{x \in \mathbb{R} : Ax \leq b\}$ para el caso presentado en (2.2). Si el conjunto S es vacío diremos que el problema es infactible, es decir, que no tiene solución.

Cuando se exige que alguna o todas las variables tomen valores enteros y se conserve las características de que la función objetivo y las restricciones sean lineales, entonces nos encontramos frente a un problema de programación lineal entero. Teniendo en cuenta los valores de las variables, se tienen 3 tipos de problemas lineales enteros:

1. **Problemas de programación entera mixta:** en estos problemas algunas variables toman valores continuos mientras que otras toman valores netamente enteros.

2. **Problemas de programación entera pura:** la característica de estos problemas es que todas sus variables toman valores enteros.

3. **Problemas de programación binaria o programación 0-1:** en este tipo de problemas las variables de decisión únicamente pueden tomar el valor de cero o uno.

Nos enfocaremos únicamente en problemas de programación lineal entera pura y nos referiremos a los mismos como problemas de programación lineal entera.

Un problema de programación lineal entera (IP) puede escribirse como:

$$\begin{aligned} max \quad & c^T x \\ Sujeto \quad & a: \\ & Ax \leq b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n \end{aligned} \tag{2.3}$$

Para este caso el conjunto de soluciones factibles estará dado por:

$$S' = \{x \in \mathbb{Z}^n : Ax \leq b\}$$

A continuación mostraremos con un ejemplo las dificultades que aparecen a la hora de calcular una solución óptima en un problema de programación lineal entera.

$$\begin{aligned} max \quad & z = x_1 + 2x_2 \\ Sujeto \quad & a: \\ & x_1 + x_2 \leq 5{,}4 \\ & x_1, x_2 \in \mathbb{Z}^+ \end{aligned} \tag{2.4}$$

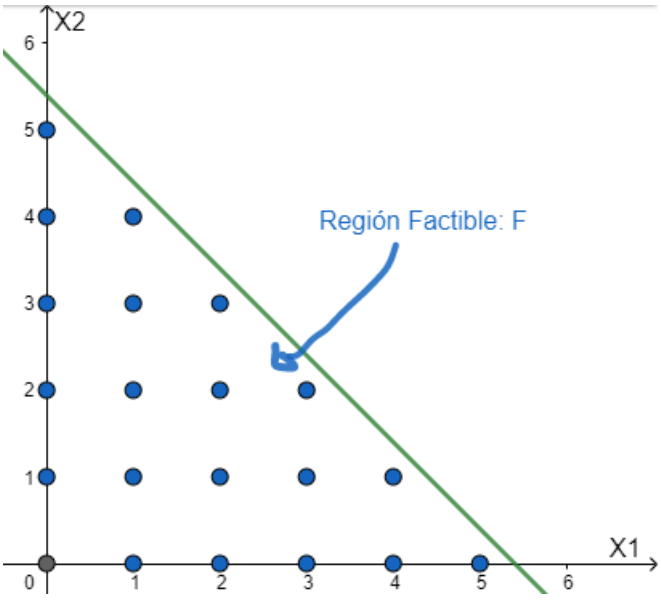


Figura 2.3: Ejemplo de programación lineal entera

Como el problema es un IP, entonces la solución óptima de dicho pro-

blema será alguno de los puntos que se pintan en la Figura (2.3). Para encontrar la solución óptima calculamos el valor de la función objetivo en cada uno de los puntos y así tenemos que:

$$\text{Solución Óptima: } x_1^* = 0 \quad \& \quad x_2^* = 5 \tag{2.5}$$

$$\text{Valor de la función objetivo: } \quad 10$$

Si resolvemos el mismo problema como uno de programación lineal, es decir, que las variables $x_1, x_2 \geq 0$, entonces tenemos:

$$\text{Solución Óptima: } \quad x_1^* = 0 \quad \& \quad x_2^* = 5{,}4 \tag{2.6}$$

$$\text{Valor de la función objetivo: } \quad 10{,}85$$

Así, podemos observar que la solución de un problema de programación lineal no coincide con la solución de un problema de programación entera y tenemos que:

$$z_{IP}^* \leq z_{LP}^*$$

La relajación lineal del problema se obtiene al omitir todas las condiciones sobre las variables.

Problema entero: IP	**Problema relajado: PR**
$max \quad z = c^T x$	$max \quad z = c^T x$
sujeto a:	sujeto a:
$Ax \leq b$	$Ax \leq b$
$x \geq 0$	$x \geq 0$
$x \in \mathbb{Z}^n$	

Hay que mencionar que al relajar un problema de programación lineal entera no se pierden soluciones factibles y trae consigo la ventaja de que el modelo se vuelve tratable.

Dado que la programación lineal entera es un problema NP-completo, se han desarrollado diferentes métodos para facilitar su solución. Entre los métodos más conocidos para resolver problema IP podemos mencionar: algoritmos de planos cortantes, algoritmos de tipo Branch & Bound y algoritmos de tipo Branch & Cut.

Capítulo 3

Planificación de líneas y enrutamiento de pasajeros

3.1. Notaciones

Consideraremos que una red de transporte de autobuses es un dígrafo $D = (V, A)$ con una función de costos sobre los arcos $c : A \to \mathbf{R}^+$, donde V representa las estaciones de buses y A corresponde a los enlaces directos entre dos estaciones diferentes. Además, c_a es el costo de viaje entre las estaciones asociadas al arco $a \in A$. Nos restringiremos a trabajar con grafos lineales debido a la estructura que presenta el sistema de transporte de Quito.

Definición 3. *Diremos que un nodo es una terminal si pueden iniciar y finalizar su ruta en ese nodo o si se les permite que buses puedan estacionarse en aquel lugar.*

Definición 4. *Una línea es un circuito que contiene por lo menos una terminal en uno de sus extremos.*

La flota de buses es homogénea y solo tendremos un modo de transporte. La capacidad del modo de transporte será denotado por $\kappa \in \mathbb{Z}^+$. El conjunto de líneas de la red de transporte es seleccionadas a priori y se denotará por L. Además, definimos por L_a al conjunto de líneas que usan el arco a y $A[l] = \{a_1(l), a_2(l), ..., a_n(l)\}$ será el conjunto de arcos que

son cubiertos por la línea $l \in L$. Cada línea $l \in L$ tiene asociado un costo fijo denotado por $K_l \in \mathbb{R}^+$ y un costo variable asociado a un solo viaje definido como $C_l = \sum_{a \in A[l]} c_a$. Finalmente, la frecuencia de cada línea $l \in L$ estará acotada por $f^{max} \in \mathbb{Z}^+$.

Para cada par de nodos $s, t \in V$, $d_{st} \in \mathbb{Z}^+$ representa el número de pasajeros que desean trasladarse de la estación s a la estación t. A lo largo de este trabajo asumiremos que $d_{st} \neq d_{ts}$ para todo $s, t \in V$. Si $d_{st} > 0$ entonces el par de nodos (s, t) es llamado OD-par y el conjunto de los OD pares esta definido por el siguiente conjunto

$$\tau = \{(s, t) \in V \times V : d_{st} > 0\}$$

Un camino simple $p = \{(v_1, v_2), (v_2, v_3), ..., (v_{r-2}, v_{r-1}), (v_{r-1}, v_r)\}$ es un st-camino si $s = v_1$, $t = v_r$ y $(v_1, v_2), ..., (v_{r-1}, v_r) \in \bigcup_{l \in L} A[l]$.

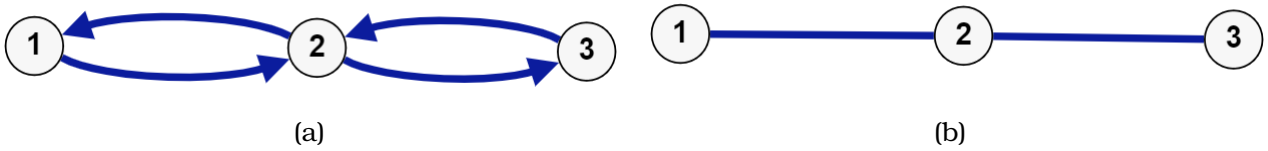
Definición 5. *Un viaje directo usando alguna línea $l \in L$ es un camino p si y solo si los arcos $(v_1, v_2), ..., (v_{r-1}, v_r) \in A[l]$, es decir, los pasajeros pueden ir de su origen s hasta su destino t exclusivamente usando la línea l.*

Definición 6. *Un viaje no directo es un camino con al menos una transferencia, es decir, p es un viaje no directo con un cambio de línea si $(v_1, v_2), ..., (v_{k-1}, v_k) \in A[l_1]$ y $(v_k, v_{k+1}), ..., (v_{r-1}, v_r) \in A[l_2]$ y $l_1 \neq l_2$.*

Al conjunto de viajes directos lo denotaremos por P^0 mientras que el conjunto de viajes no directos será denotado por P^1, además $P = P^0 \cup P^1$ representa el conjunto de todos los viajes. Para cada camino $p \in P$ se impone un costo c_p que penaliza a los viajes no directos. Así, si $p \in P^0$ entonces $c_p = 0$ mientras que si $p \in P^1$ consideraremos que $c_p > 0$. Si se desea contabilizar el número de transferencias en el sistema se fija $c_p = 1$. Finalmente, sea $P(s, t) \subset P$ es el conjunto de todos los st-caminos que van de s hacia t en el grafo dirigido D.

Por la forma simétrica de las líneas (ver Figura 3.1), es posible transformar el grafo dirigido en una versión no dirigida tomando:

$$d(s, t) = max\{d_{st}, d_{ts}\}$$



(a) (b)

Figura 3.1: Transformación de la versión dirigida en la versión no dirigida

Así una línea en un grafo no dirigido $G = (V, E)$, es un camino con al menos una terminal y tiene los mismos costos que para el caso dirigido. Un st-camino es un camino en G, P es el conjunto de viajes directos y no directos en el grafo y $\tau = \{(s,t) \in V \times V : max\{d_{st}, d_{t,s}\} > 0, s < t\}$ representa el conjunto de pares OD. Finalmente, $E[l] = \{e_1(l), e_2(l), ..., e_n(l)\}$ es el conjunto de aristas cubiertas por la línea $l \in L$.

Un problema de planificación de líneas consiste en encontrar un conjunto de líneas $\mathbb{L} \subseteq L$, con sus respectivas frecuencias de tal forma que la demanda sea satisfecha, es decir, los pasajeros sean enrutados sobre las líneas $\mathbb{L}$ seleccionadas en la solución. A continuación presentamos dos modelos que nos ayudarán a resolver este problema.

3.2. Modelo de viajes directos

El LPP de esta sección se encarga de seleccionar un conjunto de líneas $\mathbb{L} \subseteq L$ con sus respectivas frecuencias de tal forma que los pasajeros puedan ir de su origen hasta su destino solamente usando viajes directos, es decir, cualquier pasajero podría alcanzar su destino sin realizar trasbordos durante sus viajes. Esto implica que cualquier camino p se encontrará únicamente en el conjunto P^0 y p será servido por la línea $l \in L$ si y solo si p está contenido en $E[l]$.

Definimos $L(s,t) \subseteq L$ como el conjunto de líneas para las cuales un viaje directo de s a t es posible. Además, sea $\tau(l)$ el conjunto de los pares OD que pueden ser servidos por la línea l a través de un viaje directo y $\tau_{e(l)}$ es el conjunto de todos los pares OD que pueden ser servidos por la línea l usando la arista $e(l) \in E[l]$.

Las variables que usamos en este modelo son las siguientes: $x^l_{st} \in \mathbb{Z}^+$ representa el número de viajeros que van de s a t mediante un viaje

directo usando la línea l, $y_l \in \{0,1\}$ es una variable binaria que tomará el valor de 1 si la línea l es usada en la solución y 0 caso contrario. Finalmente, f_l denota la frecuencia de la línea l para todo $l \in L$.

El modelo de planificación de líneas usando exclusivamente viajes directos (DCTM) puede ser escrito de la siguiente forma:

$$\min \quad \sum_{l \in L} K_l y_l + \sum_{l \in L} C_l f_l \tag{3.1}$$

s.a.r:

$$\sum_{l \in L(s,t)} x_{st}^l = d_{st}, \qquad \forall (s,t) \in \tau, \tag{3.2}$$

$$\sum_{(s,t) \in \tau_{e(l)}} x_{st}^l \leq \kappa f_l, \qquad \forall l \in L, e(l) \in E[l], \tag{3.3}$$

$$f_l \leq f^{max} y_l, \qquad \forall l \in L, \tag{3.4}$$

$$x_{st} \in \mathbb{Z}^+, \qquad \forall (s,t) \in \tau, \tag{3.5}$$

$$f_l \in \mathbb{Z}^+, \qquad \forall l \in L, \tag{3.6}$$

$$y_l \in \{0,1\}, \qquad \forall l \in L. \tag{3.7}$$

La función objetivo (3.1) minimiza los costos totales de operación, es decir, la suma de los costos fijos y costos variables. La restricción (3.2) asegura que la demanda de cada uno de los pares OD $(s,t) \in \tau$ es satisfecha netamente por viajes directos. La restricción (3.3) nos dice que la frecuencia de la línea debe ser lo suficientemente grande para servir los viajes directos asignados a esa línea. Finalmente, la restricción (3.4) define una cota superior para cada una de las frecuencias de cada línea.

3.3. Modelo de maximización de viajes directos

El LPP tratado en esta sección consiste en encontrar un conjunto de líneas $\mathbb{L} \subseteq L$ con sus respectivas frecuencias de tal forma que los pasajeros sean enrutados a través de viajes directos y viajes con una transferencia. El objetivo se enfoca en minimizar los costos totales y maximizar los viajes directos realizados en el sistema.

Las variables que se emplean en este modelo son las siguientes: $x_{st}^p \in \mathbb{Z}^+$ representa el número de viajeros que van de la estación s a la estación t usando el camino $p \in P$, $y_l \in \{0,1\}$ corresponde a una variable binaria que toma el valor de 1 si usamos la línea l en la solución y 0 caso contrario. Finalmente, f_l denota la frecuencia de la línea l para todo $l \in L$.

El modelo integrado de planificación de líneas y enrutamiento de pasajeros (MDCTM) puede ser expresado de la siguiente forma:

$$min \quad \lambda \left(\sum_{l \in L} K_l y_l + \sum_{l \in L} C_l f_l \right) + (1-\lambda) \left(\sum_{(s,t) \in \tau} \sum_{p \in P(s,t)} c_p x_{st}^p \right) \tag{3.8}$$

s.r.a:

$$\sum_{p \in P(s,t)} x_{st}^p = d_{st}, \qquad \forall (s,t) \in \tau, \tag{3.9}$$

$$\sum_{(s,t) \in \tau} \sum_{p(s,t): e(l) \in p} x_{st}^p \leq \kappa f_l, \qquad \forall l \in L, e(l) \in E[l], \tag{3.10}$$

$$f_l \leq f^{max} y_l, \qquad \forall l \in L, \tag{3.11}$$

$$x_{st}^p \in \mathbb{Z}^+, \qquad \forall (s,t) \in \tau, p \in P, \tag{3.12}$$

$$f_l \in \mathbb{Z}^+, \qquad \forall l \in L, \tag{3.13}$$

$$y_l \in \{0,1\}, \qquad \forall l \in L. \tag{3.14}$$

La función objetivo (3.8) pretende minimizar la suma ponderada de los costos de operación y los costos de enrutamiento de pasajeros, razón por la cual esta consiste de 2 partes: la primera corresponde a la suma de los costos fijos y costos variables de las líneas y frecuencias y la segunda parte corresponde al costo incurrido por las transferencias. La prioridad es controlada por un parámetro de peso $\lambda \in [0, 1]$.

La restricción (3.9) garantiza que todos los pares OD $(s, t) \in \tau$ sean enrutados, es decir, que la demanda sea satisfecha. La restricción (3.10) determina la capacidad de la línea para transportar a los pasajeros que usen esa línea. Finalmente, la restricción (3.11) se encarga de acotar superiormente a las frecuencias de cada una de las líneas.

3.4. Aplicación de los modelos

A continuación se presenta un ejemplo de aplicación de los modelos a una red de transporte con 2 terminales (nodos de amarillo) y 3 estaciones.

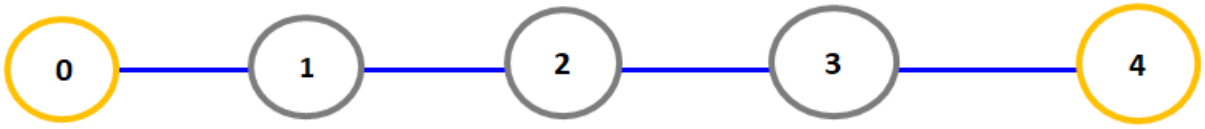


Figura 3.2: Red de transporte con 2 terminales y 3 estaciones

En la Tabla (3.1) se encuentra la información necesaria para construir cada modelo. En la primera columna están los nodos de la red de transporte que son denotados como Nd_i para $i = 0, 1, 2, 3, 4$, la segunda columna llamada Term/Est/Giro nos indica si el nodo es una terminal, una estación o una estación en la cual el trolebús puede realizar un giro. Así tenemos que:

$$Nd_i = \begin{cases} 0 & \text{si el nodo i es una estación en la que no puede girar} \\ 1 & \text{si el nodo i es una estación de giro} \\ 2 & \text{si el nodo i es una terminal} \end{cases}$$

La tercera columna (Dist) nos proporciona la distancia en kilómetros

que hay desde el nodo 0 hasta el i-ésimo nodo. Finalmente, las columnas Nd_i para $i = 0, 1, 2, 3, 4$ almacenan la información de la matriz OD.

	Term/Est/Giro	**Dist**	Nd_0	Nd_1	Nd_2	Nd_3	Nd_4
Nd_0	2	0 Km	0	80	111	94	228
Nd_1	0	2Km	164	0	168	198	187
Nd_2	1	5Km	100	150	0	225	181
Nd_3	1	8 Km	90	160	200	0	123
Nd_4	2	10Km	241	165	180	103	0

Cuadro 3.1: Matriz de demanda e información

Nuestro grafo $G = (V, E)$ estará dado por la forma de la red de transporte que se muestra en la Figura (3.2), donde: $V = \{0, 1, 2, 3, 4\}$ y $E = \{\{0, 1\}, \{1, 2\}, \{2, 3\}, \{3, 4\}\}$

Recordemos que una línea es un circuito que contiene por lo menos una terminal en uno de sus extremos. Por tanto, cada una de las líneas irá desde una terminal hasta una estación de giro u otra terminal (ver Figura 3.3). Así el conjunto $L = \{l_1, l_2, l_3, l_4, l_5\}$, donde $l_1 = \{0, 1, 2\}$, $l_2 = \{0, 1, 2, 3\}$, $l_3 = \{0, 1, 2, 3, 4\}$, $l_4 = \{2, 3, 4\}$, $l_5 = \{3, 4\}$.

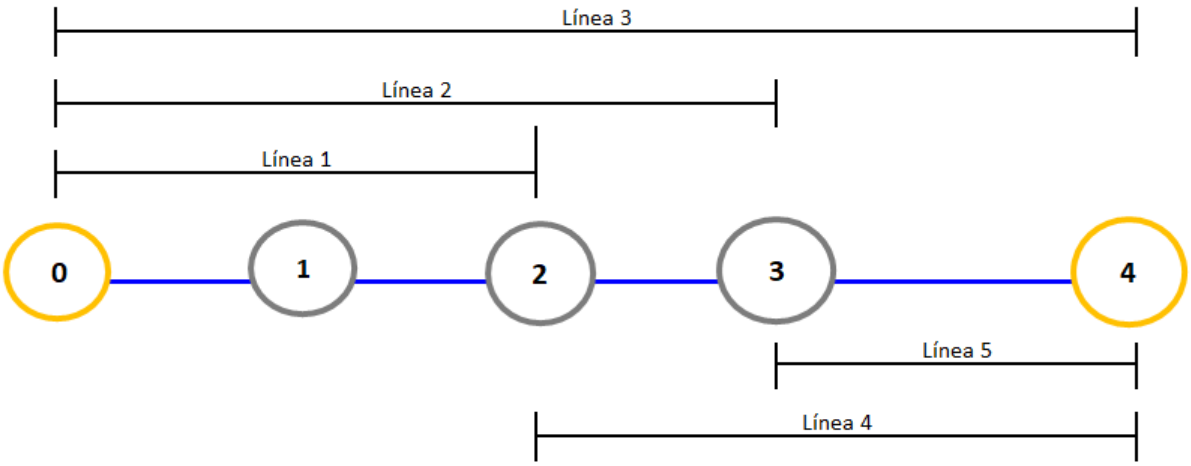


Figura 3.3: Líneas de la red

Para cada una de las líneas su costo fijo K_l será de $425 (corresponde al sueldo que se le pagará al conductor) y su respectiva frecuencia estará acotada por $f^{max} = 4$. El costo variable C_l será calculado como se muestra en la ecuación (3.15).

$$C_l = cost_{km} * d_{(v_1(l),v_n(l))} \tag{3.15}$$

donde $cost_{km}$ corresponde al costo por cada kilómetro recorrido y $d_{(v_1(l),v_n(l))}$ es la distancia que existe entre el nodo inicial $v_1(l)$ de la línea l y su nodo final $v_n(l)$. Para calcular la longitud de la línea restaremos la longitud del nodo final de la línea l menos la longitud del nodo inicia de la misma. En la Tabla (3.2) se mostrará la longitud en kilómetros de cada línea y su respectivo costo variable asociado a dicha distancia, teniendo en cuenta que $cost_{km} = \$10$.

Línea	**Longitud en Km**	**Costo Variable** C_l
l_1	5	\$50
l_2	8	\$80
l_3	10	\$100
l_4	5	\$ 50
l_5	2	\$ 20

Cuadro 3.2: Costo variable y longitud de las líneas

La capacidad del modo de transporte es $\kappa = 180$. Al resolver la instancia usando el modelo de viajes directos obtuvimos el siguiente resultado:

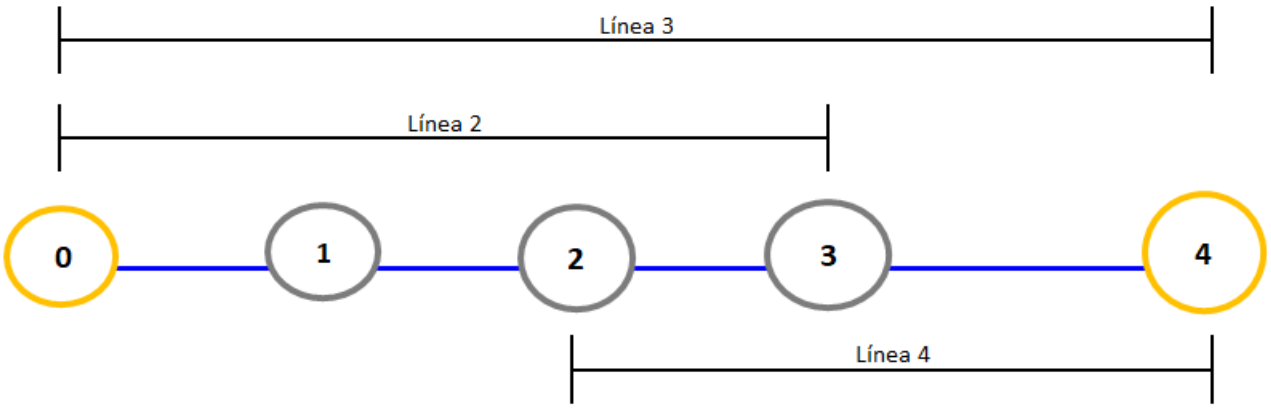


Figura 3.4: Solución del modelo de viajes directos

El valor óptimo de la función objetivo es \$1885. En la primera columna de la Tabla (3.3) se muestran las líneas que fueron seleccionadas por el modelo, mientras que en la segunda columna tenemos las frecuencias para cada una de las líneas seleccionadas.

Líneas	Frecuencias
l_2	2
l_3	4
l_4	1

Cuadro 3.3: Líneas y frecuencias de DCTM

Para el modelo de maximización de viajes directos necesitamos construir los conjuntos P^0 (conjunto de viajes directos) y P^1 (conjunto de viajes no directos). El conjunto P^0 estará conformado por viajes que usan una sola línea, mientras que P^1 estará formado por viajes que pueden ser realizados mediante la unión de 2 o más líneas (siempre que la intersección de nodos de las líneas no sea vacía). En este trabajo solamente hemos considerado la unión de 2 líneas, es decir, viajes con una sola transferencia y además la transferencia se realiza al final de la primera línea tomada en el viaje.

En la Figura (3.5) se puede observar la transferencia (línea en rojo) que se realiza desde la línea 1 hacia la línea 2, $l_{i(j)}$ representa la i-ésima línea ($i = 1, 2, 3, 4, 5$) que usa el nodo j ($j = 0, 1, 2, 3, 4$) de dicha línea.

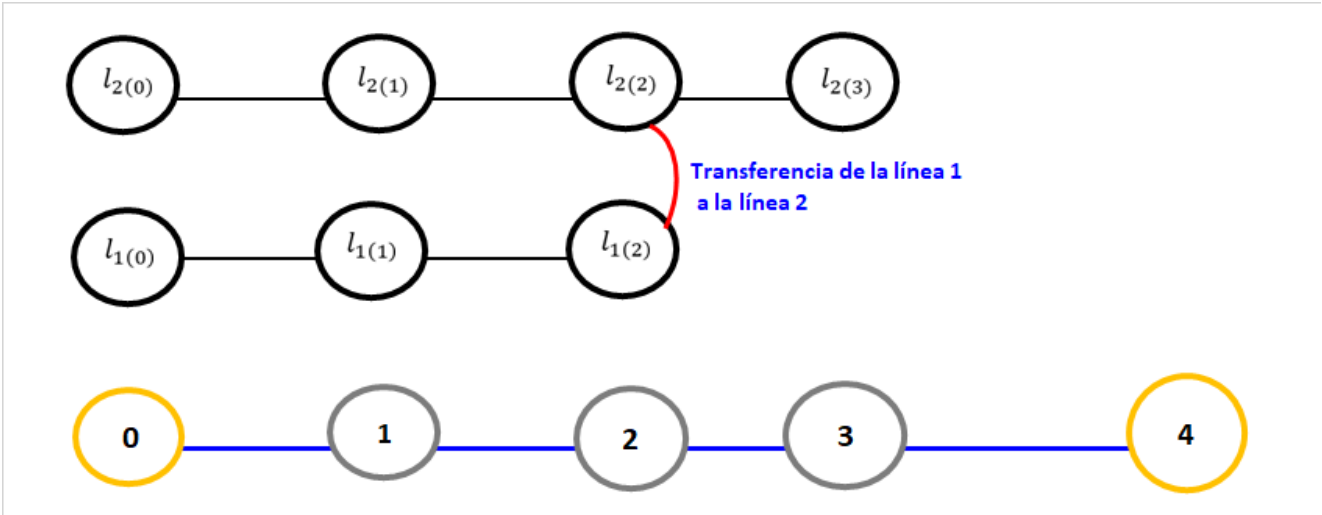


Figura 3.5: Transferencia de la línea 1 a la línea 2

Por la forma de nuestra red (ver Figura 3.6) tenemos que: $P^0 = \{p_1, p_2, p_3, p_4, p_5\}$ y $P^1 = \{p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}\}$ donde $p_1 = \{e_1(l_1), e_2(l_1)\}$, $p_2 = \{e_1(l_2), e_2(l_2), e_3(l_2)\}$, $p_3 = \{e_1(l_3), e_2(l_3), e_3(l_3), e_4(l_3)\}$, $p_4 = \{e_1(l_4), e_2(l_4)\}$, $p_5 = \{e_1(l_5)\}$,$p_6 = \{e_1(l_1), e_2(l_1), e_3(l_2)\}$, $p_7 = \{e_1(l_1), e_2(l_1), e_3(l_4), e_4(l_4)\}$, $p_8 = \{e_1(l_1), e_2(l_1), e_3(l_3), e_4(l_3)\}$, $p_9 = \{e_1(l_2), e_2(l_2), e_1(l_4),$

$e_2(l_4)\}$, $p_{10} = \{e_1(l_2), e_2(l_2), e_1(l_5)\}$, $p_{11} = \{e_1(l_2), e_2(l_2), e_3(l_2), e_4(l_3)\}$, $p_{12} = \{e_1(l_4), e_1(l_5)\}$, $p_{13} = \{e_1(l_3), e_2(l_3), e_1(l_4), e_2(l_4)\}$ y $p_{14} = \{e_1(l_3), e_2(l_3), e_3(l_3), e_1(l_5)\}$

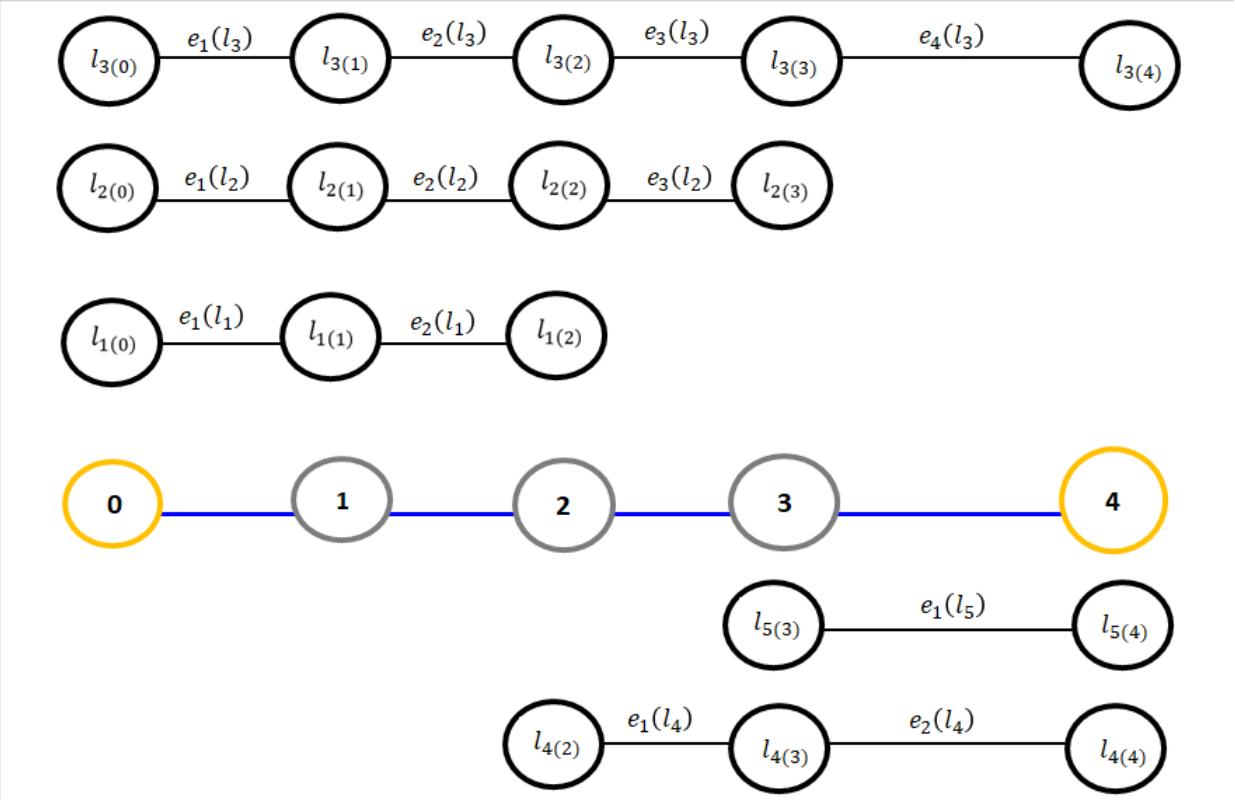


Figura 3.6: Arcos Lineas

El costo para los caminos está dado de la siguiente manera:

$$c_p = \begin{cases} 0 & \text{si } p \in P^0 \\ 1 & \text{si } p \in P^1 \end{cases}$$

Al resolver la instancia usando el modelo de maximización de viajes directos usando un parámetro de peso $\lambda = 0{,}5$ las líneas que usaremos con sus respectivas frecuencias son las siguientes:

Líneas	**Frecuencias**
l_2	2
l_3	4
l_4	1

Cuadro 3.4: Líneas y frecuencias de MDCT

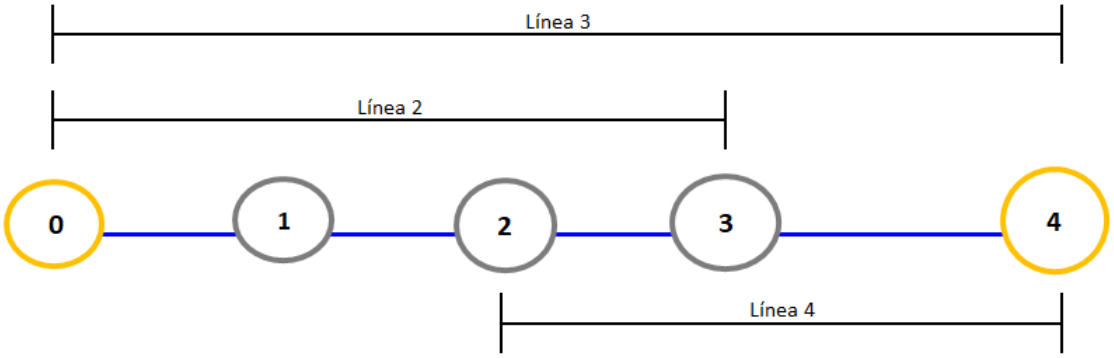


Figura 3.7: Solución modelo de maximización de viajes directos

El valor de la función objetivo es de $ 942,5. Podemos observar que el valor de la función objetivo de los modelos es diferente y esto se debe a que el modelo MDCTM contiene un parámetro de peso.

Capítulo 4

Resultados Computacionales

En este capítulo se presentan los resultados computacionales obtenidos al implementar el modelo simplificado y el modelo integrado de planificación de líneas y frecuencias y enrutamiento de pasajeros.

Todos los experimentos computacionales fueron realizados utilizando un computador con procesador Intel Core i5-8265U CPU 1.80GHz con 4 GB de memoria RAM bajo el sistema operativo Windows 10. Los modelos fueron implementados usando la aplicación Jupyter Notebook mediante el lenguaje de programación Python y con la versión 9.1.2 del solver de optimización Gurobi a través de la distribución de Anaconda.

4.1. Resultados

El costo fijo de cada línea será de $ 425 y su respectiva frecuencia estará acotada por $f^{max} = 20$. Además, la capacidad de cada una de las unidades será $\kappa = 180$ y el costo por kilómetro recorrido (costo variable) será de $10.

En las Tablas (4.1) y (4.2) se muestran los resultados obtenidos para instancias de diferentes tamaños usando el modelo DCTM. En dicha tabla se muestra la cantidad de nodos de la instancia, es decir, la cantidad de estaciones y terminales, el número total de pasajeros de la matriz OD, el número de líneas generadas, el valor de la función objetivo, la brecha de

optimalidad (gap) medido en porcentaje, el tiempo de ejecución medido en segundos y la cantidad de nodos explorados. En este caso, el tiempo máximo de ejecución fue fijado en 1800 segundos (30 minutos).

DCTM						
n	# pasajeros	# de líneas	F.Obj	Gap (%)	Tiempo	Nodos
30	31 819	51	14 869,60	0,00	64,78	1 126
30	32 083	51	15 457,80	0,00	54,26	290
30	32 084	51	15 705,40	0,19	105,71	2 463
30	32 135	51	16 490,70	0,00	342,30	3 354
30	31 961	51	16 201,40	0,00	112,23	1 307
40	43 035	69	25 579,00	0,00	1 412,73	5 472
40	43 008	69	27 304,50	1,12	1800,05	1 783
40	42 975	69	25 962,50	1,60	1800,04	2718
40	42 921	69	25 271,20	0,00	1 004,84	1 699
40	42 944	69	25 983,80	1,88	1800,04	2 069
50	54 206	114	38 253,50	3,38	1 800,03	3
50	54 180	114	40 310,80	3,70	1 800,08	3
50	54 067	114	39 636,30	5,92	1 800,03	23
50	54 354	114	38 902,60	4,22	1 800,08	18
50	53 969	114	38 477,60	4,04	1 800,25	2
60	65 350	170	58 178,20	19,70	1800,08	1
60	65 248	170	58 412,80	17,30	1800,04	1
60	65 281	170	64 743,20	19,67	1 800,03	1
60	65 258	170	60 689,00	19,12	1 800,06	1
60	65 105	170	65 200,20	25,62	1 800,06	1
70	76 295	200	83 592,20	21,81	1 800,37	1
70	76 493	200	79 434,80	17,25	1 800,84	1
70	76 338	200	80 279,60	16,72	1 965,11	1
70	76 488	200	87 665,50	24,64	1 800,10	1
70	76 156	200	91 701,30	26,73	1 800,04	1
80	87 759	273	100 475,50	17,56	1 801,17	1
80	87 847	273	100 185,60	13,22	1 802,11	1
80	87 742	273	111 853,40	22,22	1 801,20	1
80	87 784	273	113 557,10	24,10	1 800,83	1

Cuadro 4.1: Resultados modelo DCTM

DCTM						
n	# pasajeros	# de líneas	F.Obj	Gap (%)	Tiempo	Nodos
80	87 551	273	179 285,20	52,43	1 802,15	0
90	99 011	357	215 109,30	51,27	1 802,54	0
90	99 003	357	218 307,40	52,94	1 800,67	0
90	99 231	357	220 342,20	51,91	1 802,77	0
90	99 331	357	219 298,80	53,09	1 801,18	0
90	99 063	357	222 629,90	51,34	1 803,71	0
100	110 696	399	-	-	-	-
100	110 447	399	257 651,70	98,53	1 811,90	0
100	110 458	399	250 239,30	51,04	1 800,82	0
100	110 591	460	258 476,40	98,80	1 813,22	0
100	110 798	460	278 882,00	51,18	1 803,60	0
100	110 807	399	273 380,00	50,49	1 800,68	0
100	110 656	460	264 716,40	50,70	1 803,35	0
100	110 430	399	265 708,30	50,48	1 802,49	0
100	110 502	399	195 209,60	30,73	1 801,29	1

Cuadro 4.2: Resultado modelos DCTM

Podemos observar que para instancias pequeñas de hasta 50 nodos, el modelo funciona bien ya que se obtienen soluciones con un gap menor al 5 %, a pesar de que el tiempo máximo empleado fue de 1 800 segundos. Para instancias entre 60 y 80 nodos el gap se triplica en relación a las instancias menores a 50 y todas las instancias utilizan el tiempo máximo permitido. Por el contrario, las instancias mayores a 80 nodos emplean los 1 800 segundos y además se obtiene un gap mayor al 50 % e incluso en algunos casos no encuentra una solución factible.

El siguiente experimento consiste comparar el modelo MDCTM con una versión relajada del mismo. Definimos MDCTM-R al modelo MDCTM en la que únicamente las variables x^p_{st} toman valores reales, es decir, se impone $x^p_{st} \geq 0$, $f_l \in \mathbb{Z}$, $y_l \in \{0, 1\}$. Para ejecutar el modelo MDCTM y MDCTM-R se tomó como parámetro de peso $\lambda = 0,20$. En la Tabla (4.3) se presenta los resultados obtenidos para el modelo MDCTM y en la Tabla (4.4) se encuentran los resultados para la versión relajada del modelo usando instancias de tamaño $n = 30$ y $n = 40$. Las tablas contienen la

cantidad de nodos de la red de transporte, el número de pasajeros, la cantidad de camino de cada instancia, el valor de la función objetivo, la brecha de optimalidad, tiempo de ejecución y el número de nodos explorados.

MDCTM						
n	# pasajeros	# de caminos	F.Obj	Gap (%)	Tiempo	Nodos
30	31 819	1 006	2 973,92	0,00	576,70	1010
30	32 083	1 031	3 091,56	0,00	717,51	739
30	32 084	1007	3 141,08	0,00	1 149,61	3 425
30	32 135	1 022	3 308,00	2,76	1 800,96	2 812
30	31 961	1010	3 240,28	0,00	1 342,82	1 923
40	43 035	1 814	5 166,42	4,38	1 802,01	1
40	43 008	1 835	5 553,34	4,91	1 801,37	1
40	42 975	1 819	5 250,68	4,91	1 800,56	1
40	42 921	1 835	5 092,96	3,61	1 815,69	1
40	42 944	1 826	5 219,22	3,94	1 801,99	1

Cuadro 4.3: Resultado modelo MDCTM

MDCTM-R						
n	# pasajeros	# de caminos	F.Obj	Gap (%)	Tiempo	Nodos
30	31 819	1 006	2973,92	0,00	496,17	493
30	32 083	1 031	3 091,56	0,00	536,57	368
30	32 084	1 007	3 141,08	0,00	766,83	2 032
30	32 135	1 022	3 305,92	3,35	1 801,24	1 046
30	31 961	1 010	3 240,28	0,00	1 052,89	1 814
40	43 035	1 814	5 115,80	3,38	1 809,89	1
40	43 008	1 835	5 577,02	5,21	1 840,46	1
40	42 975	1 819	5 229,90	4,30	1 804,13	1
40	42 921	1 835	5 059,16	2,98	1 804,32	1
40	42 944	1 826	5 273,12	4,91	1 804,14	1

Cuadro 4.4: Resultado modelo MDCTM-R

Podemos observar que para instancias de 30 nodos el valor de la función objetivo del modelo relajado coincide en su gran mayoría de experimentos con el valor obtenido del modelo MDCTM. Sin embargo, podemos

apreciar que el tiempo de ejecución del modelo relajado en todos los experimentos realizados fue menor. En el caso de las instancias de tamaño 40 podemos apreciar que el comportamiento de los dos modelos es similar ya que al ser el modelo de gran tamaño, el método de solución solo genera un nodo en el árbol de ramificación

En la Figura (4.1) se puede visualizar el tiempo de ejecución empleado para las 5 instancias de tamaño 30. Podemos notar que el modelo DCTM tiene el menor tiempo de ejecución mientras que el modelo MDCTM es el que emplea la mayor cantidad de tiempo.

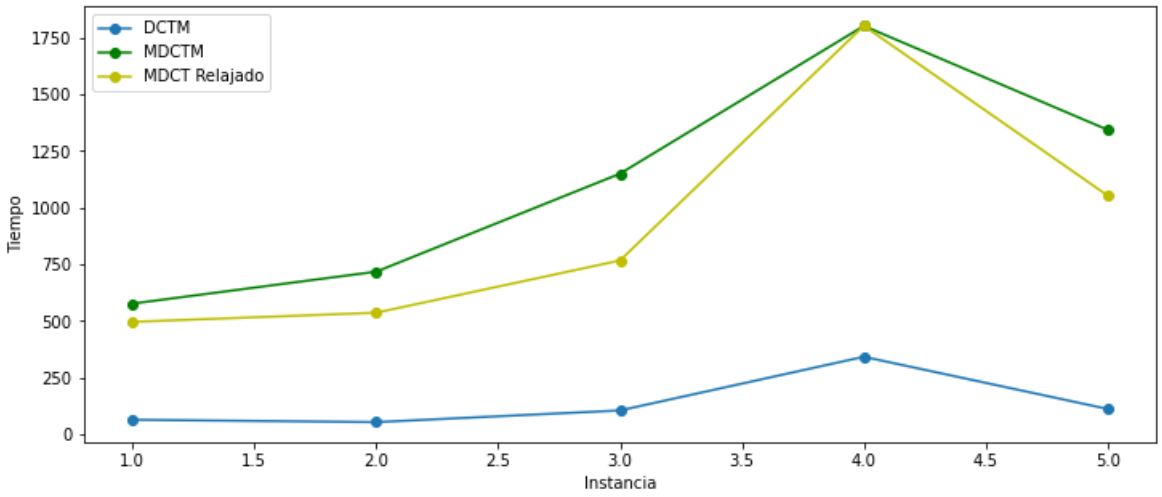


Figura 4.1: Instancia vs Tiempo

El siguiente experimento considera el modelo MDCTM usando la primera instancia de 30 nodos con la misma información del experimento anterior. Se propone modificar los valores del parámetro de peso $\lambda \in \{0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50, 0.55,0.60,0.65,0.70, 0.75,0.80,0.85,0.90,0.95,1\}$. Para este caso la demanda es de 31 819 pasajeros, la cantidad de líneas corresponde a 51 y fueron generados 1 006 caminos. Además, se generaron 259 473 variables y 1 137 restricciones.

λ	F.Objetivo	GAP (%)	Tiempo	Nodos
0,05	724,73	0,00	745,61	1 139
0,10	1 449,46	0,00	892,26	702
0,15	2 174,19	0,00	840,17	727
0,20	2 898,10	0,00	845,76	533

Cuadro 4.5: Instancia de 30 nodos con diferentes valores de λ

λ	F.Objetivo	GAP (%)	Tiempo	Nodos
0,25	3 623,65	0,00	899,96	774
0,30	4 348,38	0,00	705,10	774
0,35	5 204,36	0,00	628,95	527
0,40	5 797,84	0,00	1 660,51	645
0,45	6 691,32	0,00	693,88	692
0,50	7 247,30	0,00	1 392,88	1 049
0,55	8 178,28	0,00	772,52	624
0,60	8 696,76	0,00	1 023,51	879
0,65	9 665,24	0,00	695,82	783
0,70	10 146,22	0,00	1 086,69	721
0,75	11 152,2	0,00	1 240,31	834
0,80	11 595,68	0,00	980,68	793
0,85	12 639,16	0,00	1 385,93	898
0,90	13 045,14	0,00	1 803,72	569
0,95	14 126,12	1,87	1 801,24	733
1,00	14 896,60	2,52	1 800,71	520

Cuadro 4.6: Instancia de 30 nodos con diferentes valores de λ

En la Figura (4.2) se muestra como varía la función objetivo cuando cambiamos los valores del parámetro de peso λ. Podemos observar que si el valor de λ aumenta también aumenta el valor de la función objetivo.

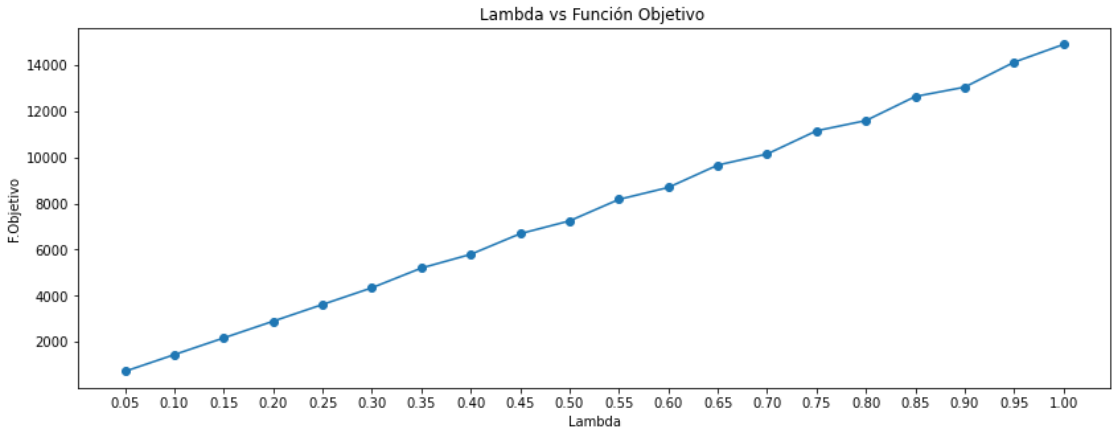


Figura 4.2: λ vs F.Objetivo

El experimento que se propone a continuación consiste en modificar la capacidad del modo de transporte y observar qué efecto tiene dicha modificación en los valores obtenidos por la función objetivo. Para realizar dicho experimento hemos tomado la primera instancia de 30 nodos con las siguientes características: el número total de pasajeros de la matriz OD es 31 819, el costo fijo de las líneas es $425, el costo variable por kilómetro recorrido corresponde a $10, las frecuencias están acotadas por $f^{max} = 20$, la cantidad de líneas es 51 y para el caso de las transferencias se generaron 1 006 caminos. Además, el parámetro de peso λ se fijó en 0.20. En las Tablas (4.7), (4.8) y (4.9) se presentan los resultados obtenidos para cada modelo.

DCTM				
κ	F.Objetivo	GAP(%)	Tiempo	Nodos
100	26 110,30	0,00	144,51	9 501
120	21 927,60	0,00	138,72	2 219
140	18 947,60	0,00	194,98	2 975
160	16 771,30	0,00	106,14	3 379
180	14 869,60	0,00	79,35	1 126
200	13 556,00	0,00	62,93	1 491
220	12 510,60	0,00	66,52	2 000
240	11 435,80	0,00	44,20	821
260	10 656,20	0,00	35,16	637
280	9 996,80	0,00	44,06	997
300	9 343,50	0,00	25,14	553

Cuadro 4.7: Instancia de 30 nodos con diferentes valores de κ (DCTM)

MDCTM				
κ	F.Objetivo	GAP(%)	Tiempo	Nodos
100	5 222,06	1,39	1 802,02	991
120	4 398,92	0,71	1 801,80	3 020
140	3 789,52	0,00	1 124,40	2 277
160	3 354,26	0,00	1 717,45	3 478
180	2 973,92	0,00	623,55	1 010
200	2 711,20	0,00	892,62	1 488
220	2 502,12	0,00	940,70	1 871
240	2 287,16	0,00	671,49	728
260	2 131,24	0,00	923,82	675
280	1 999,36	0,00	790,25	953
300	1 868,70	0,00	803,51	487

Cuadro 4.8: Instancia de 30 nodos con diferentes valores de κ (MDCTM)

MDCTM-R				
κ	F.Objetivo	GAP(%)	Tiempo	Nodos
100	5 222,06	1,36	1 816,45	1 059
120	4 385,52	0,00	1 695,98	3 278
140	3 789,52	0,00	892,20	1 715
160	3 354,26	0,41	1 875,47	2 145
180	2 973,92	0,00	645,39	493
200	2 711,20	0,00	811,74	1 138
220	2 502,12	0,00	1 003,03	1 721
240	2 287,16	0,00	635,10	367
260	2 131,24	0,00	496,59	402
280	1 999,36	0,00	593,02	485
300	1 868,70	0,00	468,74	416

Cuadro 4.9: Instancia de 30 nodos con diferentes de κ (MDCTM-R)

En las Tablas (4.8) y (4.9) podemos notar que cuando la capacidad del modo de transporte disminuye el gap empieza a aumentar.

En la Figura (4.3) se puede observar cómo se ve afectado el valor de la función objetivo cuando incrementamos la capacidad del modo de trans-

porte. Podemos apreciar que si la capacidad del modo de transporte aumenta los costos se reducen.

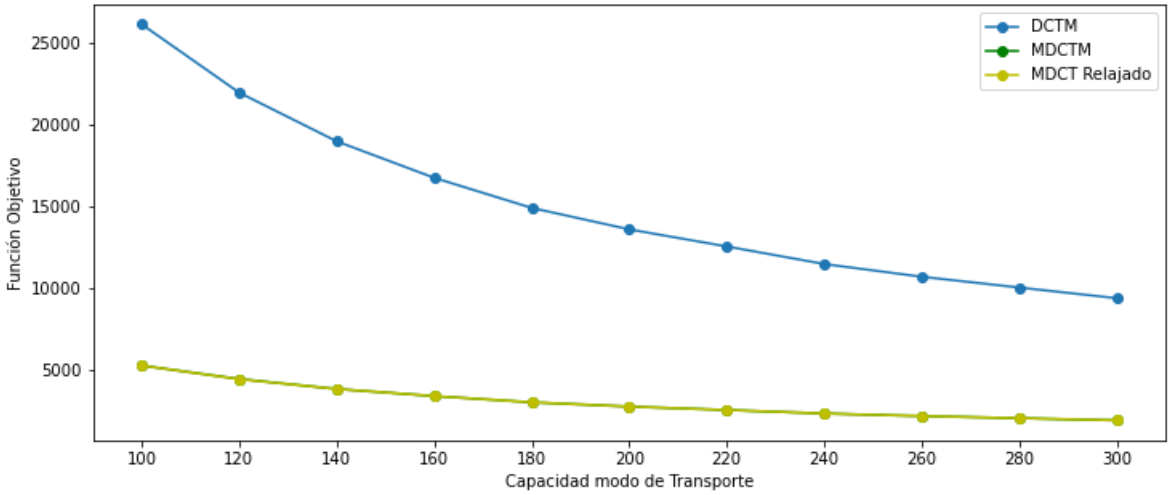


Figura 4.3: Capacidad del modo de Transporte vs Función Objetivo

Capítulo 5

Conclusiones y recomendaciones

En el presente trabajo se estudia el problema de planificación de líneas y frecuencias y enrutamiento de pasajeros, para ello se estudiaron 2 modelos que nos ayudaron a resolver este problema. El primer modelo (DCTM) enruta a los pasajeros desde su origen hasta su destino solamente a través de viajes directos y minimiza los costos de operación. Se realizaron pruebas computacionales con instancias de diferentes tamaños iniciando con 30 y aumentando el tamaño en 10 unidades hasta alcanzar 100 nodos. Se pudo evidenciar que con instancias menores a 70 se obtuvo una brecha de optimalidad menor al 30 %, mientras que para instancias superiores a 80 el gap supera el 50 %. En cuanto al tiempo de ejecución se nota que este aumenta a medida que aumenta la cantidad de nodos.

El segundo modelo (MDCTM) enruta a los pasajeros desde su origen a su destino mediante viajes directos y con una transferencias, se minimizan los costos de operación y maximiza los viajes directos. Este modelo tiene la particularidad que la función objetivo tiene un parámetro de peso $\lambda \in [0, 1]$. Se realizaron pruebas computacionales con instancias de tamaño 30 y 40 debido a que para instancias superiores a 40 nodos se supera las capacidades computacionales. Se evidenció que para las instancias ejecutadas el gap fue menor al 5 % tanto para el MDCTM como para su versión relajada.

Referencias bibliográficas

R. Borndörfer and M. Karbstein. Metric inequalities for routings on direct connections with application to line planning. *Discrete Optimization*, 18: 1–252, November 2015.

M. R. Bussieck. *Optimal lines in public rail transport*. PhD thesis, TU Braunschweig, 1997.

M. T. Claessens, N. M. van Dijk, and P. J. Zwaneveld. Cost optimal allocation of rail passenger lines. *European Journal of Operational Research*, 110(3):474–489, November 1998. URL https://ideas.repec.org/a/eee/ejores/v110y1998i3p474-489.html.

v. H. S. Goossens, J-W. and L. Kroon. A branch-and-cut approach for solving railway line-planning problems. *Transportation Science*, 38(3): 379–393, 2004.

M. M. P. D. K. L. P.-K. I. Ratajczak, P. and A. Kasprzak. Hybrid algorithm for line planning problems. *2013 International Symposium on Computational and Business Intelligence (ISCBI)*, pages 174–177, 2014.

K. Ravindra, L. Thomas, and B. James. *Network Flows:Theory,Algorithms, and Applications*. Prentice Hall, 1993.

R. Torres. Line planning and passenger routing problem with application to the Quito transportation system. *Int.J.Mathematics in Operational Research*, 19(3):332–353, 2021.

Capítulo A

Anexos

A.1. Código modelo DCTM

```
from gurobipy import *
import numpy as np
import pandas as pd
import random

costo_fijo=425
fmax = 20 #Frecuencia maxima de cada linea
kapa=180#Capacidad del modo de transporte
C_km=10 #Costo por kilometro recorrido

#Lectura del archivo txt
df = pd.read_csv("M_aleatoria_30_1.txt",sep=" ")

#Informacion de cada terminal
E=[list(i) for i in df.index]

#Lista con los nombres de las estaciones
Estaciones = [i[0] for i in E]

#Diccionario para la creacion de la matriz OD de demanda
Dic = {i[0]:pd.Series(i[3:], index = Estaciones, name = i[0])
       for i in E}

#Creacion del DataFrame que contiene la informacion de demanda de cada
```

```
   par OD
dfDemanda = pd.DataFrame(Dic)

#En la parte superior del DataFrame colocaremos el valor maximo entre
   cada par de nodos, es decir, max{d_{st},d_{ts}}
for i in range(0,len(Estaciones)):
    for j in range(0,len(Estaciones)):
        if i<j:
           dfDemanda.iloc[i,j]=max(dfDemanda.iloc[i,j],
                                   dfDemanda.iloc[j,i])

#Giro nos indica si el nodo es terminal,estacion de giro o si es una
   estacion donde no puede girar el trolebus, 1 si el nodo i es una
   estacion de giro, 2 si el nodo i es una terminal y 0 si es una
   estacion donde no se puede girar
Giro ={i[0]:i[1] for i in E}
#Estaciones_Giro contiene a todas las estaciones donde puede girar el
   trolebus
Estaciones_Giro = [i for i in Giro if Giro[i]==1]

#Terminales contiene todos los nodos que corresponden a un terminal
Terminales = [i for i in Giro if Giro[i]==2]

#lineas_terminales me ayuda a contabilizar cuantas lineas se generaran
   usando solo terminales, es decir, las lineas que van desde un
   terminal a otro terminal
lineas_terminales = 0
for i in range(0,len(Terminales)):
    lineas_terminales = lineas_terminales + len(Terminales[i:])-1

#Lineas contiene todas las lineas que existen en la red
Lineas = ["Linea_" +str(i)
          for i in range(1,len(Estaciones_Giro)*len(Terminales)+1+
          lineas_terminales)]

#Term_Est es una lista de tuplas que contienen el nodo inicial y final
   de cada linea donde el nodo inicial corresponde a un terminal y el
   nodo final es una estacion de giro
Term_Est=[(i,j) if int(i[i.find("_")+1:])+1 < int( j[j.find("_")+1:])+1
         else (j,i) for i in Terminales for j in Estaciones_Giro ]

#Term_Term es una lista de tuplas que contiene el nodo incial y final
   de cada linea donde el nodo incial corresponde a un Terminal y el
```

```
   nodo final tambien es un terminal
Term_Term=[(i,j) for i in Terminales for j in Terminales
          if int( i[i.find("_")+1:])+1 < int( j[j.find("_")+1:])+1]

Lineas_Tuplas=Term_Est + Term_Term

#Nodos_Lineas es una lista de listas de nodos que usa cada una de las
   lineas disponibles
Nodos_Lineas=[[k for k in
             range(int( i[i.find("_")+1:]),int( j[j.find("_")+1:])+1)]
              for (i,j) in Lineas_Tuplas]

#Distancia es un diccionario que nos indica la distancia que existe
   desde el Terminal 0 hasta el nodo i-esimo
Dist_0i ={i[0]:i[2] for i in E}

#Distancia de cada linea
distancia_lineas=[round(Dist_0i[j]-Dist_0i[i],2)
                  for (i,j) in Lineas_Tuplas]

# L contiene a las lineas de la red, en CF se encuentra el costo fijo
   de cada linea, CV tiene el costo variable por km de las lineas y NL
    los nodos de las lineas
 L, CF, CV ,NL = multidict(tupledict({Lineas[i]:
         (costo_fijo,round(C_km*distancia_lineas[i],2),Nodos_Lineas[i])
          for i in range(0,len(Lineas))}))


Estaciones_T = [i for i in range (0,len(Estaciones))]
dic_Estaciones = dict(zip(Estaciones,Estaciones_T))

#OD contiene a todos los pares OD y T contiene a los pares OD con su
   respectiva demanda
OD , T = multidict(tupledict({(dic_Estaciones[i],dic_Estaciones[j]):
       dfDemanda.iloc[dic_Estaciones[i],dic_Estaciones[j]]
       for i in dic_Estaciones for j in dic_Estaciones
       if dic_Estaciones[i]<dic_Estaciones[j]}))

#LP es un diccionario que tiene como clave las lineas y como valores
   listas de arcos usados por cada linea
LP ={j:[(i,i+1) for i in NL[j] if i<NL[j][-1]] for j in NL}

#DL es un diccionario que tiene como claves las lineas y como valores
```

```
   listas de pares OD de cada linea
DL={l:[(i,j) for i in NL[l] for j in NL[l] if i<j] for l in NL.keys()}

#Contiene los indices con los cuales indexaremos a la variable x
varX = tuplelist([(i,k,j) for (k,j) in OD for i in NL.keys()
       if NL[i][0]<=k and j<=NL[i][len(NL[i])-1]])

#Construccion conjunto para restriccion de asientos
TE = {(k,i,j):[(k,p,q) for (p,q) in DL[k] if p<=i and q>=j]
     for k in LP.keys() for (i,j) in LP[k]}

m = Model('DCT2_Matriz_30')

#Variables del modelo
y = m.addVars(L, vtype = GRB.BINARY, name="y")
f = m.addVars(L,vtype = GRB.INTEGER, name = "f")
x = m.addVars(varX,vtype = GRB.INTEGER, name = "x")
#Costos fijos
K = y.prod(CF,'*')

#Costos variables
C = f.prod(CV,'*')

#Funcion Objetivo
m.setObjective(K+C,GRB.MINIMIZE)

#Restricciones
Demanda_satisfecha = m.addConstrs((x.sum('*',i,j) == T[(i,j)]
                    for (i,j) in OD ), "Demanda")
Frecuencia = m.addConstrs((f[i]<=fmax*y[i] for i in L),"Frecuencias")
asientos = m.addConstrs((quicksum(x[l]
           for l in TE[(k,i,j)] ) <= kapa*f[k] for k in L
           for (i,j) in LP[k]),"Asientos")
m.Params.timeLimit = 1800.0
m.optimize()
```

Code Listing A.1: Código modelo DCTM

A.2. Código modelo MDCTM y MDCTM-R

Para correr el modelo MDCTM-R solamente hay que cambiar la línea 174 del siguiente código. En el parámetro vtype hay que colocar

GRB.CONTINUOUS.

```
from gurobipy import *
import numpy as np
import pandas as pd

landa = 0.20 #Parametro de peso
costo_fijo=425
fmax = 20 #Frecuencia maxima de cada linea
kapa=180 #Capacidad del modo de transporte
C_km=10 #Costo por kilometro recorrido

#Lectura de archivo txt
df = pd.read_csv("M_aleatoria_30_1.txt",sep=" ")

#Informacion de cada terminal
E=[list(i) for i in df.index]

#Lista con los nombres de las estaciones
Estaciones = [i[0] for i in E]

#Diccionario para la creacion de la matriz OD de demanda
Dic = {i[0]:pd.Series(i[3:], index = Estaciones, name = i[0])
       for i in E}

#Creacion del DataFrame que contiene la informacion de demanda de cada
    para OD
dfDemanda = pd.DataFrame(Dic)

#En la parte superior del DataFrame colocaremos el valor maximo entre
    cada par de nodos, es decir, max{d_{st},d_{ts}}
for i in range(0,len(Estaciones)):
    for j in range(0,len(Estaciones)):
        if i<j:
           dfDemanda.iloc[i,j]=max(dfDemanda.iloc[i,j],
                                   dfDemanda.iloc[j,i])

#Giro nos indica si el nodo es terminal,estacion de giro o si es una
    estacion donde no puede girar el trolebus, 1 si el nodo i es una
    estacion de giro, 2 si el nodo i es una terminal y 0 si es una
    estacion donde no se puede girar
Giro ={i[0]:i[1] for i in E}

```

```
#Estaciones_Giro contiene a todas las estaciones donde puede girar el
    trolebus
Estaciones_Giro = [i for i in Giro if Giro[i]==1]

#Terminales contiene todos los nodos que corresponden a un terminal
Terminales = [i for i in Giro if Giro[i]==2]

#lineas_terminales me ayuda a contabilizar cuantas lineas se generan
    entre terminales, es decir, las lineas que van desde un terminal a
    otro terminal
lineas_terminales = 0
for i in range(0,len(Terminales)):
    lineas_terminales = lineas_terminales + len(Terminales[i:])-1

#Lineas contiene todas las lineas que existen en la red
Lineas = [i for i in range (1,len(Estaciones_Giro)*len(Terminales)+1+
          lineas_terminales)]

#Term_Est es una lista de tuplas que contiene el nodo inicial y final
    de cada linea donde el nodo inicial corresponde a un terminal y el
    nodo final es una estacion de giro
Term_Est=[(int( i[i.find("_")+1:]),int( j[j.find("_")+1:]))
         if int( i[i.find("_")+1:]) < int( j[j.find("_")+1:])
         else (int(j[j.find("_")+1:]),int(i[i.find("_")+1:]))
         for i in Terminales for j in Estaciones_Giro ]

#Term_Term es una lista de tuplas que contiene el nodo inicial y final
    de cada linea donde el nodo inicial corresponde a un terminal y el
    nodo final es un terminal
Term_Term=[(int(i[i.find("_")+1:]),int( j[j.find("_")+1:])) for i in
         Terminales for j in Terminales
         if int( i[i.find("_")+1:])+1 < int( j[j.find("_")+1:])+1]

Lineas_Tuplas=Term_Est + Term_Term

#Lista que contiene los nodos de cada linea
Nodos_Lineas=[[k for k in range(i,j+1)] for (i,j) in Lineas_Tuplas]

#Dist_0i es un diccionario que nos indica la distancia que existe desde
     el Terminal 0 hasta el i-esimo nodo
Dist_0i ={int(i[0][i[0].find("_")+1:]):i[2] for i in E}
distancia_lineas=[round(Dist_0i[j]-Dist_0i[i],2)
                  for (i,j) in Lineas_Tuplas]
```

```
71
72 #L contiene a las lineas de la red, CF almacena el costo fijo de las
       lineas, CV contiene el costo variable por km recorrido de las
       lineas y LP contiene a los nodos de cada linea
73 L, CF, CV ,LP = multidict(tupledict({Lineas[i]:
74             (costo_fijo,C_km*distancia_lineas[i],Nodos_Lineas[i])
75             for i in range(0,len(Lineas))}))
76
77 #Arcos_LP contiene a los arcos usados por cada linea
78 Arcos_LP={i:[(j,j+1) for j in LP[i] if j != LP[i][-1]] for i in LP}
79
80 #Contiene a los terminales y estaciones de la red
81 list_Estaciones = [i for i in dfDemanda.columns]
82
83 #OD contiene a todos los pares OD y T contiene a los pares OD con su
       respectiva demanda
84 OD , T = multidict(tupledict({(i,j):
85          dfDemanda.iloc[int(i[i.find("_")+1:]),int(j[j.find("_")+1:])]
86          for i in list_Estaciones for j in list_Estaciones
87          if int(i[i.find("_")+1:])<int(j[j.find("_")+1:])}))
88
89 #Contiene los nodos de cada camino
90 C_1={(i,j):list(set(LP[i]+LP[j])) for i in LP for j in LP
91      if set(LP[i])&set(LP[j])!=set() and i<=j}
92
93 #Contiene toda la informacion de los caminos como por ejemplo la lineas
       que se unieron para formar el camino y cuales nodos pertenecen a
       cada linea
94 C_2=dict()
95 for i in LP:
96     for j in LP:
97         if i==j:
98            C_2[(i,j)]={i:Arcos_LP[i]}
99         if set(LP[i])&set(LP[j])!=set() and i<j:
100           if len(LP[i])<=len(LP[j]):
101                aux=list(set(Arcos_LP[j])-set(Arcos_LP[i]))
102                C_2[(i,j)]={i:Arcos_LP[i],j:aux}
103           else:
104                aux=list(set(Arcos_LP[i])-set(Arcos_LP[j]))
105                C_2[(i,j)]={i:aux,j:Arcos_LP[j]}
106
107 for (i,j) in C_2:
108     C_2[i,j][i].sort()
```

```
109     C_2[i,j][j].sort()
110 
111 #Informacion sobre la demanda de la linea
112 DL={i:[(k,l) for k in range(LP[i][0],LP[i][-1]+1)
113     for l in range(LP[i][0],LP[i][-1]+1) if k<l] for i in LP}
114 
115 #Nodos de cada camino
116 C_1_aux=list()
117 for i in C_1:
118     C_1_aux.append(list(C_1[i]))
119 
120 # Caminos con sus respectivos nodos
121 Paths={i+1:C_1_aux[i] for i in range(0,len(C_1_aux))}
122 
123 #Ordeno de forma ascendente los nodos de cada camino
124 for i in Paths:
125     Paths[i].sort()
126 
127 #Nos indica como de que lineas se formo el camino por ejemplo k:(i,j)
    nos dice que el k-esimo camino se formo de unir los nodos de las
    lineas i y j
128 Inf_Transferencias=dict(zip(Paths,C_2))
129 
130 #Caminos_Transferencia contiene a todos los caminos que tiene una
    transferencia
131 Caminos_Transferencias = [k for k in Inf_Transferencias
132                 if Inf_Transferencias[k][0]!=Inf_Transferencias[k][1]]
133 
134 #Nos da el costo para cada camino 0 si el camino no tiene
    transferencias y 1 si las tiene
135 Cp={i:1 if i[0] in [j for j in Caminos_Transferencias]
136     else 0 for i in varX}
137 
138 #Contien los indices con los cuales indexaremos a la variable x
139 varX = tuplelist([(i,int(k[k.find("_")+1:]),int(j[j.find("_")+1:]))
140       for (k,j) in OD for i in Paths.keys()
141       if Paths[i][0]<=int(k[k.find("_")+1:])
142       and int(j[j.find("_")+1:])<=Paths[i][len(Paths[i])-1]])
143 
144 #En este diccionario se almacena la informacion de los caminos por
    ejemplo (i,j):k nos indica que el camino k esta formado por los
    nodos de las lineas i y j
145 Inf_Transferencias2=dict(zip(C_2,Paths))
```

```
146
147 #ODPL contiene los pares OD de cada linea por nodo
148 ODPL=dict()
149 for (i,j) in C_2:
150     if i==j:
151         ODPL[(i,j)]={(i,k,l):[(Inf_Transferencias2[(i,j)],s,t)
152                     for s in C_1[(i,j)] for t in C_1[(i,j)] if s<=k
153                     and t>=l]for (k,l) in C_2[(i,j)][i]}
154     else:
155          aux_i=dict()
156          aux_j=dict()
157          aux_i[(i,j)]={(i,k,l):[(Inf_Transferencias2[(i,j)],s,t)
158                       for s in C_1[(i,j)] for t in C_1[(i,j)] if s<=k
159                       and t>=l] for (k,l) in C_2[(i,j)][i]}
160          aux_j[(i,j)]={(j,k,l):[(Inf_Transferencias2[(i,j)],s,t)
161                      for s in C_1[(i,j)] for t in C_1[(i,j)] if s<=k
162                      and t>=l]for (k,l) in C_2[(i,j)][j]}
163          ODPL[(i,j)]={**aux_i[(i,j)],**aux_j[(i,j)]}
164
165 #Este conjunto nos sirve para crear la restriccion de asientos
166  R2={(k,i,j):[(p,s,t) for (m,n) in C_2 if (k,i,j) in ODPL[(m,n)]
167      for (p,s,t) in ODPL[(m,n)][k,i,j]]
168
169 m = Model('Modelo_Transferencias')
170
171 #Variables del modelo
172 y = m.addVars(L, vtype = GRB.BINARY, name="y")
173 f = m.addVars(L,vtype = GRB.INTEGER, name = "f")
174 x = m.addVars(varX,vtype = GRB.INTEGER, name = "x")#Para correr el
    modelo MDCTM-R hay que colocar vtype=GRB.CONTINUOUS
175
176 #Costos Fijos
177 K = y.prod(CF,'*')
178
179 #Costos Variables
180 C = f.prod(CV,'*')
181
182 #Costos Transferencias
183 Costo_Transferencia = x.prod(Cp,'*','*')
184
185 #Funcion Objetivo
186 m.setObjective(landa*(K+C)+(1-landa)*Costo_Transferencia,GRB.MINIMIZE)
187
```

```
#Restricciones
Demanda_satisfecha = m.addConstrs((x.sum('*',int(i[i.find("_")+1:]),int
    (j[j.find("_")+1:])) == T[(i,j)] for (i,j) in OD ), "Demanda")
Frecuencia = m.addConstrs((f[i]<=fmax*y[i] for i in L),"Frecuencias")
asientos = m.addConstrs((quicksum( [x[n]
          for n in R2[(k,i,j)]] )<= kapa*f[k]
          for k in LP.keys() for (i,j) in Arcos_LP[k] ),"Asientos")
m.Params.timeLimit=1800.0
m.optimize()
```

Code Listing A.2: Código modelo MDCTM