Proyecto Fin de Máster

Organización Industrial y Gestión de Empresas

Diseño de redes de autobuses y establecimiento de la flota

Autor: ANTÍA FERNÁNDEZ OLVEIRA

Tutor: ALICIA DE LOS SANTOS PINEDA

Dpto. Organización Industrial y Gestión de Empresas I

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2023

Proyecto Fin de Máster

Organización Industrial y Gestión de Empresas

Diseño de redes de autobuses y establecimiento de la flota

Autor: ANTÍA FERNÁNDEZ OLVEIRA

Tutor: ALICIA DE LOS SANTOS PINEDA

Dpto. Organización Industrial y Gestión de Empresas I

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2023

AGRADECIMIENTOS

Mis agradecimientos a mi tutora Alicia, por darme la idea del trabajo y creer en mí para llevarlo a cabo. Gracias por la ayuda, por todas las correcciones realizadas y las interminables tardes en el despacho cuando nos salía algún error en el código a última hora.

A los profesores que, durante todo el máster me han formado y convertido en lo que hoy soy. A lo largo de mi vida siempre me he quedado con lo mejor de cada persona que ha pasado por mi vida y, del máster me llevo muchos conocimientos importantes, tanto de la vida como a nivel profesional. A Davis Canca, Eva Barrena y Alicia de los Santos por dejarme usar su trabajo original y ampliarlo hasta lograr este trabajo. Espero haber estado a la altura.

Y de nuevo a Alicia, por estar dispuesta a aguantarme un poco más para seguir desarrollando este trabajo y seguir mejorándolo.

RESUMEN

Los problemas relativos a planificación de transportes son de enorme complejidad, motivo por el cual suelen tratarse por etapas; como diseño de rutas, determinación de frecuencias, determinación de horarios, de la flota, etc. Aunque desde un punto de vista computacional dividir por etapas hace los problemas más manejables, desde la perspectiva de encontrar el óptimo, nos alejamos. Hay una fuerte tendencia en la integración de etapas más que en la separación de las mismas.

El presente proyecto se centra en la integración de las etapas diseño de rutas, determinación de frecuencias y flota. El problema consiste en diseñar las líneas del transporte público de autobuses desde cero, determinando así, para cada línea, su itinerario, paradas terminales, frecuencia y flota (número y tipo de autobús). Al mismo tiempo, se establecen las rutas que los pasajeros seguirían según el criterio de minimizar tiempo de viaje. Para ello, se utiliza una red de referencia con sus paradas y su demanda, pero sin considerar un conjunto de líneas candidatas. Se considera una red bimodal compuesta por dos capas según en modo de transporte: capa de peatón y capa de autobús. Para modelar el problema se presenta un modelo de programación lineal entero mixto.

Se realizan experimentos computacionales sobre una red bastante usada en la literatura: la red de *Mandl's, 1980* [1] compuesta por 15 nodos y 21 arcos bidireccionales que representan la unión de 15 ciudades suizas. Sobre esta red se consideran distintos escenarios considerando distintas combinaciones de los parámetros de entrada. Para cada escenario se resuelve el modelo matemático cuyo principal objetivo es buscar una red que resulte lo más atractiva posible para los pasajeros. Se tendrá en cuenta el tiempo total de viaje del pasajero, considerando los tiempos de espera, los tiempos de viaje en el autobús, tiempos de caminata y tiempos de transbordo.

Dado que el objetivo del pasajero y el del operador son contrapuestos, finalmente se espera conseguir un equilibrio entre el tiempo de viaje y el costo de la flota, es por ello por lo que se restringe el número de autobuses disponibles.

ABSTRAT

The problems related to transport planning are extremely complex, which is why they are usually dealt with in stages; such as route design, determination of frequencies, determination of schedules, of the fleet, etc. Although from a computational point of view dividing by stages makes the problems more manageable, from the perspective of finding the optimum, we are far from it. There is a strong tendency to integrate stages rather than separate them.

This project focuses on the integration of the route design, frequency determination and fleet stages. The problem consists in designing the public transport bus lines from scratch, thus determining, for each line, its itinerary, terminal stops, frequency and fleet (number and type of bus). At the same time, the routes that passengers would follow according to the criteria of minimizing travel time are established. To do this, a reference network is used with its stops and demand, but without considering a set of candidate lines. A bimodal network is considered composed of two layers depending on the mode of transport: pedestrian layer and bus layer. To model the problem, a mixed integer linear programming model is presented.

Computational experiments are carried out on a network widely used in the literature: the Mandl's network, 1980 [1] composed of 15 nodes and 21 bidirectional arcs that represent the union of 15 Swiss cities. On this network, different scenarios are considered considering different combinations of input parameters. For each scenario, the mathematical model is solved whose main objective is to find a network that is as attractive as possible for passengers. The passenger's total travel time will be taken into account, considering waiting times, bus travel times, walking times and transfer times.

Since the objective of the passenger and that of the operator are opposed, finally it is expected to achieve a balance between travel time and the cost of the fleet, which is why the number of buses available is restricted.

ÍNDICE

1. Introducción

Los sistemas de transporte público, y en particular los autobuses, han sido estudiados muy a menudo en la literatura. Estos estudios consideran distintos objetivos, distintos enfoques o distintas variables. La planificación de redes de transporte es una actividad de enorme complejidad. Así, los problemas de planificación de transporte público se subdividen en una serie de problemas entrelazados entre sí que se organizan de manera jerárquica. En el caso de redes de autobuses, se subdividen en: diseño de red, diseño de líneas, planificación de horarios, determinación de la flota y gestión del personal.

Esta descomposición en etapas hace que los problemas de transportes sean más manejables desde el punto de vista computacional, pero al mismo tiempo, las soluciones obtenidas se alejan del óptimo global del problema. Estas divisiones no son independientes unas de otras y suelen estar entrelazadas. Actualmente, hay una fuerte tendencia a considerar la integración de las diferentes etapas para evitar obtener soluciones subóptimas. En especial, este trabajo se centra en la integración del problema de diseño de red y el de líneas, incluyendo la determinación de frecuencias y material rodante. De esta forma, la solución del problema considerado estaría formado por un conjunto de líneas, donde cada línea está determinada por un origen, un destino, un itinerario, frecuencia y flota.

En el problema del diseño de redes, primera etapa en la descomposición de problemas de transporte se tiene como datos de entrada una red subyacente formada por un conjunto de paradas potenciales y sus conexiones (ver **Figura 1**). Estas paradas y sus conexiones se seleccionan creando un subconjunto de acuerdo con algún criterio como tiempo de viaje, frecuencia, beneficio… El problema puede tratarse a través de modelos matemáticos o metaheurísticos. Las variables principales serían la selección de paradas y las conexiones, teniendo en cuenta las restricciones del diseño.

Tradicionalmente, cuando se diseña una red de autobuses, se parte de un conjunto potencial de líneas, llamado pool de líneas, y, a partir de éstas, se selecciona el subconjunto más conveniente de acuerdo con el objetivo establecido. El problema podría modelarse definiendo tantas variables binarias como líneas en el pool de líneas

haya. Esto hace que el problema pueda resolverse con instancias de redes con bastantes nodos (puede trabajarse con redes reales) y admita otro tipo de variables, como las frecuencias.

Figura 1: Diseño de una red de transporte en la ciudad de Sevilla

Los problemas de transporte suelen tener tres perspectivas: el pasajero, el operador y la comunidad. Generalmente el objetivo del problema, desde el punto de vista del pasajero, consiste en obtener el mayor número de viajes directos, o pocos transbordos, cubriendo la mayor área posible y minimizando el tiempo de viaje total del pasajero (*Guihaire and Hao, 2008*) [2]. Desde el punto de vista del operador, cuyo objetivo principal es mover al máximo número de pasajeros al menor coste, se puede destacar la longitud de las líneas, costes de operación de la flota de autobuses, número de líneas. La comunidad actúa como balance entre los objetivos del operador y de los pasajeros, dado que sus fines son sociales.

En los problemas con multi objetivos, donde ambas perspectivas son consideradas, se puede usar un peso representando la importancia de cada objetivo (*Iliopoulou et al., 2019*) [3] o minimizar ambas perspectivas a la vez. *Chew et al. (2013)* [4] establece un número límite de líneas, nodos por líneas y número de pasajeros. Otros muchos autores configuran las rutas a partir de un pool de líneas candidatas mediante algoritmos

heurísticos (*Pattnaik et al., 1998* [5]; *Bielli et al., 2002* [6]; *Chakroborty, 2003* [7]; *Tom and Mohan, 2003* [8]; *Lee and Vuchic, 2005* [9]; *Zhao and Zeng, 2007* [10], [11]). Para *Marwah et al. (1984)* [12], el problema se puede resolver con la combinación de dos métodos: asignando la demanda a las líneas y diseñando un pool de líneas con la finalidad de minimizar el número de transbordos a realizar. En *Guan et al. (2009)* [13] se propone que, si el problema consiste en una selección de líneas de un pool de líneas que conecte todas las estaciones de una infraestructura, minimizando el largo total de las líneas, la función objetivo será una combinación del recorrido total, el tiempo de viaje del pasajero y el número de pasajeros transportados.

El algoritmo heurístico propuesto por *Van Nes et al. (1988)* [14] utiliza un pool de líneas, con el mayor número de viajes directos, y determina unas frecuencias base. Dichas frecuencias se van aumentando hasta llegar al límite de presupuesto y el tamaño de flota permitido. En *Farahani et al. (2013)* [15] se hace una revisión de la formulación de problemas de transporte y las soluciones algorítmicas obtenidas. *Borndörfer et al. (2005)* [16] presenta un modelo para el diseño de red y el ajuste de frecuencias, cuyo objetivo es minimizar el total del tiempo de viaje de los pasajeros y el coste del operador. Para ello, la ruta que los pasajeros puede seguir es libre y las líneas se generan dinámicamente. *Borndörfer et al. (2007)* [17] añade las frecuencias a su anterior trabajo y su objetivo ahora es minimizar el coste de la operación y el total del tiempo de viaje del pasajero, sin considerar tiempo de transbordo o espera. A diferencia de la mayoría de los estudios, en vez de usar un pool de líneas, opta por usar un conjunto dado de nodos de parada.

Hay redes de transporte público que solo buscan mover al pasajero desde las zonas más marginales. Estos sistemas están lejos de los objetivos de obtener beneficios o reducir los costes y necesitan la subvención de la administración pública (*Winston and Maheshri, 2007*) [18].

La literatura en cuanto a modelos matemáticos todavía es escasa y solo han sido probados en líneas muy pequeñas (*Zhang et al., 2014*) [19]. Sigue habiendo una brecha entra el modelo estratégico y el táctico a la hora de considerar más de un modo de transporte (*Camporeale et al., 2016*) [20].

El artículo *De los Santos et al, (2021)* [21] es el primero en modelar el problema sin definir un pool de líneas, dejando libertad para la creación de las mismas, considerando una red bimodal pedestre-pública para modelar las rutas de los pasajeros. Las variables de diseño permiten definir las líneas como un conjunto de arcos conexos, puesto que los arcos de cada línea son variables del modelo. Además, este trabajo, incorpora el problema del reparto de la demanda, cuyos trayectos tienen en cuenta la red peatonal y los posibles transbordos entre líneas. La frecuencia de los autobuses no es una variable, sino que es un parámetro. Se realizan experimentos computacionales sobre redes reales considerando distintos conjuntos de frecuencias. Se empieza disponiendo una red subyacente potencial representando la infraestructura de autobús y otra red potencial con la red peatonal. Ambas redes cuentan con su propio conjunto de nodos (que pueden coincidir) y su propio conjunto de arcos específicos. El problema consiste en encontrar la localización de las paradas de autobús y definir las líneas, teniendo en cuenta la asignación de pasajeros. El objetivo es minimizar el tiempo total de viaje por parte de todos los pasajeros que usan la red bimodal, teniendo en cuenta los tiempos en el autobús, los tiempos de transbordo, los tiempos de espera y los tiempos peatonales, contando al mismo tiempo con restricciones del operador (número máximo de líneas, número de paradas por línea o el largo máximo permitido de la línea).

Este Trabajo de Fin de Máster es una extensión del citado artículo *De los Santos et al, (2021)* [21]. Se incorporan nuevas variables al modelo, lo que implica nuevos conjuntos de restricciones y sus correspondientes linealizaciones. Las frecuencias, las capacidades de las líneas, la flota de autobuses y el tipo de autobuses son variables de decisión del nuevo modelo, lo que conlleva a una reformulación del trabajo de *De los Santos et al, (2021)* [21] y la inclusión de nuevas restricciones.

La organización de este trabajo está dividida en secciones. En la Sección 1 se comentan los objetivos principales del trabajo y la revisión de la literatura. En la Sección 2 se explica la motivación del trabajo. En la Sección 3 se describe el problema. La Sección 4 engloba las consideraciones a tener en cuenta para desarrollar el problema. En la Sección 5 se detalla el modelo matemático, con los datos, variables y restricciones. La Sección 6 detalla la implementación del modelo en Python. Finalmente, en la Sección 7 y la Sección 8 se estudian las pruebas llevadas a cabo.

2. Justificación y objetivo

Cada vez son más las ciudades cuyos centros urbanos no son transitables con vehículo propio. Esto, en muchas ocasiones, surge de la necesidad de conservar el patrimonio de los cascos históricos y, en las que no, la construcción de los cascos antiguos de las ciudades fue hecha aprovechando el espacio al máximo y sin tener en cuenta la posibilidad futura de vehículos de transporte, cuyo único método era el caballo y el coche, nada más que un sueño.

Esta necesidad de conservación se extiende al resto de la ciudad moderna cuando la maximización del vehículo comienza a suponer un peligro para la calidad de vida de las personas. La evolución ha conseguido que, los trayectos que antes se realizaban en días, puedan hacerse en horas, lo que genera una gran conectividad entre las personas y eso está bien, ya que el ser humano es un ser sociable que necesita la interacción con su entorno. Pero el transporte masivo también conlleva a una nueva contaminación masiva.

Los cascos históricos intentan solucionar esto mediante la incorporación de líneas de bus que, permiten el transporte de un gran número de pasajeros a la vez y contaminan menos que la gran cantidad de coches particulares que pasarían por el mismo lugar en su defecto. Si se amplía la mira, se puede conseguir que toda la ciudad (provincia, país…) esté conectada por líneas de bus óptimas y eficaces.

A parte de reducir la contaminación, se estaría consiguiendo que las personas sin capacidad económica de adquisición de su propio vehículo puedan desplazarse, así como también que aquellos que no son aptos para el manejo de un coche (causas físicas o edad), tengan la posibilidad de moverse por donde necesiten.

El principal objetivo de este proyecto es diseñar una red de transporte de autobuses atractiva para los pasajeros, teniendo en cuenta el tiempo total de viaje del pasajero que se mueve de un punto a otro de la red.

Al considerarse una red bimodal, donde la red pedestre y la de bus se diseñan conjuntamente de modo cooperativo, se está diseñando una red de autobús en la que se está poniendo en valor el modo pedestre, que no se tiene en cuenta en otros estudios.

Por tanto, este trabajo se centra en determinar el conjunto de líneas de autobuses (paradas y conexiones), itinerario, frecuencia de cada línea, número de autobuses por cada línea y tipo de autobús, minimizando el tiempo total por parte de todos los pasajeros que usan la red bimodal. Se entiende por tiempo total a la suma del tiempo que tardan todos los pasajeros en llegar andando a la parada de autobús (a través de la red pedestre), el tiempo que tarda el autobús en llegar a la parada (tiempo de espera) y el tiempo que tarda el autobús en llegar al destino (a través de la red de autobús). En caso de que el trayecto no sea directo y sea necesario hacer algún tipo de transbordo, se tendrá en cuenta el tiempo de espera al siguiente autobús si el transbordo se realiza en la misma parada o el tiempo de traslado a la parada de la línea a la que se hace transbordo más tiempo de espera al próximo autobús, si no es la misma parada.

Este estudio pretende motivar las siguientes acciones:

- Uso eficiente del transporte público en zonas urbanas protegidas.
- Reducción de la contaminación general de la ciudad, causada por los vehículos de uso particular.
- Ganar espacios verdes en las ciudades al no necesitar tantos lugares de aparcamiento.
- Permitir el desplazamiento a aquellas personas que no sean aptas para el uso de vehículo (menores de edad, mayores, ciertas discapacidades…) y aquellas personas que no pueden o no quieren disponer de su propio vehículo.

3. Descripción del problema

Los sistemas de transporte público, en general, se pueden definir como un conjunto de líneas. Cada línea está determinada por un origen, un destino, un conjunto de paradas y las conexiones entre ellas. Esto puede aplicarse a los sistemas de los autobuses, los trenes o los tranvías.

Como comentamos, en los problemas relativos al diseño de redes de transporte público, se parte de una red subyacente, es decir, una red que contiene información sobre el conjunto de paradas potenciales y sus conexiones.

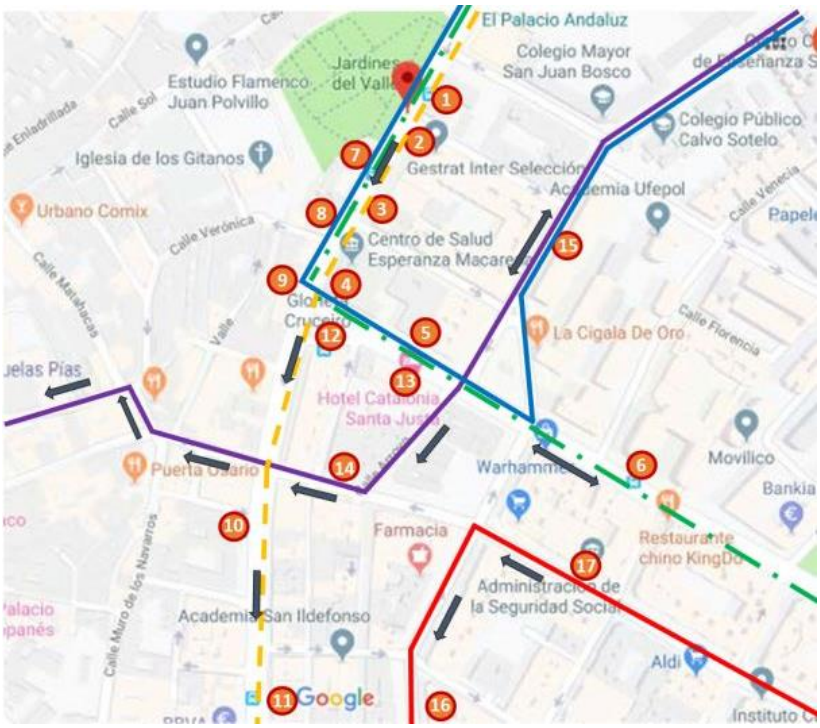


Figura 2: Red de autobuses

En este trabajo se considera una red bimodal constituida por una red de autobuses y una red pedestre. En el caso de la red de autobuses, podría verse como la red de carreteras de la zona que involucra las paradas (**Figura 2**). En el caso de la red pedestre, serían las calles por donde transitan los pasajeros (**Figura 3**).

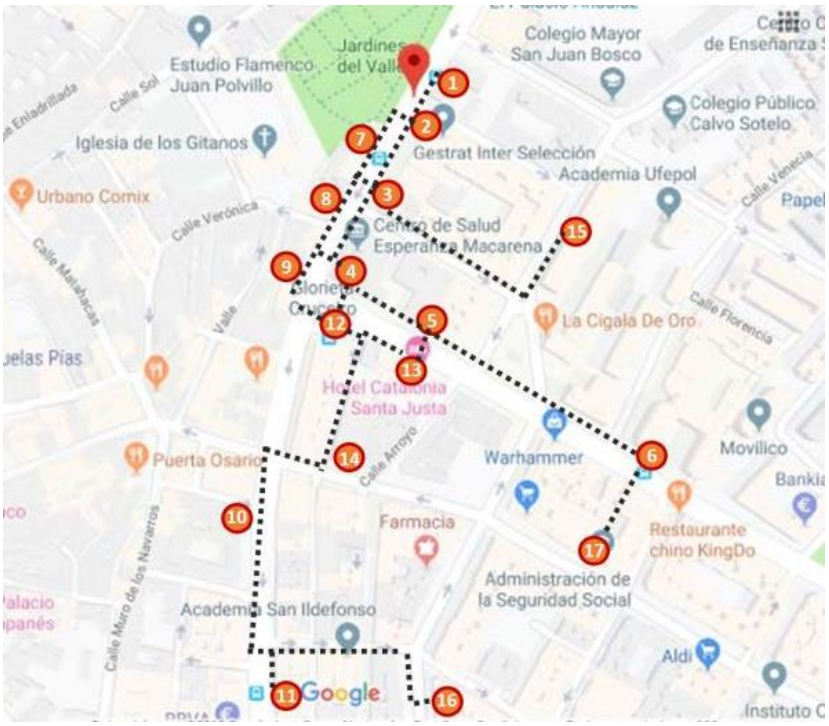


Figura 3: Red pedestre

En este trabajo estamos considerando una red bimodal, en la que se consideran tanto la red de autobús como la red pedestre para describir los trayectos de los pasajeros (**Figura 4**).

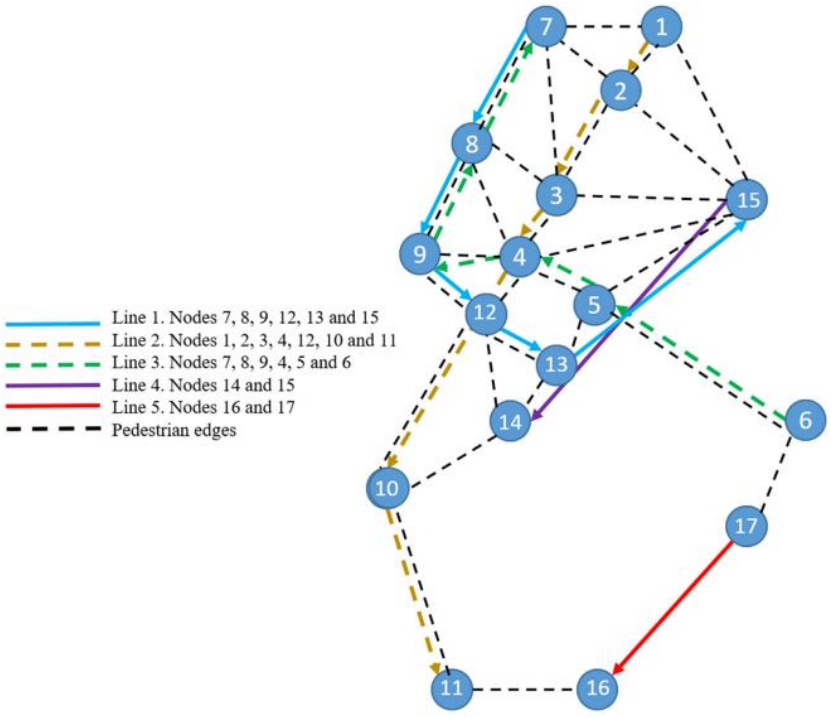


Figura 4: Red bimodal

Los pasajeros quieren realizar su viaje lo más rápido posible, con las esperas más cortas y los menores transbordos necesarios. Para obtener el camino más corto se usa el algoritmo de *Dijkstra (1959)* [22].

Por ejemplo, si se quiere realizar un viaje desde el nodo 17 al nodo 4, no habrá transbordo entre líneas ya que los pasajeros pueden caminar desde el nodo 17 al 6 sobre la red peatonal y luego pueden moverse en autobús hasta llegar al destino (nodo 17), haciendo una combinación de modos sin hacer transbordo entre líneas (ver **Figura 5**).

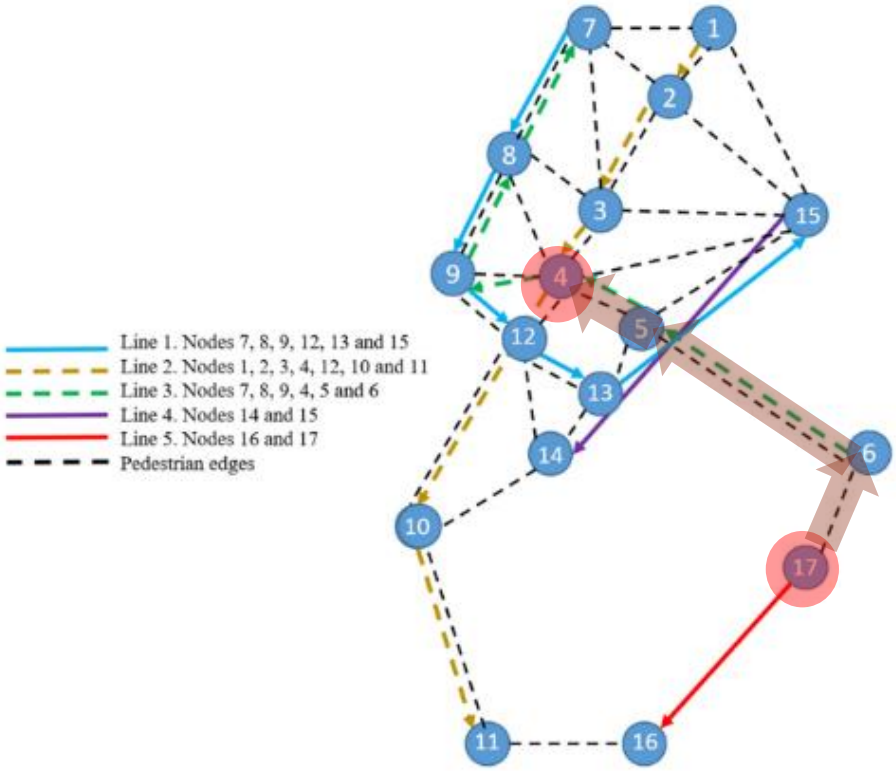


Figura 5: Trayecto del nodo 17 al nodo 4

En caso de ir del nodo 7 al nodo 10, una ruta posible consistiría en ir del nodo 7 al nodo 12 en la línea de autobús 1 (azul) y hacer un transbordo a la línea de autobús 2 (amarillo) en el nodo 12 (ver **Figura 6**). Para el segundo tipo de transbordo, para ir del nodo 6 al nodo 15, el pasajero puede usar la línea de autobús 3 (verde) desde el nodo 6 hasta el nodo 5, andar desde el 5 hasta el nodo 13 y, finalmente tomar la línea 1 de autobús (azul) hasta llegar al nodo 15 (ver **Figura 7**).

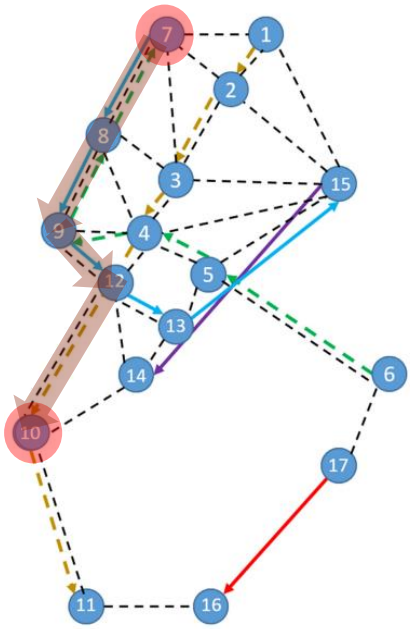


Figura 6: Transbordo en la misma parada

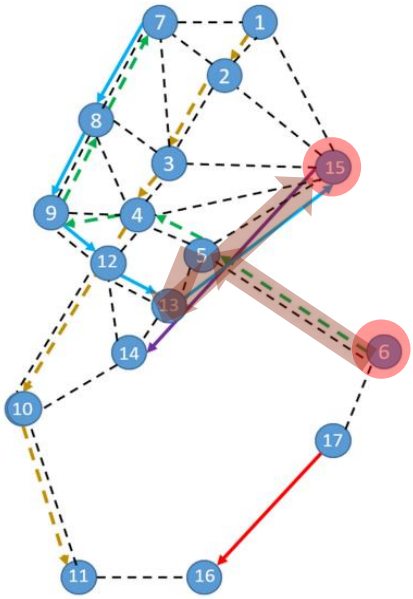


Figura 7: Transbordo en diferentes paradas

Para el desarrollo de este trabajo se busca encontrar las localizaciones más convenientes de las paradas de autobús según los movimientos de los pasajeros y diseñar las líneas considerando la afluencia de estos.

El pasajero podrá optar por el modo de transporte pedestre para viajar desde su origen a su destino o combinarlo con el modo de transporte de bus, según su conveniencia, creando así una red de transporte bimodal. Con la finalidad de representar esta situación, se puede definir una nueva red con las dos capas superpuestas: la de los movimientos permitidos a en modo peatón (a pie, el pasajero andando) y la de la subestructura vial por la que circulan los buses.

El principal objetivo del modelo es orientado al pasajero, por lo que se busca minimizar el tiempo total de viaje, considerando el tiempo de traslado hasta la primera parada, el tiempo de espera del bus, el tiempo de trayecto dentro del bus, el tiempo de traslada desde la parada final al destino y si lo hubiera, el tiempo de transbordo entre autobuses (incluyendo el desplazamiento andando -si hay- y la espera en la segunda parada). Se podría considerar en una siguiente instancia al operador añadiendo restricciones al problema que incorporen las preferencias del operador tales como el coste. Aunque también hay que tener en cuenta que para que el operador le saque el

mayor rendimiento a un proyecto semejante, es imprescindible obtener un cliente satisfecho que pasa por el objetivo de minimizar tiempos de viaje. Con esto en cuenta, los pasajeros de cada par origen-destino (OD) deben seguir la ruta más corta, que dependerá del diseño de la red de autobús.

Se modela la capa de autobús con el grafo $G = G(S, A)$ que representa la subestructura vial definida por un conjunto de nodos S de posibles paradas de autobús y un conjunto de arcos A que representan las posibles conexiones entre los nodos.

La capa peatonal se representa con otro grafo $G' = G(S, E)$, donde S son el conjunto de localizaciones centroides o paradas de autobús y el conjunto de aristas E que simboliza el paso entre posibles paradas de la red peatonal.

Se entiende por transbordo a la transferencia del pasajero entre autobuses de diferentes líneas para poder llegar a su destino en caso de no existir una línea directa para su trayecto. Para ser más fieles a la realidad, se consideran dos tipos de transbordo:

- Transbordo entre líneas de autobús en la misma parada.
- Transbordo entre líneas de autobús que se encuentran en diferentes paradas, lo que ocasiona un tiempo extra que se traduce en que el peatón debe caminar (modo peatón y uso de las aristas de paso E) hasta alcanzar la nueva parada.

De la superposición de ambas capas (autobús y peatonal) se obtiene la red bimodal cuyo grafo se representa como $BG = G(S, E \cup A)$. Sobre este grafo se busca la ruta más corta y directa para minimizar el tiempo de transporte del pasajero, lo que conlleva que cada camino esté compuesto por aristas peatonales, arcos de autobús y paradas.

La frecuencia de los buses, que corresponde al número de veces que un autobús perteneciente a una línea pasa por una parada, vendrá dada por una lista de frecuencias en una franja de tiempo establecida, en este caso, número de autobuses que pasan cada 60 minutos. En contrapartida, el headway, que corresponde al tiempo que tarda en pasar un autobús de una línea por una parada concreta, estará determinada por una lista de headways para el mismo tiempo dado de la frecuencia. De este modo, el producto de la frecuencia por el headway debe dar, en todos los casos, el total del tiempo bajo el que se estudian ambos conceptos: 60 minutos.

Las frecuencias más altas afectan directamente a los costes del operador, que influyen en el total de la flota requerida. Por otro lado, ayuda a un sistema de transporte más eficiente y, por lo tanto, más atractivo para los pasajeros ya que esperarán menos tiempo al autobús. El número de autobuses para operar en la red es esencial para transportar a los pasajeros y representar el coste del operador.

4. Consideraciones previas

Algunas de las consideraciones a tener en cuenta durante el desarrollo del problema considerado en este proyecto son las siguientes:

- Las líneas son bidireccionales, puesto que los nodos son paradas físicas y las paradas cambian según el sentido.
- No se considera un conjunto dado de líneas, no hay líneas candidatas, todas parten de cero y se construyen a partir de arcos y nodos.
- La red bimodal se representa con el grafo $BG = G(S, E \cup A)$, resultado de la unión de las capas de autobuses y peatonal.
- Cada arco del conjunto A de la capa autobús (direccional), tiene asociado un tiempo de viaje, teniendo en cuenta la velocidad media a la que se mueve el autobús.
- Cada arista del conjunto E de la capa peatonal (bidireccional), tiene asociado un tiempo de viaje, en el que se considera la velocidad media a la que se mueve caminando un peatón.
- Para cada par origen-destino OD, los pasajeros de desplazan por el grafo BG siguiendo el camino con las combinaciones más convenientes de transporte y transbordo, de acuerdo con la función objetivo (el tiempo total de viaje por parte de todos los pasajeros).
- Los pasajeros que solo usan el modo peatonal se consideran demanda no atendida por el autobús.
- Hay un máximo de líneas de autobús a diseñar.
- Para cada OD debe existir un camino que conecte el origen y el destino a través de la red bimodal.

- Se establecerá un máximo de flota posible a repartir entre las diferentes líneas, según la necesidad de cada una.
- Si la capacidad de la línea no es suficiente para satisfacer la demanda, la demanda puede realizar el trayecto mediante la red pedestre.
- Se tienen en cuenta restricciones del operador al considerar limitación de flota, longitud de las líneas y número de paradas de la línea.

5. Modelo matemático

En esta sección se describe el modelo matemático para el problema que estamos tratando. Se detallan los datos de entrada y notaciones, variables del modelo, función objetivo, restricciones y linealizaciones de variables y restricciones para hacer obtener un modelo lineal.

5.1. Datos y nomenclaturas

- $S = \{1 \dots n\}$, conjunto de posibles nodos (paradas) de bus.
- $A = \{(i,j): i,j \in S\}$, conjunto de posibles arcos de la red de autobuses.
- d_{ij} , longitud de cada arco $(i,j) \in A$, en kilómetros.
- $E = \{\{i,j\}: i,j \in S\}$, conjunto de posibles aristas de paso permitidas entre las paradas.
- $G = G(S,A)$, grafo dirigido que representa la capa de modo bus.
- $G' = G(S,E)$, grafo no dirigido que representa la capa de modo peatonal.
- $BG = G(S, E \cup A)$, grafo dirigido que representa la red bimodal formada por las capas bus-pedestre.
- $\Theta = \{\theta_1 \dots \theta_{|\Theta|}\} \subset S \times S$, conjunto de pares origen-destino OD. Cada elemento $\theta = (\theta^o . \theta^d) \in \Theta$ está definido por un origen, θ^o y un destino, θ^d.
- $f_1, .., f_k$ lista de frecuencias para una línea de bus.
- $\mathcal{F}^{max}$, frecuencia máxima que puede tener una línea de bus.
- $h_1, .., h_k$, lista de headway para una línea de bus, cada valor viene dado por el cociente del tiempo entre su valor correspondiente de la lista de frecuencias.
- fl^{max}, número máximo de autobuses permitidos en toda la flota.
- fl_1^{max}, número máximo de autobuses simples.

- fl_2^{max}, número máximo de autobuses dobles.
- $\mathcal{C}_1$, número de pasajeros que pueden viajar en un autobús doble.
- $\mathcal{C}_2$, número de pasajeros que pueden viajar en un autobús simple.
- p_θ, número de pasajeros por hora que viajan desde el origen θ^o hasta el destino θ^d del par OD θ.
- ℓ, índice genérico para denotar una línea de bus.
- $\mathcal{L}$, conjunto de posibles líneas de autobús.
- $\mathcal{L}^{max}$, número máximo de líneas de autobús permitidas en la red.
- S_{min}, límite inferior del número de paradas que puede tener una línea de bus.
- S_{max}, límite superior del número de paradas que puede tener una línea de bus.
- $\mathcal{M}$, longitud máxima permitida para la línea de bus.
- v_{ped}, velocidad media de los peatones, en kilómetros/ hora.
- v_{bus}, velocidad media del bus, en kilómetros/ hora.

5.2. Variables

- $s_i^\ell = 1$, si la parada i pertenece a la línea ℓ, 0 en caso contrario.
- $u_{ij}^\ell = 1$, si el arco $(i,j)\ \epsilon\ A$ es utilizado por la línea ℓ, 0 en caso contrario.
- $r_{ij}^{\theta\ell} = 1$, si el flujo del par θ atraviesa el arco $(i,j)\ \epsilon\ A$ usando la línea ℓ, 0 en caso contrario.
- $t_{ij}^\theta = 1$, si el par θ atraviesa la arista peatonal $\{i,j\}\ \epsilon\ E$, 0 en caso contrario.
- $v_i^{\theta\ell\ell'} = 1$, si los pasajeros correspondientes al par θ hacen transbordo en la parada i de la línea ℓ hacia la línea ℓ', 0 en caso contrario.
- $w_i^{\theta\ell'} = 1$, si pasajeros correspondientes al par θ cambian del modo peatonal al modo autobús de la línea ℓ'en la parada i.
- $m^\ell = 1$, si existe la línea de bus, 0 en caso contrario.
- fr^ℓ, frecuencia de la línea ℓ, variable entera.
- hw^ℓ, headway correspondiente a la frecuencia de la línea ℓ, variable entera.
- z^ℓ, flota requerida para cumplir la demanda de la línea ℓ, variable entera.
- z_1^ℓ, flota de la línea ℓ del tipo de autobús simple, variable entera.
- z_2^ℓ, flota de la línea ℓ del tipo de autobús doble, variable entera.
- $\delta_1 = 1$, si el autobús es simple, 0 en caso contrario.

- $\delta_2 = 1$, si el autobús es doble, 0 en caso contrario.

5.3. Variables auxiliares

A la hora de resolver el problema mediante programación lineal, hay que tener en cuenta la diferencia entre las variables binarias y las enteras. Para poder realizar los cálculos, se crean las siguientes variables auxiliares:

- $fr_k^\ell = 1$, si se activa la frecuencia, 0 en caso contrario.
- ψ_{ij}^ℓ, variable entera, sirve para linealizar el producto entre $u_{ij}^\ell \times fr^\ell$.
- $\rho_i^{\theta\ell\ell'}$, variable entera, sirve para linealizar el producto entre $w_i^{\theta\ell'} \times hw^\ell$.
- $\eta_i^{\theta\ell\ell'}$, variable entera, sirve para linealizar el producto entre $v_i^{\theta\ell\ell'} \times hw^\ell$.
- $\sigma_{ij}^{\theta\ell}$, variable entera, sirve para linealizar el producto entre $r_{ij}^{\theta\ell} \times hw^\ell$.

5.4. Función objetivo

La función objetivo tiene en cuenta el tiempo total del viaje, contando los tiempos de desplazamiento, de espera, de traslado y de transbordo, en caso de haberlo. Minimizar el tiempo corresponde al punto de vista del pasajero, pero como también se busca considerar el punto de vista del operario, se deben respetar las limitaciones de éste. Por lo tanto, habrá limitación en el número de líneas de bus, el número de nodos permitidos por línea y la longitud permitida de cada línea, teniendo en cuenta que cuanto mayor sea el número y la longitud de las líneas, mayor será el costo ocasionado para el operador del servicio y el número de autobuses disponibles.

Se define el tiempo total de viaje en autobús z_{bus} como la suma del tiempo de viaje considerando todos los pares OD que viajan por los arcos seleccionados de la red de autobús (Eq. 1).

$$z_{bus} = \frac{60}{v_{bus}} \cdot \sum_{\theta \epsilon \Theta} \sum_{\ell \epsilon \mathcal{L}} \sum_{(i,j) \epsilon A} r_{ij}^{\theta\ell} \cdot d_{ij} \cdot p_\theta \quad (1)$$

El tiempo de viaje a pie z_{walk} de todos los pares OD que utilizan el modo peatonal se define de forma similar (Eq 2).

$$z_{walk} = \frac{60}{v_{ped}} \cdot \sum_{\theta \epsilon \Theta} \sum_{\{i,j\} \epsilon E} t_{ij}^{\theta\ell} \cdot d_{ij} \cdot p_\theta \quad (2)$$

El tiempo de espera z_{wait} en la parada de origen de todos los pares, viene dado a través el producto $r_{ij}^{\theta\ell} \times hw^{\ell}$ representado por la variable $\sigma_{ij}^{\theta\ell}$ y el número de pasajeros del par (Eq 3).

$$z_{wait} = \sum_{\theta\in\Theta} \sum_{\substack{(i,j)\in A \\ i=\theta_o}} \sum_{\ell\in\mathcal{L}} p_\theta \cdot \frac{\sigma_{ij}^{\theta\ell}}{2} \quad (3)$$

El tiempo de espera transbordando z_{trans} entre la línea de autobús ℓ y la línea de autobús ℓ', se calcula con el producto del tiempo de espera medio (headway entre dos) y la demanda de todos los pares que pasan (Eq 4).

$$z_{trans} = \sum_{\theta\in\Theta} \sum_{\substack{i\neq\theta_o \\ i\neq\theta_d}} \sum_{\ell'\neq\ell} \sum_{\ell} p_\theta \cdot \frac{\eta_i^{\theta\ell\ell'}}{2} \quad (4)$$

El tiempo de cambio de modos z_{change} en la parada i de la línea ℓ se calcula a través de la variable $\rho_i^{\theta\ell\ell'}$ que denota el producto $w_i^{\theta\ell'} \times hw^{\ell}$, (tiempo de espera medio si hay cambios) y del número de pasajeros del par p_θ (Eq 5).

$$z_{change} = \sum_{\theta\in\Theta} \sum_{i\in\mathbb{N}_{ped}} \sum_{\ell\in\mathcal{L}} p_\theta \cdot \frac{\rho_i^{\theta\ell\ell'}}{2} \quad (5)$$

5.4.1. Linealización de la variable auxiliar $\sigma_{ij}^{\theta\ell}$

La variable $\sigma_{ij}^{\theta\ell} = r_{ij}^{\theta\ell} \times hw^{\ell}$ que aparece en la (Eq 3) es el producto de una variable binaria y otra entera, lo cual hace que el modelo sea no lineal. Para evitar esta no linealidad, se introduce un conjunto de nuevas restricciones al modelo, que representa de forma lineal el producto de dichas variables:

$$\sigma_{ij}^{\theta\ell} \leq hw^{\ell}, \quad (i,j) \in A, \ell \in \mathcal{L}, \forall\theta \quad (6)$$

$$\sigma_{ij}^{\theta\ell} \leq \frac{60}{\mathcal{F}min} \cdot r_{ij}^{\theta\ell}, \quad (i,j) \in A, \ell \in \mathcal{L}, \forall\theta \quad (7)$$

$$\sigma_{ij}^{\theta\ell} \geq hw^{\ell} - \frac{60}{\mathcal{F}min} \cdot \left(1 - r_{ij}^{\theta\ell}\right), \quad (i,j) \in A, \ell \in \mathcal{L}, \forall\theta \quad (8)$$

5.4.2. Linealización de la variable auxiliar $\eta_i^{\theta\ell\ell'}$

La linealización del parámetro $\eta_i^{\theta\ell\ell'} = v_i^{\theta\ell} \times hw^{\ell}$ se realiza de forma similar.

$$\eta_i^{\theta\ell\ell\prime} \leq hw^{\ell}, \quad \forall\ell,\ell',\ell\neq\ell', i\in\mathbb{N}, i\neq\theta_o, i\neq\theta_d, \forall\theta \quad (9)$$

$$\eta_i^{\theta\ell\ell'} \leq \frac{60}{\mathcal{F}^{min}} \cdot v_i^{\theta\ell\ell'}, \quad \forall \ell, \ell', \ell \neq \ell', , i \in \mathbb{N}, i \neq \theta_o, i \neq \theta_d, \forall \theta \tag{10}$$

$$\eta_i^{\theta\ell\ell'} \geq hw^{\ell} - \frac{60}{\mathcal{F}^{min}} \cdot \left(1 - v_i^{\theta\ell\ell'}\right), \quad \forall \ell, \ell', \ell \neq \ell', \quad i \in \mathbb{N}, i \neq \theta_o, i \neq \theta_d, \forall \theta \tag{11}$$

5.4.3. Linealización de la variable auxiliar $\rho_i^{\theta\ell\ell'}$

La linealización del parámetro $\rho_i^{\theta\ell} = w_i^{\theta} \times hdw^{\ell}$ cumple lo siguiente:

$$\rho_i^{\theta\ell\ell'} \leq hw^{\ell}, \quad \ell \neq \ell' \tag{12}$$

$$\rho_i^{\theta\ell\ell'} \leq \frac{60}{\mathcal{F}^{min}} \cdot z_k^{\theta\ell\ell'}, \quad \ell \neq \ell' \tag{13}$$

$$\rho_i^{\theta\ell\ell'} \geq hw^{\ell} - \frac{60}{\mathcal{F}^{min}} \cdot \left(1 - z_k^{\theta\ell\ell'}\right), \quad \ell \neq \ell' \tag{14}$$

5.5. Restricciones del diseño de red

Se selecciona un arco A que forma parte de la línea ℓ solo si ya se han seleccionado previamente las paradas adyacentes (Eq 15, Eq 16).

$$u_{ij}^{\ell} \leq s_i^{\ell}, \quad (i,j) \in A, \quad \ell \in \mathcal{L} \tag{15}$$

$$u_{ij}^{\ell} < s_j^{\ell}, \quad (i,j) \in A, \quad \ell \in \mathcal{L} \tag{16}$$

Solo se activa la parada i de la línea ℓ si existe un arco activo entrando o saliendo en el nodo i de la línea ℓ (Eq 17, Eq 18).

$$s_j^{\ell} = \sum_{(i,j) \in A} u_{ij}^{\ell}, \quad j \in S, \quad \ell \in \mathcal{L} \tag{17}$$

$$s_j^{\ell} = \sum_{(j,i) \in A} u_{ji}^{\ell}, \quad j \in S, \quad \ell \in \mathcal{L} \tag{18}$$

Las líneas deben ser circulares (Eq 19). Cada línea tiene al menos un mínimo de paradas S_{min} y un máximo S_{max} siempre que se cree la línea de autobús (Eq 20). Cada línea debe tener una longitud máxima (Eq 21).

$$\sum_{(i,j) \in A} u_{ij}^{\ell} = \sum_{i \in S} s_i^{\ell}, \ \ell \in \mathcal{L} \tag{19}$$

$$m^{\ell} \cdot S_{min} \leq \sum_{i \in S} s_i^{\ell} \leq S_{max} \cdot m^{\ell}, \quad \ell \in \mathcal{L} \tag{20}$$

$$\sum_{(i,j)\in A} u_{ij}^{\ell} \cdot d_{ij} \leq \mathcal{M},\ \ \ell \in \mathcal{L} \tag{21}$$

Solo se puede transbordar en el nodo k de la línea $\ell \in \mathcal{L}$ a la línea $\ell' \in \mathcal{L}$ si el nodo k se ha activado para ambas líneas (Eq 22, Eq 23).

$$v_k^{\theta\ell\ell'} \leq s_k^{\ell},\ \ \ell \neq \ell' \tag{22}$$

$$v_k^{\theta\ell\ell'} \leq s_k^{\ell'},\ \ \ell' \neq \ell \tag{23}$$

La línea ℓ se activa si existe un arco $(i,j) \in A$ activo para esa línea en la capa de autobús.

$$m^{\ell} \leq \sum_{(i,j)\in A} u_{ij}^{\ell}, \ell \in \mathcal{L} \tag{24}$$

$$v_i^{w\ell\ell'} \leq 1/2 \cdot (m^{\ell} \cdot m^{\ell'}) \tag{25}$$

5.6. Restricciones de transbordo

La siguiente restricción obliga a que solo pueda hacerse transbordo en la parada k de la línea $\ell \in \mathcal{L}$ a la línea $\ell' \in \mathcal{L}$ si la variable $v_k^{\theta\ell\ell'}$ se activa para ese par θ (Eq 26).

$$\sum_{(i,k)\in A} r_{ik}^{\theta\ell} + \sum_{l'\neq l} \sum_{(k,i)\in A} r_{kj}^{\theta\ell'} \leq 1 + v_k^{\theta\ell\ell'},\ k \in S,\ \ \ell \in \mathcal{L},\ \ \theta \in \Theta \tag{26}$$

Si un par OD θ entra en la parada $k\ \epsilon\ S$ usando una arista peatonal y sale de k usando un arco de bus, se realiza un cambio de modos en esa parada k (Eq 27).

$$\sum_{k\neq i} \sum_{(i,k)\in A} t_{ik}^{\theta} + \sum_{l\in L} \sum_{(k,i)\in A} r_{kj}^{\theta\ell} \leq 1 + w_k^{\theta\ell'},\ k \in S,\ \ \theta \in \Theta \tag{27}$$

5.7. Restricción de asignación

Para cada par OD θ, si la línea no tiene seleccionado el arco (i,j), el flujo de pasajeros que atraviesa ese arco no puede activarse:

$$r_{ij}^{\theta\ell} \leq u_{ij}^{\ell},\ \ \ell \in \mathcal{L},\ \ \theta \in \Theta,\ \ (\mathrm{i,j}) \in \mathrm{A} \tag{28}$$

5.8. Restricciones de conservación de flujo

El siguiente conjunto de restricciones representan la conservación de flujo. (Eq 29) impone que, del origen de cada par, se tenga que ir o bien por un arco de autobús o por una arista pedestre. Similarmente, (Eq 30) obliga a entrar en el destino o bien en autobús

o bien andando. (Eq 31) representa el balance en los nodos intermedios que no son ni origen ni destino.

$$\sum_{\ell \in L} \sum_{(\theta^o,k)\in A} r_{\theta^o k}^{\theta \ell} + \sum_{\{\theta^o,k\}\in E} t_{\theta^o k}^{\theta} = 1\,,\ \theta = (\theta^o,\theta^d) \in \Theta \tag{29}$$

$$\sum_{\ell \in L} \sum_{(i,\theta^o)\in A} r_{i\theta^d}^{\theta \ell} + \sum_{\{i,\theta^d\}\in E} t_{i\theta^d}^{\theta} = 1,\ \ \theta = (\theta^o,\theta^d) \in \Theta \tag{30}$$

$$\sum_{(i,k)\in A} r_{ik}^{\theta \ell} + \sum_{\{i,k\}\in E} t_{ik}^{\theta} = \sum_{(k,i)\in A} r_{kj}^{\theta \ell} + \sum_{\ell \neq \ell'} \sum_{(k,j)\in A} r_{kj}^{\theta \ell'} + \sum_{\{k,j\}\in E} t_{kj}^{\theta} \tag{31}$$

5.9. Restricciones de frecuencia y headway

La frecuencia es el número de veces que un autobús de la línea ℓ pasa por una determinada parada en un tiempo establecido (que suponemos 60 minutos). La frecuencia de la línea ℓ fr^{ℓ} se define como una combinación lineal convexa de variables binarias fr_k^{ℓ} (Eq 32). El headway es el tiempo que pasa entre un autobús y el siguiente. Si la línea se activa, el producto headway y frecuencia es igual a 60 (Eq 33, Eq 34).

$$\sum_{k=1}^{|F|} fr_k^{\ell} = 1,\ fr^{\ell} \in \mathbb{Z},\ \ fr_k^{\ell} \mathbb{B} \tag{32}$$

$$\sum_{k=1}^{|F|} f_k \cdot fr_k^{\ell} = fr^{\ell} \tag{33}$$

$$\sum_{k=1}^{|F|} f_k \cdot \varepsilon_k^{\ell} = 60 \cdot m^{\ell},\ \varepsilon_k^{\ell} \in \mathbb{Z} \tag{34}$$

5.9.1. Linealización de la variable auxiliar ε_k^{ℓ}

La variable auxiliar $\varepsilon_k^{\ell} = hw^{\ell} \cdot fr_k^{\ell}$ necesita ser linealizada puesto que el producto de dos variables es no lineal (Eq 35, Eq 36, Eq 37):

$$\varepsilon_k^{\ell} \leq hw^{\ell},\ \ \forall \ell, k = 1..|F| \tag{35}$$

$$\varepsilon_k^{\ell} \leq \frac{60}{f_k} \cdot fr_k^{\ell},\ \ \forall \ell, k = 1..|F| \tag{36}$$

$$\varepsilon_k^\ell \geq hw^\ell - \frac{60}{f_k} \cdot \left(1 - fr_k^\ell\right), \quad \forall \ell, k = 1..|F| \tag{37}$$

5.10. Restricciones de flota

La flota requerida por todas las líneas de autobús debe ser menos que la flota máxima de la que se dispone (Eq 38). La flota debe ser suficiente para cubrir las líneas de autobús (Eq 39, Eq 40).

$$\sum_{\ell=1}^{|\mathcal{L}|} z^\ell \leq fl^{max} \tag{38}$$

$$z^\ell \geq \sum_{ij} \frac{d_{ij}}{v_{bus}} \cdot \psi_{ij}^\ell \tag{39}$$

$$z^\ell \geq 1 + \sum_{ij} \frac{d_{ij}}{v_{bus}} \cdot \psi_{ij}^\ell \tag{40}$$

5.10.1. Linealización de la variable auxiliar ψ_{ij}^ℓ

La linealización del parámetro auxiliar $\psi_{ij}^\ell = u_{ij}^\ell \cdot fr^\ell$ se obtiene de la siguiente manera:

$$\psi_{ij}^\ell \leq fr^\ell, \quad (i,j) \in A, \forall \ell \tag{41}$$

$$\psi_{ij}^\ell \leq \mathcal{F}^{max} \cdot u_{ij}^\ell, \quad (i,j) \in A, \forall \ell \tag{42}$$

$$\psi_{ij}^\ell \geq fr^\ell - \mathcal{F}^{max} \cdot \left(1 - u_{ij}^\ell\right), \quad (i,j) \in A, \forall \ell \tag{43}$$

5.11. Restricciones de capacidad

El número total de pasajeros de la línea ℓ por hora no debe ser menor que la capacidad de la línea (Eq 44). Si la línea se activa, solo puede asignarse un tipo de autobús (Eq 45).

$$\sum_{\Theta} p_\theta \cdot r_{ij}^{\theta\ell} \leq fr^\ell \cdot (\mathcal{C}_1 \cdot \delta_1^\ell + \mathcal{C}_2 \cdot \delta_2^\ell), \quad \forall (i,j), \quad \forall \ell \tag{44}$$

$$\delta_1^\ell + \delta_2^\ell \leq 1, \forall \ell \tag{45}$$

La suma de todos los autobuses de un tipo (simple o doble) deben ser menor que el máximo de la flota de su mismo tipo (Eq 46, Eq 47). La flota total, será entonces, la suma de la flota de ambos tipos (Eq 48).

$$\sum_{\ell=1}^{|\mathcal{L}|} z_1^{\ell} \leq fl_1^{max} \tag{46}$$

$$\sum_{\ell=1}^{|\mathcal{L}|} z_2^{\ell} \leq fl_2^{max} \tag{47}$$

$$z^{\ell} = z_1^{\ell} + z_2^{\ell} \tag{48}$$

Para cada hora, la flota de autobuses de la línea de un tipo debe ser menor que el máximo de la flota del mismo tipo (Eq 49, Eq 50).

$$z_1^{\ell} \leq fl_1^{max} \cdot \delta_1^{\ell} \tag{49}$$

$$z_2^{\ell} \leq fl_2^{max} \cdot \delta_2^{\ell} \tag{50}$$

5.12. Formulación matemática del modelo

Teniendo en cuenta las variables y las restricciones explicadas en los apartados anteriores, el modelo final sería:

$$\begin{aligned} Minimizar \quad & \xi_{bus} \cdot z_{bus} + \xi_{walk} \cdot z_{walk} + \xi_{trans} \cdot z_{trans} + \xi_{wait} \cdot z_{wait} \\ & + \xi_{change} \cdot z_{change} \\ s.a: \quad & \\ & Diseño\ de\ red: (15) — (25) \\ & Transbordo: (26) - -(27) \\ & Asiganción: (28) \\ & Conservación\ de\ flujo: (29) - -(31) \\ & Frecuencia\ y\ headway: (32) - -(37) \\ & Flota: (38) - -(43) \\ & Capacidad: (44) - -(50) \end{aligned}$$

Como la función objetivo busca minimizar el tiempo de viaje, el modelo tenderá a asignar a los pasajeros por el camino más rápido siempre que las restricciones de

capacidad lo permitan. Para los pares OD que solo usan el método peatonal se considera demanda no atendida.

6. Implementación en Gurobi-Python

Los experimentos computacionales llevados a cabo permiten evaluar la veracidad del modelo planteado. El modelo se implementa con el lenguaje de Python y se utilizada Gurobi para obtener mejores resultados.

1. Primero se importan las librerías y datos externos que van a usarse a lo largo del código.

```python
import gurobipy as gp
from gurobipy import GRB
from gurobipy import *
from inputs_data import *
from preprocesamiento import *
```

2. Se definen las variables de entrada y los ficheros de los que se extraen la red de prueba.

```python
user_variables = {
    'NumLineas': 3,
    'MinLen': 1,
    'MaxLen': 150,
    'S_min': 1,
    'S_max': 14,
    'preprocesamiento': True,
    'stop_file_name':"Stops_Mandl.txt",
    'stopPED_file_name': "Stops_PED_Mandl.txt",
    'arcBus_file_name': "Arcs_bus_Mandl_reescalado.txt",
    'arcPED_file_name': "Arcs_PED_Mandl_reescalado.txt",
    'OD_name': "OD_Mandl.txt",
    'coordenadas_name': "Coordenadas.txt",
    'headway_list': [5, 3],
    'frequecies': [12, 20],
    'fleet_max': 6,
    'v_bus': 20,
    'v_PED': 3,
    'M': 500,
    'cap_1': 110,
    'cap_2': 81,
    'fleet_1_max': 3,
    'fleet_2_max': 3,
}
```

3. Se crean los diccionarios para el uso del preprocesamiento, donde se filtran los datos para escoger los caminos más cortos para crear las líneas de autobús.

```
# -----------------     Creation of dictionaries     --------------------
# ----------------------------------------------------------------------
preprocessing=user_variables['preprocesamiento']
# lista de arcos con las distancias; lista con arco y distancias
Lista_arcos_distancias_bus = Lectura_ArcsBusFile(pathData, arcBus_file_name, 2)
if preprocessing:
    # --diccionario tipo; parOD: lista cuyos eltos son listas con los nodos de cada path:
    dicc_parOD_listaPaths = CreaDiccionario_KshortestPaths(Lista_nodos, Lista_arcos_distancias_bus, Lista_OD)

    # --diccionario tipo; parOD: lista con todos los nodos implicados en todos los paths:
    dicc_parOD_listaNodos = CreaDiccionario_listaNodos_KshortestPaths(dicc_parOD_listaPaths)

    # --diccionario tipo; parOD: lista con todos los arcos implicados en todos los paths:
    dicc_parOD_listaArcos = CreaDiccionario_listaArcos_KshortestPaths(dicc_parOD_listaPaths)

else: # -----------------     Creation of dictionaries without PREPROCESSING    --------------------
    # --diccionario tipo; parOD: lista con todos los nodos del grafo bus:
    dicc_parOD_listaNodos=CreaDiccionario_listaNodos(Lista_nodos,Lista_OD)

    # --diccionario tipo; parOD: lista con todos los arcos del grafo bus:
    dicc_parOD_listaArcos=CreaDiccionario_listaArcos(Lista_arcos_bus, Lista_OD)

# --diccionario tipo; parOD: lista con todos las aristas PED:
dicc_parOD_listaAristasPED=CreaDiccionario_listaArcos(Lista_arcs_PED, Lista_OD)

print("dicc_parOD_listaArcos", dicc_parOD_listaArcos)
```

3. Se declaran las variables.

```
#-- desicion variables:
hw = {}#headway of each line
fr = {}#frequency of each line
z = {}#required fleet
s = {}#stops of each line
u = {}#arcs of each line
r = {}#flow
t = {}#pedestrian arcs
v = {}#transfer between lines
w = {}#change from pedestrian to bus
delta_1={} # selección del tipo 1 de bus
delta_2={} # selección del tipo 2 de bus
z_1={} #nb buses tipo 1
z_2={}#nb buses tipo 2
h={}#variable representando si la linea se activa o no

#--  auxiliary variables:
fr_k = {}#para definir headway como una c.l. de binarias
#eps_k = {}#para linealizar el producto hw y fr_k
psi={}#variable representando el producto de u y fr
eta={}#variable representando el producto de v y h
sigma={}#variable representando el producto de r y h
rho={}#variable representado el producto w and h
```

4. Se crean las variables de diseño del modelo.

```
#-------- DESIGN:

NumLines = user_variables['NumLineas'] #donde se recogen los datos de entrada
Lines = list(range(1, NumLines + 1))

# ------Indicador de activacion de lineas
for l in Lines:
    h[l] = mod.addVar(lb=0,ub=1.0, vtype=GRB.BINARY, name="h_%s" % (l))
mod.update()


# s_i^l = 1 si la parada i se activa para la linea l
for i in Lista_nodos:
    for l in Lines:
        s[i, l] = mod.addVar(lb=0, ub=1.0, vtype=GRB.BINARY, name="s_%s_%s" % (i,l))
mod.update()

# u_ij^l =1, si el arco (i,j) se activa en l
for (i, j) in Lista_arcos_bus:
     for l in Lines:
         u[i, j, l] = mod.addVar(lb=0, ub=1.0, vtype=GRB.BINARY, name="u_%s_%s_%s" % (i, j, l))
mod.update()
```

5. Se crean las variables de conservación de flujo del modelo.

```
#-------- FLOW:
# r_ij^Ol =1, si el flujo del par O pasa por (i,j) de la linea l
for (origen, destino) in dicc_parOD_listaArcos: #dicc tipo; parOD: listaArcos
    for (i,j) in dicc_parOD_listaArcos[(origen,destino)]:
        for l in Lines:
            r[(i, j), (origen, destino), l] = mod.addVar(lb=0, ub=1.0, vtype=GRB.BINARY,
                                    name="r_(%s,%s)_(%s,%s)_%s" % (i, j, origen, destino, l))

mod.update()

# t_ij^O =1, si el flujo del par O pasa por arco pedestre (i,j)
for (origen, destino) in dicc_parOD_listaAristasPED: #dicc tipo; parOD: listaArcos
    for (i,j) in dicc_parOD_listaAristasPED[(origen,destino)]:
        t[(i, j), (origen, destino)] = mod.addVar(lb=0, ub=1.0, vtype=GRB.BINARY,
                                    name="t_(%s,%s)_(%s,%s)" % (i, j, origen, destino))

mod.update()
```

```
# v_i^Oll' =1, si los pasajeros del par O transfieren de la linea l a la l' en parada i
for (origen, destino) in dicc_parOD_listaNodos: #dicc tipo; parOD: listaNodos
    for i in dicc_parOD_listaNodos[(origen,destino)]: #cogemos un nodo de la lista
        if i != origen and i != destino: #seleccionamos los que no son ni origen ni destino
            for l1 in Lines:
                for l2 in Lines:
                    if l1 != l2:
                        v[i, (origen, destino), l1, l2] = mod.addVar(lb=0, ub=1.0, vtype=GRB.BINARY,
                                    name="v_%s_(%s,%s)_%s_%s" % (i, origen, destino, l1, l2))

mod.update()

# w_i^Ol =1, si los pasajeros del par O cambian del modo pedestre al modo bus en el nodo i de la linea l
for (origen, destino) in dicc_parOD_listaNodos: #dicc tipo; parOD: listaNodos
    for i in dicc_parOD_listaNodos[(origen,destino)]: #cogemos un nodo de la lista de bus
        if Lista_nodos_PED.__contains__(i): #siempre que i esté en los pedestres
            for l in Lines:
                w[i, (origen, destino), l] = mod.addVar(lb=0, ub=1.0, vtype=GRB.BINARY,
                                    name="w_%s_(%s,%s)_%s" % (i, origen, destino, l))

mod.update()
```

6. Se crean las variables de frecuencia y headway del modelo.

```
#-------- FREQUENCIES AND HEADWAYS AND TYPE OF BUS:
# h^l headway line l
for l in Lines:
    hw[l] = mod.addVar(lb=0, vtype=GRB.INTEGER, name="hw_%s" % (l))
mod.update()

# fr^l frequency line l
for l in Lines:
    fr[l] = mod.addVar(lb=0, vtype=GRB.INTEGER, name="fr_%s" % (l))
mod.update()

# DELTA_1^l tipo 1 de bus, line l
for l in Lines:
    delta_1[l] = mod.addVar(lb=0, vtype=GRB.BINARY, name="delta_1_%s" % (l))
mod.update()

# DELTA_2^l tipo 2 de bus, line l
for l in Lines:
    delta_2[l] = mod.addVar(lb=0, vtype=GRB.BINARY, name="delta_2_%s" % (l))
mod.update()

#-- auxiliary variables for headway and frequency:
# fr_k^l frequency_k for line l
print(list(range(1,len(frequencies_list)+1)))
for k in range(1,len(frequencies_list)+1):
    for l in Lines:
        fr_k[k,l]=mod.addVar(lb=0, ub=1.0, vtype=GRB.BINARY, name="fr_k_%s_%s" % (k,l))
mod.update()
```

7. Se crean las variables de la flota del modelo.

```
#-------- FLEET:
# z^l required fleet
for l in Lines:
    z[l] = mod.addVar(lb=0, vtype=GRB.INTEGER, name="z_%s" % (l))
mod.update()

for l in Lines:
    z_1[l] = mod.addVar(lb=0, vtype=GRB.INTEGER, name="z1_%s" % (l))
mod.update()

for l in Lines:
    z_2[l] = mod.addVar(lb=0, vtype=GRB.INTEGER, name="z2_%s" % (l))
mod.update()

#-- auxiliary variables for fleet
for (i, j) in Lista_arcos_bus:
    for l in Lines:
        psi[i, j, l] = mod.addVar(lb=0, vtype=GRB.INTEGER, name="psi_%s_%s_%s" % (i, j, l))
mod.update()

```

```
#-- auxiliary variables for product headway and transfers:
for (origen, destino) in dicc_parOD_listaNodos:
    for i in dicc_parOD_listaNodos[(origen,destino)]:
        if i != origen and i != destino:
            for l1 in Lines:
                for l2 in Lines:
                    if l1 != l2:
                        eta[i,(origen, destino), l1, l2] = mod.addVar(lb=0, vtype=GRB.INTEGER,
                                    name="eta_%s_(%s,%s)_%s_%s" % (i, origen,destino, l1,l2))
mod.update()

#-- auxiliary variables for product headway and flow:
for (origen, destino) in dicc_parOD_listaArcos: #dicc tipo; parOD: listaArcos
    for (i,j) in dicc_parOD_listaArcos[(origen,destino)]:
        for l in Lines:
            sigma[(i, j), (origen, destino), l] = mod.addVar(lb=0, vtype=GRB.INTEGER,
                        name="rho_(%s,%s)_(%s_%s)_%s" % (i, j, origen, destino, l))
mod.update()

#-- auxiliary variables for product w and h:
for (origen, destino) in dicc_parOD_listaNodos: #dicc tipo; parOD: listaNodos
    for i in dicc_parOD_listaNodos[(origen,destino)]:
        if Lista_nodos_PED.__contains__(i):# siempre que i esté en los pedestres
            for l in Lines:
                rho[i, (origen, destino), l] =mod.addVar(lb=0, vtype=GRB.INTEGER,
                            name="rho_%s_(%s,%s)_%s" % (i, origen, destino, l))
```

8. Se crean las restricciones de frecuencia y headway.

```
# ---------   Constraint: HEADWAYS AND FREQUENCIES:
# ---------   Constraint linearization of products fr_l * h^l    ------------------------

#-------- sum_k fr_k^l=1:
for l in Lines:
    mod.addConstr(quicksum(fr_k[k,l] for k in range(1,len(frequencies_list)+1)) == h[l],
                  name="sum_fr_k_%s" % (l))
mod.update()

#-------- sum_k *freq_k * fr_k^l=fr^i:
for l in Lines:
    mod.addConstr(quicksum(fr_k[k,l]* frequencies_list[k-1] for k in range(1,len(frequencies_list)+1)) == fr[l],
                  name="sum_fr_freq_%s" % (l))
mod.update()

#-------- sum_k *60/freq_k * fr_k^l=hw^l:
for l in Lines:
    mod.addConstr(quicksum(fr_k[k,l] * 60/frequencies_list[k - 1] for k in range(1,
                           len(frequencies_list) + 1)) == hw[l], name="sum_hw_freq_%s" % (l))
mod.update()
```

9. Se crean las restricciones de flota

```
# ---------   Constraint: FLEET
mod.addConstr(quicksum(z_1[l]  for l in Lines) <= fleet_1_max, name="sum_z1_%s<upper")
mod.update()
mod.addConstr(quicksum(z_2[l]  for l in Lines) <= fleet_2_max, name="sum_z2_%s<upper")
mod.update()
mod.addConstr(quicksum(z[l]  for l in Lines) <= fleet_max, name="sum_z_%s<upper")
mod.update()
for l in Lines:
    mod.addConstr(z_1[l]  <= fleet_1_max*delta_1[l], name="z1_%s<upper")
    mod.update()
    mod.addConstr(z_2[l]  <= fleet_2_max*delta_2[l], name="z2_%s<upper")
    mod.update()
    mod.addConstr(z[l] == z_1[l] +z_2[l] , name="z_equal z1_z2_")
    mod.update()
```

```
#-------- z^l<= 1+ sum_ij (d_ij/v_bus * psi_ij^l):
for l in Lines:
    mod.addConstr(z[l] <= 1+quicksum(float(dicc_arco_distancia[(i,j)])/v_bus*psi[i,j,l]
                            for (i,j) in dicc_arco_distancia), name="z_rigth_hand_%s" % ( l))
mod.update()

#-------- psi_ij^l <= fr^l:

for (i,j) in Lista_arcos_bus:
    for l in Lines:
        mod.addConstr(psi[i,j,l] <= fr[l], name="psi_fr_%s_%s_%s" % (i,j, l))
mod.update()

#-------- psi_ij^l <= freq_max* u_ij^l:

for (i,j) in Lista_arcos_bus:
    for l in Lines:
        mod.addConstr(psi[i,j,l] <= freq_max* u[i,j,l], name="psi_u_%s_%s_%s" % (i,j, l))
mod.update()
#-------- psi_ij^l >= fr^l-freq_max* (1-u_ij^l):

for (i,j) in Lista_arcos_bus:
    for l in Lines:
        mod.addConstr(psi[i,j,l] >= fr[l]-freq_max* (1-u[i,j,l]), name="psi_fr_u_%s_%s_%s" % (i,j, l))
mod.update()
```

10. Se crean las restricciones de capacidad.

```
# ---------   Constraint: Capacity constraints
# Pasajeros viajando en arco ij de línea l #dicc tipo; parOD: listaArcos
aux=0
for l in Lines:
    for (i, j) in Lista_arcos_bus:
        for (origen, destino) in dicc_parOD_listaArcos:  # dicc tipo; parOD: listaArcos
            for (k, m) in dicc_parOD_listaArcos[(origen, destino)]:
                    if i == k and j==m:
                        aux=aux+r[(i, j), (origen, destino), l] * float(dicc_OD_demanda[(origen, destino)])
        mod.addConstr(aux <= fr[l] * (cap_1 * delta_1[l] + cap_2 * delta_2[l]),
                      name = "Restricapacidad(%s,%s)_%s" % (i, j, l))
        aux=0
mod.update()

#--------  delta_1+delta_2<1
for l in Lines:
    mod.addConstr(delta_1[l]+delta_2[l]<=1, name="delta%s" % (l))
mod.update()
```

11. Se crean las restricciones de diseño de red.

```
# ---------   Constraint: NETWORK DESIGN

#-------- u_ij^l <= s_i^l:
for (i,j) in Lista_arcos_bus:
    for l in Lines:
        mod.addConstr(u[i,j,l] <= s[i,l], name="u_s_i%s_%s_%s" % (i,j, l))
mod.update()

#-------- u_ij^l <= s_j^l:
for (i,j) in Lista_arcos_bus:
    for l in Lines:
        mod.addConstr(u[i,j,l] <= s[j,l], name="u_s_j%s_%s_%s" % (i,j, l))
mod.update()

#-------- s_j^l =sum_{ij}{u_ij^l}:
for j in Lista_nodos:
    for l in Lines:
        mod.addConstr(s[j,l] ==quicksum(u[i,k,l] for (i,k) in Lista_arcos_bus if k == j),
                      name="s_j_sum_u_ij%s_%s" % (j, l))
mod.update()

#-------- s_j^l =sum_{ji}{u_ji^l}:
for j in Lista_nodos:
    for l in Lines:
        mod.addConstr(s[j,l] ==quicksum(u[k,i,l] for (k,i) in Lista_arcos_bus if k == j),
                      name="s_j_sum_u_ji%s_%s" % (j, l))
mod.update()
```

```
#-------- sum_ij{u_ij^l} =sum_{i}{s_i^l}:
for l in Lines:
    mod.addConstr(quicksum(u[i,j,l] for (i,j) in Lista_arcos_bus) == quicksum(s[i,l]
                                    for i in Lista_nodos), name="s_j_sum_u_ji%s_" % ( l))
mod.update()

#-------- S_min <= sum_{i}{s_i^l}:
for l in Lines:
    mod.addConstr(S_min *h[l]<= quicksum(s[i,l] for i in Lista_nodos), name="S_min_sum_s_i_l%s_" % ( l))
mod.update()

#-------- sum_{i}{s_i^l}<= S_max:
for l in Lines:
    mod.addConstr(quicksum(s[i,l] for i in Lista_nodos)<=S_max*h[l], name="sum_s_i_l_S_max%s_" % ( l))
mod.update()

#-------- sum_{ij}{u_ij^l * dis}<= M:
for l in Lines:
    mod.addConstr(quicksum(u[i,j,l]*dis  for (i,j,dis) in Lista_arcos_distancias_bus)<=M,
                  name="sum_s_i_l_S_max%s_" % ( l))
mod.update()
```

```
#-------- r_ij0^l <= u_ijl:
for (origen, destino) in dicc_par0D_listaArcos: #dicc tipo; par0D: listaArcos
    for (i,j) in dicc_par0D_listaArcos[(origen,destino)]:
        for l in Lines:
            mod.addConstr(r[(i, j), (origen, destino), l] <=u[i,j,l],
                          name="r_ij0^l <= u_ijl(%s,%s)_(%s,%s)_%s" % (i,j,origen, destino, l))
mod.update()

for (origen, destino) in dicc_par0D_listaArcos: #dicc tipo; par0D: listaArcos
    for l1 in Lines:
        for l2 in Lines:
            if l1 != l2:
                for k in dicc_par0D_listaNodos[(origen,destino)]:
                    if k != destino and k != origen:
                        mod.addConstr(v[k, (origen, destino), l1, l2] <= 0.5*(h[l1]+h[l2]),
                                      name="nodo transfer_%s_(%s,%s)_%s_%s" % (k,origen, destino, l1,l2))
mod.update()

for l in Lines:
    mod.addConstr(quicksum(u[i,j,l] for (i,j) in Lista_arcos_bus)>=h[l], name="u_con h %s_" % ( l))
mod.update()
```

12. Se crean las restricciones de conservación de flujo.

```
# ---------   Constraint: FLOW CONSERVATION

# Flujo_Saliendo_Origen
for (origen, destino) in dicc_parOD_listaArcos: #dicc tipo; parOD: listaArcos
    mod.addConstr(quicksum(quicksum(r[(i, k), (origen, destino), l]
                for (i, k) in dicc_parOD_listaArcos[(origen,destino)] if i == origen) for l in Lines)
                  + quicksum(t[(i,k),(origen, destino)]
                             for (i,k) in dicc_parOD_listaAristasPED[(origen,destino)] if i == origen) == 1,
                  name="Flujo_Saliendo_Origen(%s,%s)_(%s,%s)_%s" % (i,j,origen, destino, l))
mod.update()

# Flujo_Entrando_Destino
for (origen, destino) in dicc_parOD_listaArcos: #dicc tipo; parOD: listaArcos
    mod.addConstr(quicksum(quicksum(r[(i, k), (origen, destino), l]
                    for (i, k) in dicc_parOD_listaArcos[(origen,destino)] if k == destino) for l in Lines)
                  + quicksum(t[(i,k),(origen, destino)]
                             for (i,k) in dicc_parOD_listaAristasPED[(origen,destino)] if k == destino) == 1,
                  name="Flujo_Entrando_Destino(%s,%s)_(%s,%s)_%s" % (i,j,origen, destino, l))
mod.update()
```

```
#--flow balance1
for (origen, destino) in dicc_parOD_listaArcos: #dicc tipo; parOD: listaArcos
    for l in Lines:
        for k in dicc_parOD_listaNodos[(origen,destino)]:#lista de aristas ped siempre debe tener una clave para cada OD
            if k != destino and k != origen:
                mod.addConstr(quicksum(r[(i, j), (origen, destino), l]
                                       for (i, j) in dicc_parOD_listaArcos[(origen,destino)] if j == k)
                    + quicksum(t[(i,j),(origen, destino)]
                               for (i,j) in dicc_parOD_listaAristasPED[(origen,destino)] if j == k)
                               ==
                    quicksum(r[(i, j), (origen, destino), l]
                             for (i, j) in dicc_parOD_listaArcos[(origen, destino)] if i == k)
                             +
                        quicksum(quicksum(r[(i, j), (origen, destino), l2]
                          for ( i, j) in dicc_parOD_listaArcos[(origen, destino)] if i == k)
                                for l2 in Lines if l2 != l)
                    + quicksum(t[(i, j), (origen, destino)]
                               for (i, j) in dicc_parOD_listaAristasPED[(origen, destino)] if i == k)
                     ,name="Flujo_Balance1(%s,%s)_(%s,%s)_%s" % (i,j,origen, destino, l))
mod.update()
```

13. Se crean las restricciones de transferencia.

```
# ---------   Constraint: TRANSFER
# transfers between lines:
for (origen, destino) in dicc_parOD_listaArcos: #dicc tipo; parOD: listaArcos
    for l1 in Lines:
        for l2 in Lines:
            if l1 != l2:
                for k in dicc_parOD_listaNodos[(origen,destino)]:
                    if k != destino and k != origen:
                        mod.addConstr(quicksum(r[(i, j), (origen, destino), l1]
                                               for (i, j) in dicc_parOD_listaArcos[(origen,destino)] if j == k)
                                      +quicksum(r[(i, j), (origen, destino), l2]
                                                for ( i, j) in dicc_parOD_listaArcos[(origen, destino)] if i == k) <=
                                                v[k, (origen, destino), l1, l2],
                                                name="Transfer_between_lines_%s_(%s,%s)_%s_%s" % (k,origen, destino, l1,l2))
mod.update()
```

```
# change of modes:
for (origen, destino) in dicc_parOD_listaArcos: #dicc tipo; parOD: listaArcos
    for l in Lines:
        for k in dicc_parOD_listaNodos[(origen,destino)]:
            if Lista_nodos_PED.__contains__(k):
                mod.addConstr(quicksum(t[(i,j),(origen, destino)]
                                       for (i,j) in dicc_parOD_listaAristasPED[(origen,destino)] if j == k)+
                              quicksum(r[(i, j), (origen, destino), l]
                                       for (i, j) in dicc_parOD_listaArcos[(origen, destino)]
                                   if i == k)
                              <=1 + w[k, (origen, destino), l],
                              name="Change_of_modes%s_(%s,%s)_%s" % (k,origen, destino, l))
mod.update()

# v_k debe ser nodo de interseccion_1:
for (origen, destino) in dicc_parOD_listaArcos: #dicc tipo; parOD: listaArcos
    for l1 in Lines:
        for l2 in Lines:
            if l1 != l2:
                for k in dicc_parOD_listaNodos[(origen,destino)]:
                    if k != origen and k != destino:
                        mod.addConstr( v[k, (origen, destino), l1, l2] <=s[k,l1],
                                       name="v_k_nodoInterseccion_1%s_(%s,%s)_%s_%s" % (k,origen, destino, l1,l2))
mod.update()
```

```
# v_k debe ser nodo de interseccion_2:
for (origen, destino) in dicc_parOD_listaArcos: #dicc tipo; parOD: listaArcos
    for l1 in Lines:
        for l2 in Lines:
            if l1 != l2:
                for k in dicc_parOD_listaNodos[(origen,destino)]:
                    if k != origen and k != destino:
                        mod.addConstr( v[k, (origen, destino), l1, l2] <=s[k,l2],
                                       name="v_k_nodoInterseccion_2%s_(%s,%s)_%s_%s" % (k,origen, destino, l1,l2))
mod.update()
```

14. Se crea la función objetivo y sus correspondientes linealizaciones.

```
# ------------      OBJECTIVE FUNCTION         --------------------
z_bus = mod.addVar(lb=0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="z_bus")
z_walk = mod.addVar(lb=0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="z_walk")
z_trans = mod.addVar(lb=0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="z_trans")
z_wait = mod.addVar(lb=0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="z_wait")
z_change = mod.addVar(lb=0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="z_change")
f_objetivo=mod.addVar(lb=0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="f_obj")

mod.addConstr(z_bus == 60/v_bus*
              quicksum(quicksum(quicksum(r[(i, j), (origen, destino),
              l]*float(dicc_OD_demanda[(origen, destino)])*float(dicc_arco_distancia[(i, j)])
                                  for (i,j) in dicc_parOD_listaArcos[(origen, destino)] )
                                      for l in Lines) for (origen, destino) in dicc_parOD_listaArcos),
                                          name="z_bus")
mod.update()


mod.addConstr(z_walk == 60/v_PED*
              quicksum(quicksum(t[(i, j), (origen, destino)]*float(dicc_OD_demanda[(origen,
                                                  destino)])*float(dicc_arco_distancia_PED[(i, j)])
                                  for (i,j) in dicc_parOD_listaAristasPED[(origen, destino)] )
                                      for (origen, destino) in dicc_parOD_listaAristasPED), name="z_walk")
mod.update()
```

```
mod.addConstr(z_trans == quicksum(quicksum(quicksum(quicksum(
    eta[i, (origen, destino), l1,l2]*float(dicc_OD_demanda[(origen, destino)])/2 for l2 in Lines if l2 !=l1)
        for l1 in Lines)
            for i in dicc_parOD_listaNodos[(origen, destino)]  if i != origen and i != destino)
                for (origen, destino) in dicc_parOD_listaNodos), name="z_trans")
mod.update()

#----- z_wait=sum_OD{sum_(O,j){sum_l{sigma^0l*p_O/2
mod.addConstr(z_wait ==
             quicksum(quicksum(quicksum(sigma[(i, j),
             (origen, destino), l] * float(dicc_OD_demanda[(origen, destino)]) / 2
                for l in Lines)
                    for (i,j) in dicc_parOD_listaArcos[(origen, destino)] if i ==origen)
                        for (origen, destino) in dicc_parOD_listaArcos), name="z_wait")
mod.update()

#----- z_change=sum_OD{sum_(O,j){sum_l{rho_i^0l*p_O/2
mod.addConstr(z_change ==quicksum(quicksum(quicksum(rho[i,
(origen, destino), l] * float(dicc_OD_demanda[(origen, destino)]) / 2
     for l in Lines)
        for i in dicc_parOD_listaNodos[(origen, destino)] if Lista_nodos_PED.__contains__(i))
            for (origen, destino) in dicc_parOD_listaNodos), name="z_change")
mod.update()
```

```
# ---------   Constraint linearization of products v_i Ôll' * h^l    ------------------------
#-------- eta_i^Oll' <= hw^l':
for l1 in Lines:
    for l2 in Lines:
        if l1 != l2:
            for (origen, destino) in dicc_parOD_listaNodos:
                for i in dicc_parOD_listaNodos[(origen, destino)]:
                    if i!= origen and i != destino:
                        mod.addConstr(eta[i,(origen,destino), l1, l2] <= hw[l2],
                                      name="eta_h_%s_(%s,%s)_%s_%s" % (i, origen, destino, l1,l2))
mod.update()

#-------- eta_i^Oll' <= 60/freq_max*v_i ^Oll':
for l1 in Lines:
    for l2 in Lines:
        if l1 != l2:
            for (origen, destino) in dicc_parOD_listaNodos:
                for i in dicc_parOD_listaNodos[(origen, destino)]:
                    if i!= origen and i != destino:
                        mod.addConstr(eta[i,(origen,destino), l1, l2] <= 60/freq_min*v[i, (origen, destino), l1, l2],
                                      name="eta_h_%s_(%s,%s)_%s_%s" % (i, origen, destino, l1,l2))
mod.update()
```

```
#-------- eta_i^Oll' >= h^l'- 60/freq_max*(1-v_i ^Oll'):
for l1 in Lines:
    for l2 in Lines:
        if l1 != l2:
            for (origen, destino) in dicc_parOD_listaNodos:
                for i in dicc_parOD_listaNodos[(origen, destino)]:
                    if i!= origen and i != destino:
                        mod.addConstr(eta[i, (origen, destino), l1, l2] >= hw[l2] - 60 / freq_min * (
                                    1 - v[i, (origen, destino), l1, l2]),
                                      name="eta_h_%s_(%s,%s)_%s_%s" % (i, origen, destino, l1, l2))
mod.update()

# ---------   Constraint linearization of products r_ij Ôl * h^l    ------------------------
#-------- sigma_ij^Ol <= hw^l:
for (origen, destino) in dicc_parOD_listaArcos: #dicc tipo; parOD: listaArcos
    for (i,j) in dicc_parOD_listaArcos[(origen,destino)]:
            for l in Lines:
                        mod.addConstr(sigma[(i, j), (origen, destino), l] <= hw[l],
                                      name="sigma_h_(%s,%s)_(%s,%s)_%s" % (i, j, origen, destino, l))
mod.update()

#-------- sigma_ij^Ol <= 60/freq_max*r_ij^Ol:
for (origen, destino) in dicc_parOD_listaArcos: #dicc tipo; parOD: listaArcos
    for (i,j) in dicc_parOD_listaArcos[(origen,destino)]:
            for l in Lines:
                        mod.addConstr(sigma[(i, j), (origen, destino), l] <= 60 / freq_min * r[(i, j),
                        (origen, destino), l], name="sigma_r_(%s,%s)_(%s,%s)_%s" % (i, j, origen, destino, l))
mod.update()
```

```
#-------- sigma_ij^Ol >=h^l- 60/freq_max*(1-r_ij^Ol):
for (origen, destino) in dicc_parOD_listaArcos: #dicc tipo; parOD: listaArcos
    for (i,j) in dicc_parOD_listaArcos[(origen,destino)]:
            for l in Lines:
                        mod.addConstr(sigma[(i, j), (origen, destino), l] >= hw[l] - 60 / freq_min * (1 - r[(i, j),
                        (origen, destino), l]), name="sigma_h_r_(%s,%s)_(%s,%s)_%s" % (i, j, origen, destino, l))
mod.update()

# ---------   Constraint linearization of products w_i^Ol * h^l    ------------------------
#----- rho_i^Ol <= w_i^Ol
for (origen, destino) in dicc_parOD_listaNodos:
    for i in dicc_parOD_listaNodos[(origen, destino)]:
        if Lista_nodos_PED.__contains__(i):
            mod.addConstr(rho[i, (origen, destino), l] <= hw[l], name="rho_h_%s(%s,%s)_%s" % (i, origen, destino, l))
mod.update()

#----- rho_i^Ol <= 60/freq_max* w_i^Ol
for (origen, destino) in dicc_parOD_listaNodos:
    for i in dicc_parOD_listaNodos[(origen, destino)]:
        if Lista_nodos_PED.__contains__(i):
            mod.addConstr(rho[i, (origen, destino), l] <= 60/freq_min*w[i, (origen, destino), l],
                          name="rho_w_%s(%s,%s)_%s" % (i, origen, destino, l))
mod.update()
```

```
#----- rho_i^Ol >= h^l-60/freq_max* (1-w_i^Ol)
for (origen, destino) in dicc_parOD_listaNodos:
    for i in dicc_parOD_listaNodos[(origen, destino)]:
        if Lista_nodos_PED.__contains__(i):
            mod.addConstr(rho[i, (origen, destino), l] >= hw[l]-60/freq_min*(1-w[i, (origen, destino), l]),
                          name="rho_h_w_%s(%s,%s)_%s" % (i, origen, destino, l))
mod.update()

# ---------   CREATION OF OBJECTIVE FUNCTION    ------------------------
mod.addConstr(f_objetivo == z_trans+z_bus+z_change+z_wait+z_walk, name="travel_time")
mod.setObjective(f_objetivo, sense=GRB.MINIMIZE)
mod.Params.MIPGap= 0.010
mod.Params.TIME_LIMIT = 2000
mod.write('Buses.lp')#imprime el modelo tal cual
mod.optimize()
```

15. Finalmente, se recoge la información obtenida para poder estudiar los resultados.

Se utilizará la red de referencia de *Mandl's, 1980* [1] compuesta por 15 nodos y 21 arcos bidireccionales que representan la unión de 15 ciudades suizas. Esta red es muy utilizada en la literatura, pero en este caso, se escalan las distancias para que sean compatibles en la red pedestre. Los atributos sobre los arcos denotan la distancia d_{ij} de viaje desde una parada hasta la siguiente. La red de bus, por la que circulan los autobuses, representada como $G = G(S, A)$, se muestra en la **Figura 8**, izquierda. La red pedestre, por donde caminan los pasajeros, representada como $G' = G(S, E)$, se muestra en la **Figura 9**, derecha.

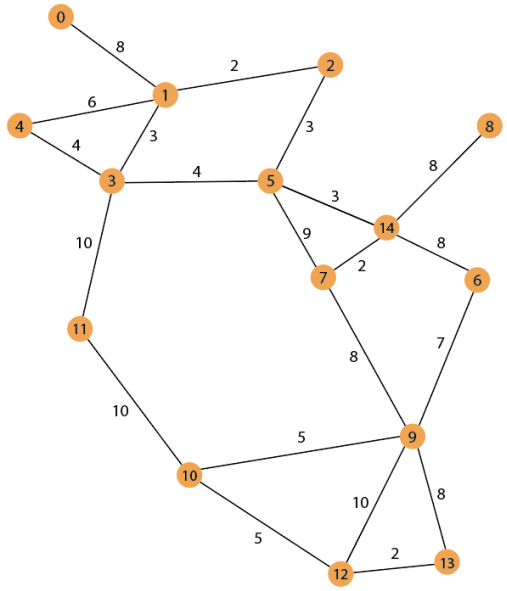


Figura 8: Red de autobús de Madl´s

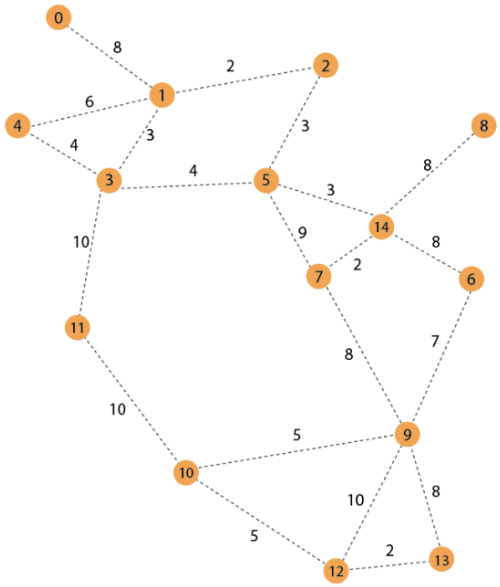


Figura 9: Red pedestre de Madl´s

La unión de ambas capas da lugar a la red bimodal, donde se tiene en cuenta tanto los movimientos de la red de autobús como la pedestre. Está representada como $BG = G(S, E \cup A)$, como se muestra en la **Figura 10**. La matriz de demanda de tránsito origen/destino de los pasajeros para la hora en punto se muestra en la **Tabla 1.**

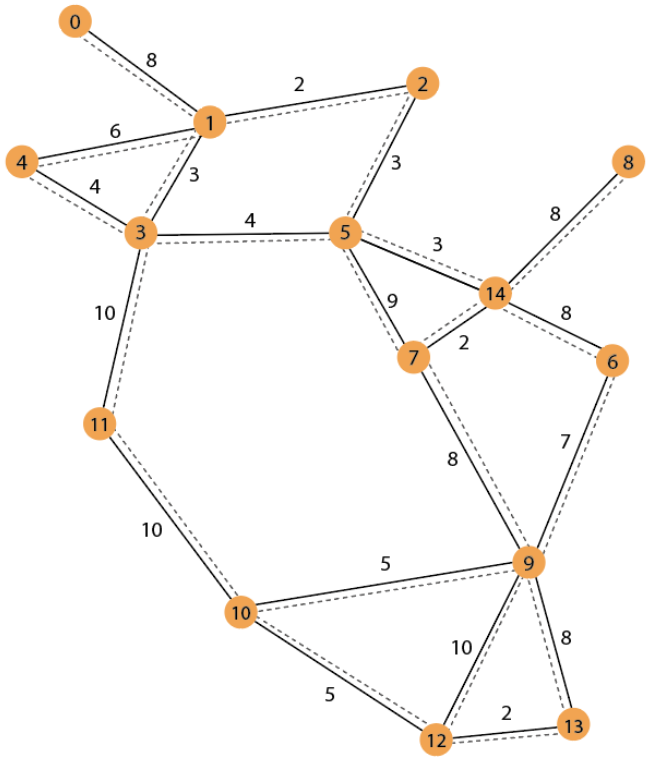


Figura 10: Red bimodal, basada en la red de Mandl´s

Los experimentos computacionales llevados a cabo permiten evaluar la veracidad del modelo planteado. El modelo se implementa con el lenguaje de Python y se utilizada Gurobi para obtener mejores resultados. Todos los cálculos se llevan a cabo en un ordenador estándar.

Tabla 1: Matriz origen/destino [23]

Origin/ Destination	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Sum by Destination
1	0	400	200	60	80	150	75	75	30	160	30	25	35	0	0	1320
2	400	0	50	120	20	180	90	90	15	130	20	10	10	5	0	1140
3	200	50	0	40	60	180	90	90	15	45	20	10	10	5	0	815
4	60	120	40	0	50	100	50	50	15	240	40	25	10	5	0	805
5	80	20	60	50	0	50	25	25	10	120	20	15	5	0	0	480
6	150	180	180	100	50	0	100	100	30	880	60	15	15	10	0	1870
7	75	90	90	50	25	100	0	50	15	440	35	10	10	5	0	995
8	75	90	90	50	25	100	50	0	15	440	35	10	10	5	0	995
9	30	15	15	15	10	30	15	15	0	140	20	5	0	0	0	310
10	160	130	45	240	120	880	440	440	140	0	600	250	500	200	0	4145
11	30	20	20	40	20	60	35	35	20	600	0	75	95	15	0	1065
12	25	10	10	25	15	15	10	10	5	250	75	0	70	0	0	520
13	35	10	10	10	5	15	10	10	0	500	95	70	0	45	0	815
14	0	5	5	5	0	10	5	5	0	200	15	0	45	0	0	295
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sum by Origin	1320	1140	815	805	480	1870	995	995	310	4145	1065	520	815	295	0	**15570**

6.1. Parámetros

Los requisitos de entrada del modelo son los siguientes:

- Red de bus/ pedestre: 15 nodos y 21 arcos bidireccionales.
- Matriz origen/destino representando los viajes entre los nodos.
- Velocidad media del autobús, $v_{bus} = 20\ km/h$.
- Velocidad media peatonal, $v_{ped} = 3\ km/h$.
- Límite inferior del número de paradas de una línea de autobús, $S_{min} = 1$.
- Límite superior del número de paradas de una línea de autobús, $S_{max} = 14$.
- Longitud máxima permitida de la línea de autobús, $\mathcal{M} = 50\ km$.
- Capacidad máxima de los autobuses dobles, $\mathcal{C}_1 = 110$.
- Capacidad máxima de los autobuses simples, $\mathcal{C}_2 = 81$.

6.2. Datos de entrada

Para analizar la influencia de los parámetros de entrada en la solución, se han definido 8 escenarios distinto variando los parámetros: número de líneas máximo, frecuencia de autobuses, número máximo de flota y número máximo de flota de cada tipo tal como se muestra en la **Tabla 2**, columnas 2-6, respectivamente. En la primera columna de esta tabla se muestra una numeración de los 8 escenarios, en la columna 7 el número de pasajeros de la red, en la 8 el número de pares totales de la red, en la 9 el GAP y en la décima columna, el CPU. Se puede observar cómo, para la misma lista de frecuencia [12, 20], el tiempo en resolver el modelo incrementa considerablemente del escenario con 3 líneas máximas de autobús al escenario con 5 líneas máximo. Como se muestra en la tabla, el CPU aumenta con el número de líneas.

Tabla 2: Variables de entrada

Esc	Num max de líneas	Lista de frecuencias	Num max flota	Num max flota Tipo 1	Num max flota Tipo 2	Pasajeros de la red	Num pares de la red	GAP	CPU
1	3	[12, 20]	6	3	3	15570	210	0.00947184	19.329
2	3	[12, 20]	10	5	5	15570	210	0.007715	10.47
3	3	[2, 4, 6]	6	3	3	15570	210	0.00707738	17.2509999
4	3	[2, 4, 6]	10	5	5	15570	210	0.0087827	18.9100001
5	5	[12, 20]	6	3	3	15570	210	0.00996487	488.543

6	5	[12, 20]	10	5	5	15570	210	0.00892408	106.714
7	5	[2, 4, 6]	6	3	3	15570	210	0.00966739	46.0669999
8	5	[2, 4, 6]	10	5	5	15570	210	0.00806225	41.9449999

Para estos parámetros, el escenario 2 cuenta con el mayor número de transbordos y el mejor valor de la función objetivo. Los escenarios 5, 7 y 8 no tienen transbordos (ver **Tabla 3**).

Tabla 3: Tiempos de viaje por cada escenario y función objetivo

Esc	Num viajes directos	Num transbordos	Tiempo en bus (z_bus)	Tiempo andando (z_walk)	Tiempo transbordando (z_transfer)	Tiempo esperando (z_wait)	Tiempo espera cambio modos (z_change)	Valor de la f.o.
1	164	46	10561.5	117804	4050	7590	0	140005.5
2	110	100	12880.5	102116	8620	8170	442.5	132229
3	166	44	11817	110892	12300	22250	0	157259
4	166	44	11313	112332	12300	21250	0	157195
5	210	0	8529	131212	0	8835	0	148576
6	158	52	11415	111972	6360	8115	0	137862
7	210	0	9477	126492	0	25450	0	161419
8	210	0	9477	126492	0	25450	0	161419

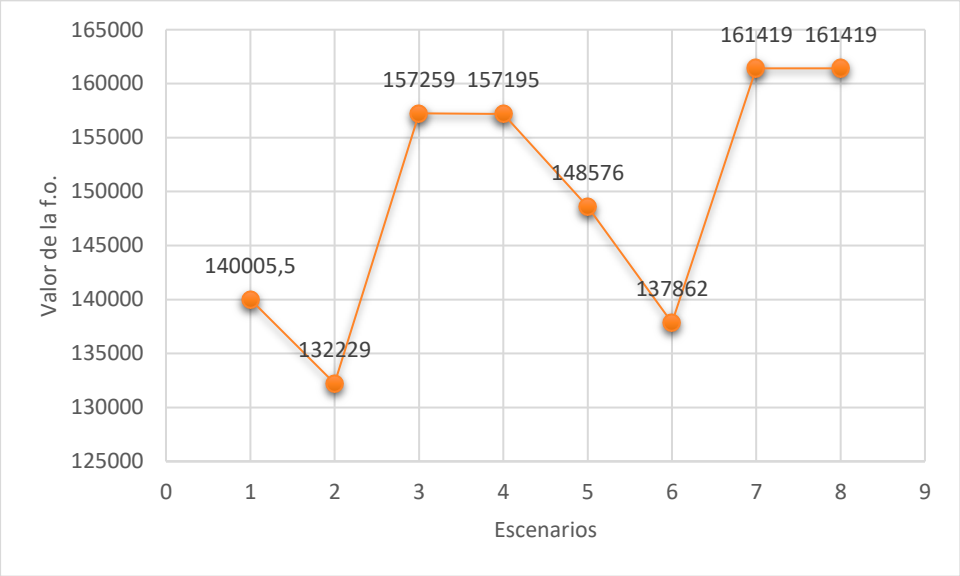
El modelo no tiene en cuenta la eliminación de bucles, por lo que en algunos escenarios no se realizan transbordos o cambios de modo, pero aun así presentar pero valor de la función objetivo.

6.3. Análisis de los resultados

En esta sección se hace un análisis de los resultados obtenidos en los 8 escenarios. Primero se analiza los resultados de manera global, luego por escenarios y finalmente se comparan los distintos escenarios.

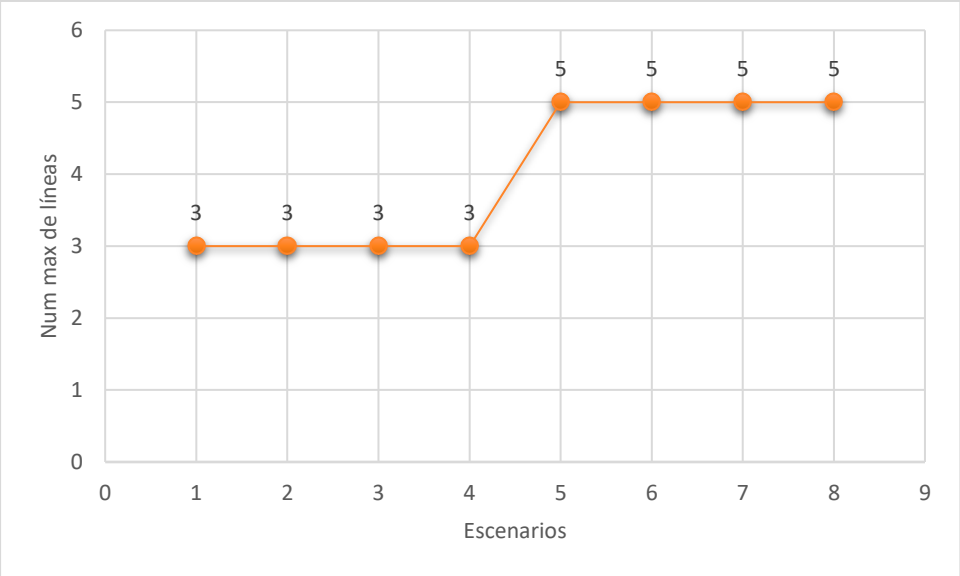
6.3.1. Resultados generales

Como se pretende minimizar el valor de la función objetivo, los mejores escenarios son el 2 y el 6, con unos valores de 132,229 y 137,862, respectivamente. Por otro lado, los peores escenarios serian el 8 y el 9 con 164,419, en ambos casos (**Gráfica 1**).



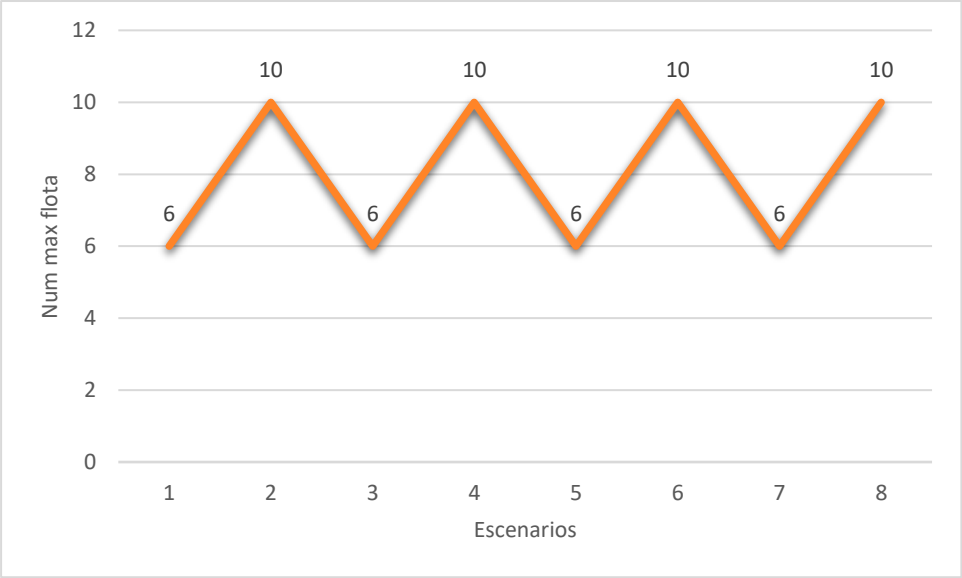
Gráfica 1: Valor de la función objetivo de cada escenario

El número máximo de líneas permitidas varia de 3 líneas para los escenarios 1, 2, 3 y 4; y de 5 líneas para los escenarios 5, 6, 7 y 8 (**Grafica 2**).



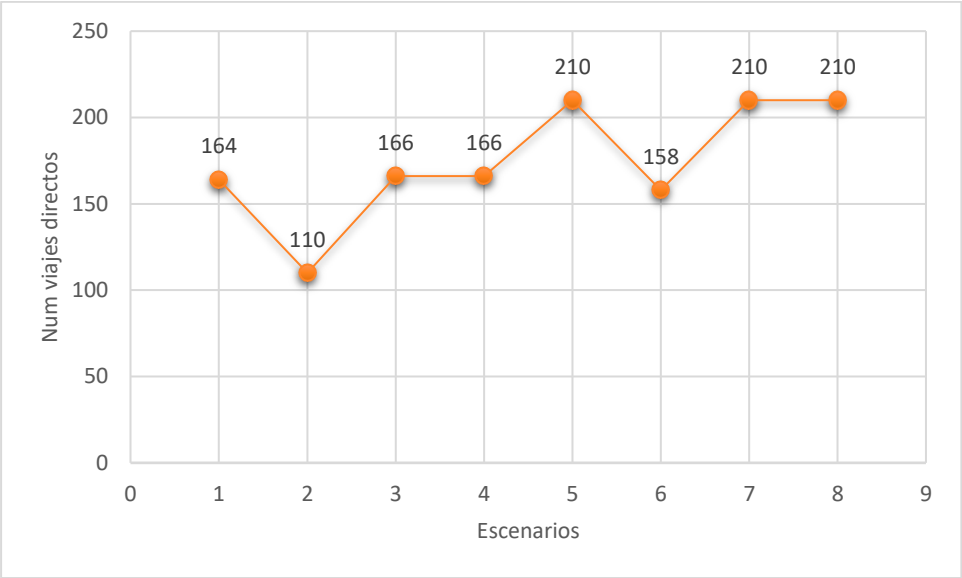
Gráfica 2: Número de líneas permitidas por escenario

El número máximo de la flota permitida va de 6 autobuses para los escenarios 1, 3, 5 y 7; y 10 autobuses para los escenarios 2, 4, 6 y 8 (**Gráfica 3**).



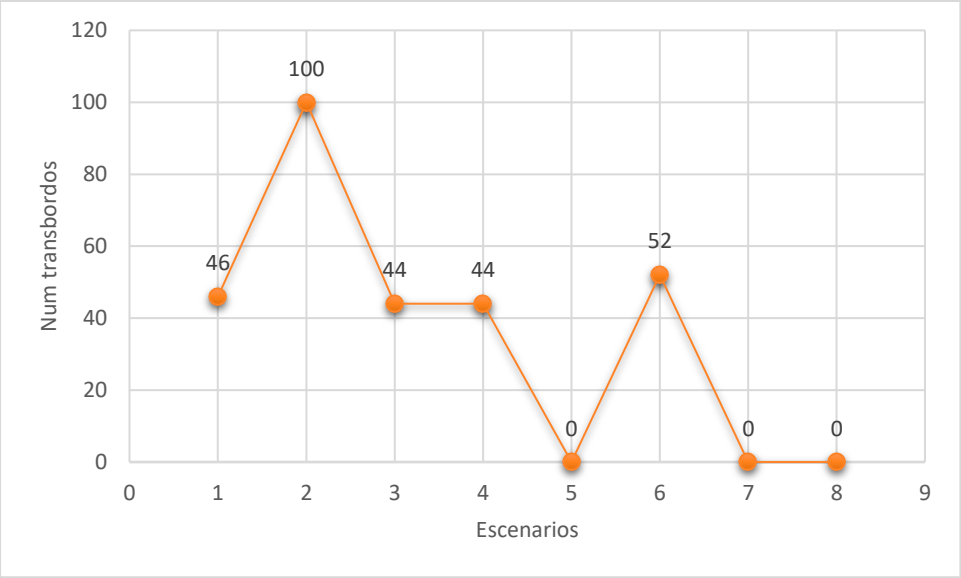
Gráfica 3: Número máximo de flota por escenario

Para que los pasajeros encuentren las líneas de autobús lo más atractivas posibles, el número de viajes directos debe ser lo más grande posible. Los escenarios con más viajes directos son el 5, el 7 y el 8, frente al escenario 2, con el menor número de viajes directos (**Gráfica 4**).



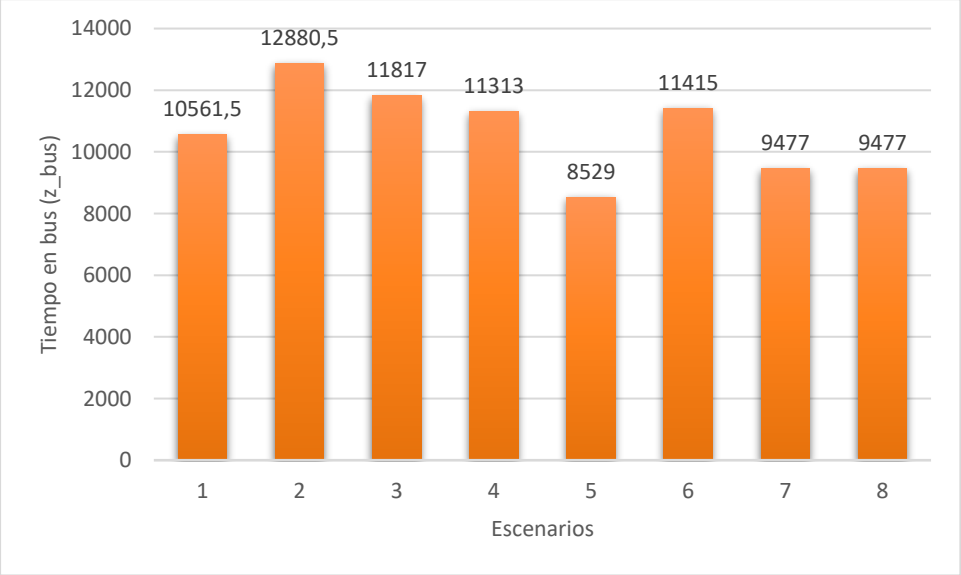
Gráfica 4: Número de viajes directos por escenario

En contrapartida, lo que el pasajero no quiere, es tener que hacer transbordo. Por ello, los escenarios con mayor número de transbordo son menos atractivos. El escenario 2 tiene hasta 100 transbordos mientras que los escenarios 5, 7 y 8 no tienen transbordos (**Gráfica 5**).



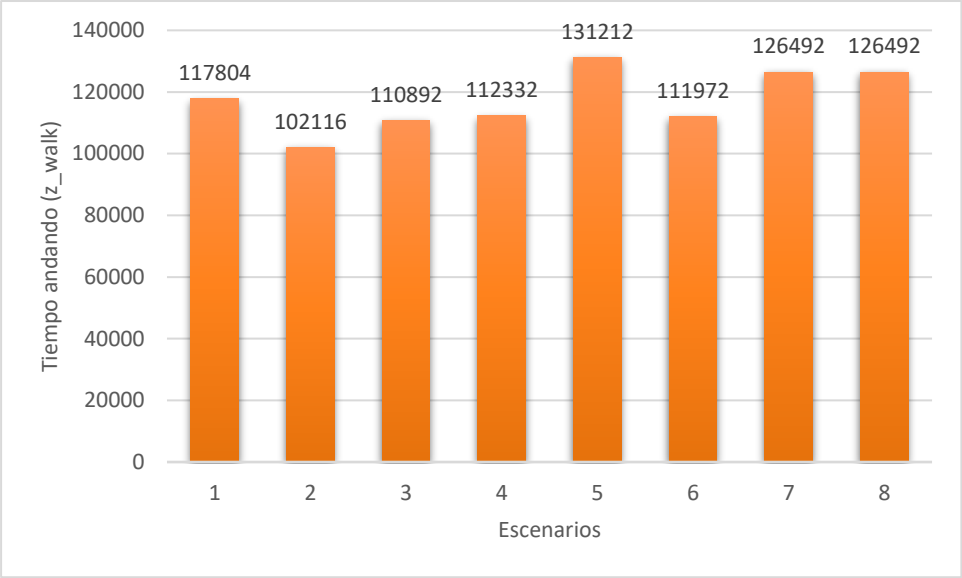
Gráfica 5: Número de transbordos por escenario

Para minimizar el valor de la función objetivo, se debe minimizar los tiempos totales de viaje del pasajero. El escenario 5 cuenta con el menor tiempo de transporte en autobús frente al resto de escenarios (**Gráfica 6**).



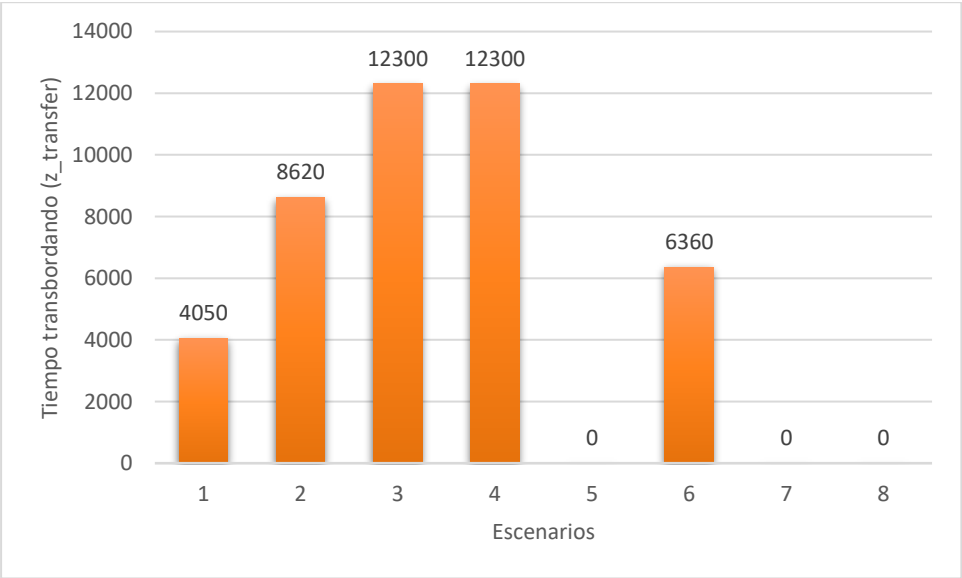
Gráfica 6: Tiempo de transporte en el autobús por escenario

El escenario 2 tiene el menor tiempo andando del pasajero, seguido muy de cerca por los escenarios 3, 4 y 6 (**Gráfica 7**).



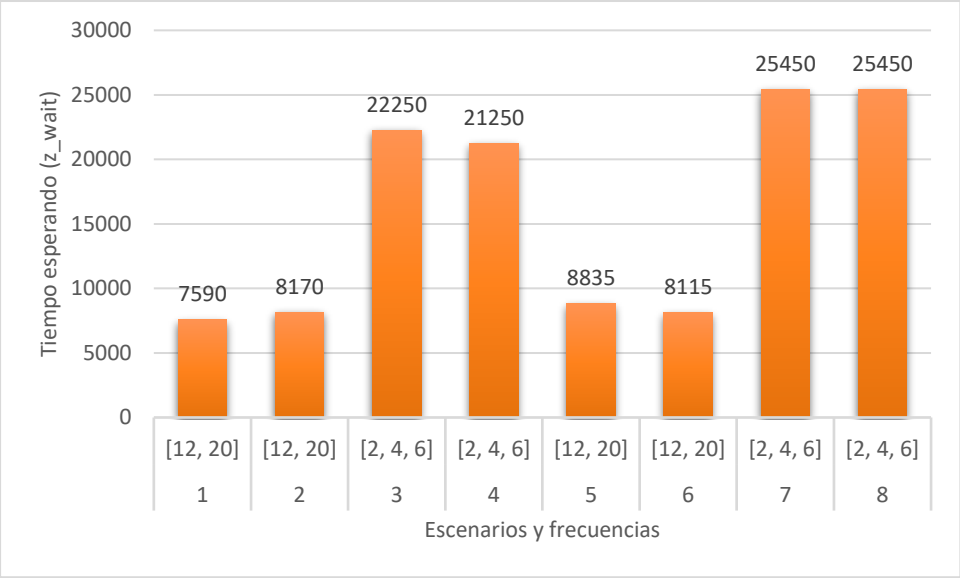
Gráfica 7: Tiempo andando del pasajero por escenario

Como se observa en la **Gráfica 5**, los escenarios 5, 7 y 8 no tienen transbordo, por lo tanto, serán estos mismos los que no tengan tiempo de transbordo, seguido por el escenario 1, con la menor cantidad (**Gráfica 8**).



Gráfica 8: Tiempo de transbordo por escenario

Aquellos escenarios que menos tiempo de espera tienen son el 1, el 2, el 5 y el 6, coincidiendo, además, con los escenarios con mayor frecuencia (**Gráfica 9**).



Gráfica 9: Tiempo de espera por escenario y frecuencia

6.3.2. Resultados por escenario

En la **Tabla 4** se muestran los resultados obtenidos de cada escenario, las líneas que se activan para cada uno, los nodos y los arcos que las componen, el número de transbordos necesarios para satisfacer la frecuencia, el tipo de autobús para transportar a los pasajeros de la línea y la longitud de la misma.

Tabla 4: Resultados por cada escenario

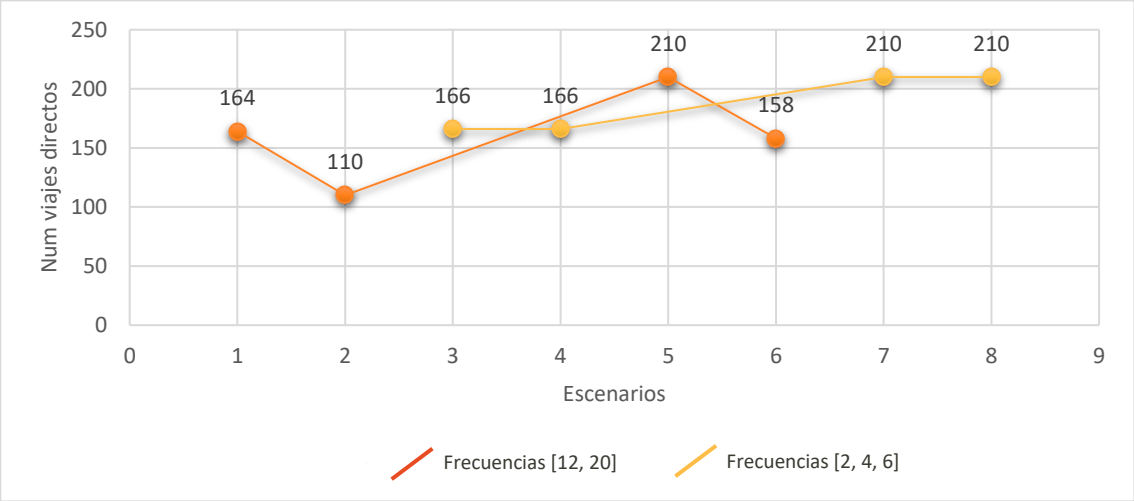
Esc	Línea activada	Nº nodos	Nº arcos	Nº autobuses	Tipo autobus	Frecuencia	Longitud	Pasajeros que usan línea	Nº pares que usan línea
1	1	4	4	2	Tipo 2	20	2	1335	12
	2	4	4	3	Tipo 1	20	3	2675	19
	3	4	4	1	Tipo 2	12	1.6	1440	4
2	1	10	10	5	Tipo 1	20	5	3760	37
	2	8	8	3	Tipo 2	12	5	1695	22
	3	6	6	2	Tipo 2	20	2	1735	9
3	1	10	10	2	Tipo 1	6	5.4	1795	12
	2	10	10	3	Tipo 2	6	7.2	1885	22
	3	8	8	1	Tipo 1	6	3.2	2000	4
4	1	8	8	2	Tipo 1	6	4.8	2330	22
	2	12	12	3	Tipo 1	6	8.2	2030	13
	3	8	8	2	Tipo 2	6	4	1120	6
5	1	4	4	2	Tipo 2	20	2	1240	4
	3	8	8	3	Tipo 1	12	5	2070	8
	4	2	2	1	Tipo 2	20	1	1200	2
6	1	4	4	3	Tipo 1	20	3	2600	15
	2	7	7	3	Tipo 2	20	2.9	1890	9

	3	6	6	1	Tipo 1	12	1.6	290	4
	4	4	4	1	Tipo 1	12	1.6	310	3
	5	4	4	2	Tipo 2	20	2	980	4
7	1	10	10	1	Tipo 1	6	3	1680	5
	2	6	6	1	Tipo 2	6	2.8	1100	4
	3	10	10	2	Tipo 2	6	6.6	2170	10
	4	8	8	1	Tipo 1	6	2.48	140	3
8	1	12	12	2	Tipo 2	6	5.34	655	4
	2	10	10	2	Tipo 1	6	4.1	890	5
	3	9	9	2	Tipo 2	6	5.1	1410	6
	4	12	12	2	Tipo 1	6	5.14	1600	5
	5	10	10	1	Tipo 1	6	2.84	535	2

Se observa que el escenario 5, a pesar de contar con 5 líneas máximas de bus, solo activa 3 de ellas.

6.3.3. Comparación de escenarios

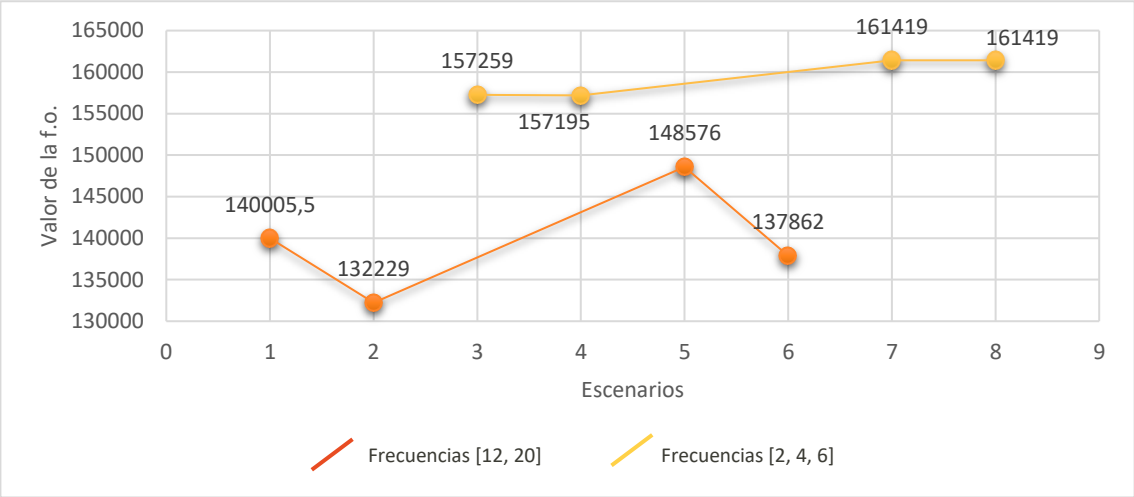
Agrupando los resultados según las diferentes frecuencias, se obtienen dos bloques: uno cuyas frecuencias son [2, 4, 6] y otro con frecuencias de [12, 20]. Para estos dos grupos, el número de viajes directos se muestra en la **Gráfica 10**. El grupo de escenarios con la frecuencia más alta (naranja) tiene una media de 160 viajes directos, frente a las frecuencias más bajas (amarillo) cuya media es de 188 viajes directos. La frecuencia no es un indicador de la cantidad de viajes directos de los escenarios, pero ayuda a comprender los resultados.



Gráfica 10: Número de viajes directos por grupos de frecuencias

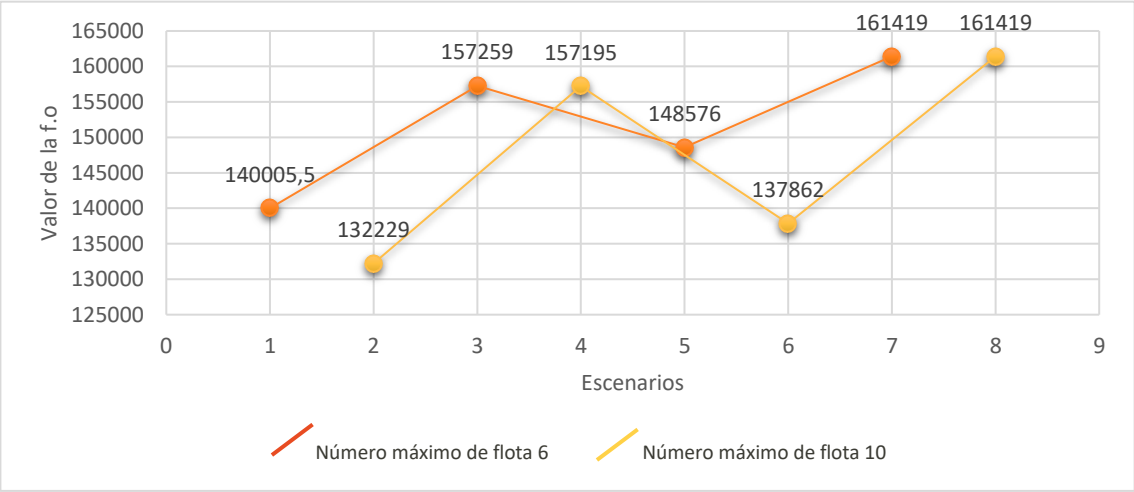
El valor de la función objetivo para las frecuencias más altos (naranja) es bastante más pequeño que para las frecuencias más bajas (amarillo). Esto es debido a que, a

mayores frecuencias, menores tiempos de viaje de los pasajeros y por lo tanto mejores resultados (**Gráfica 11**).



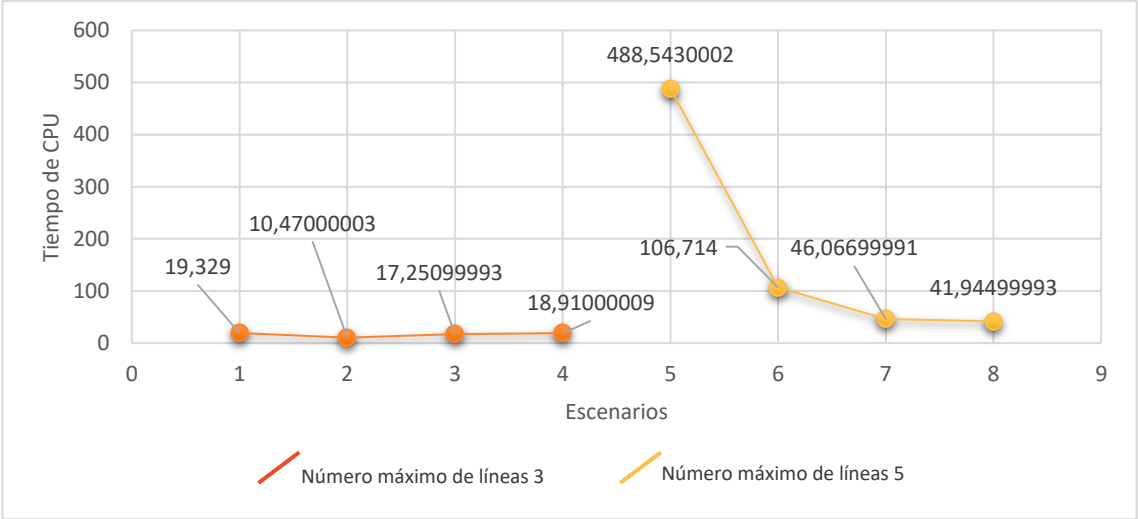
Gráfica 11: Valores de la función objetivo por grupos de frecuencias

Para los escenarios cuya flota máxima es de 6 autobuses (naranja) la media de la función objetivo es de 152,814.875 frente a los escenarios con una flota máxima de 10 autobuses (amarilla) cuyo valor de la función objetivo medio es de 147,176.25 (**Gráfica 12**). Puede observarse que una mayor flota permite tener una mayor frecuencia, lo que si repercute en la función objetivo.



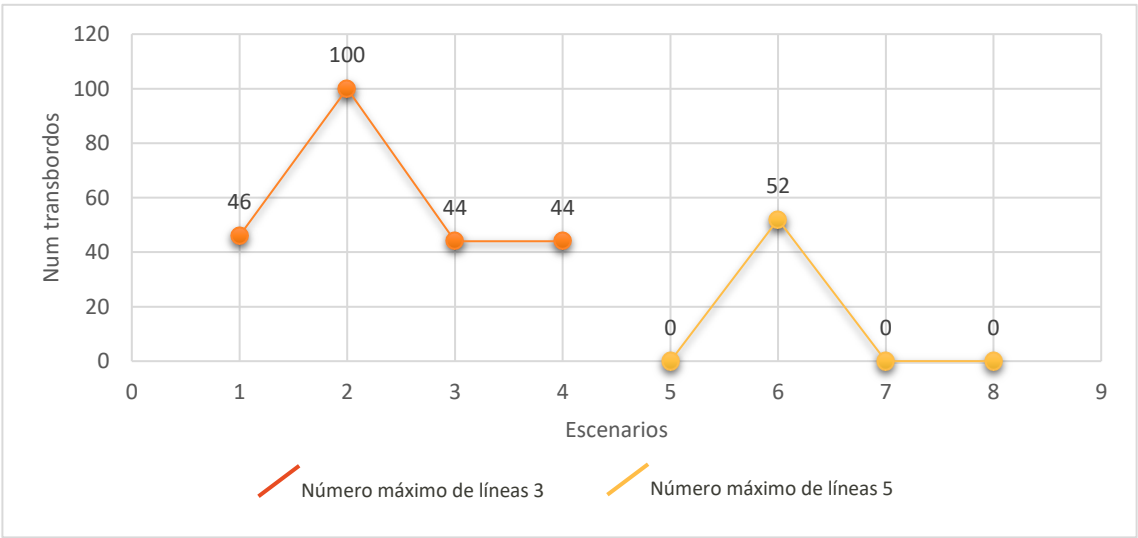
Gráfica 12: Valor de la función objetivo por grupos de flota

El tiempo de computación necesario para resolver el modelo es bastante superior en aquellos escenarios con un máximo de 5 líneas de autobús (amarillo). Los escenarios con un máximo de 3 líneas de autobús se compilan más rápido (naranja). A mayor líneas de bus, más tiempo es necesario para la obtención de resultados (**Gráfica 13**).



Gráfica 13: Tiempo de CPU por grupos de líneas de autobús

Con mayor número de líneas (amarillo), los transbordos necesarios son menores, llegando a no ser necesario ningún tipo de transbordo en los escenarios 5, 7 y 8. Para el grupo con un máximo de 3 líneas de autobús (naranja), se comprueba que el número de transbordo es superior (**Gráfica 14**).



Gráfica 14: Número máximo de transbordos por grupos de líneas de autobús

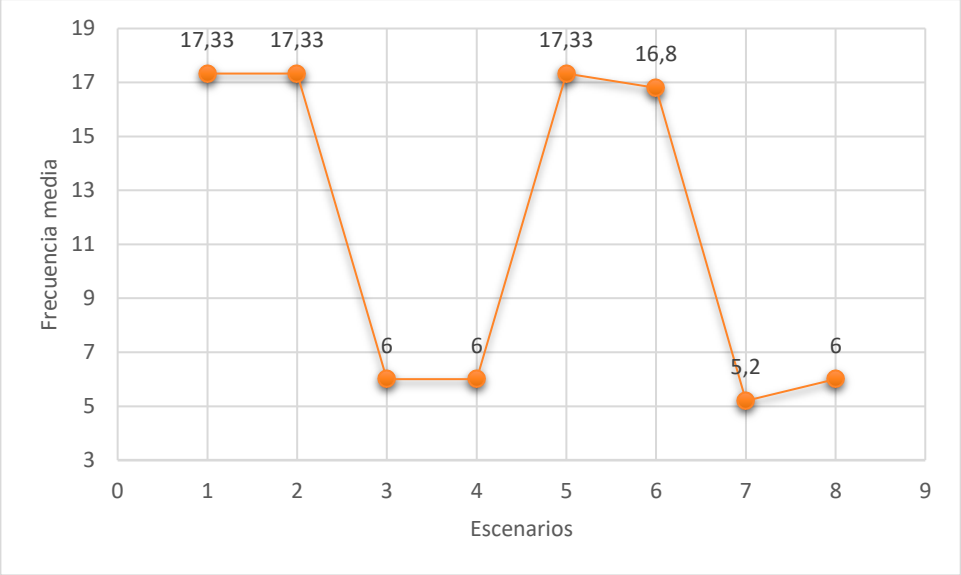
6.3.4. Comparación global

Poniendo en perspectiva todos los datos obtenidos, se consigue la **Tabla 5**, que muestra los valores medios de cada escenario y el porcentaje de tiempos utilizados en cada caso.

Tabla 5: Valores medios y resumen por escenario

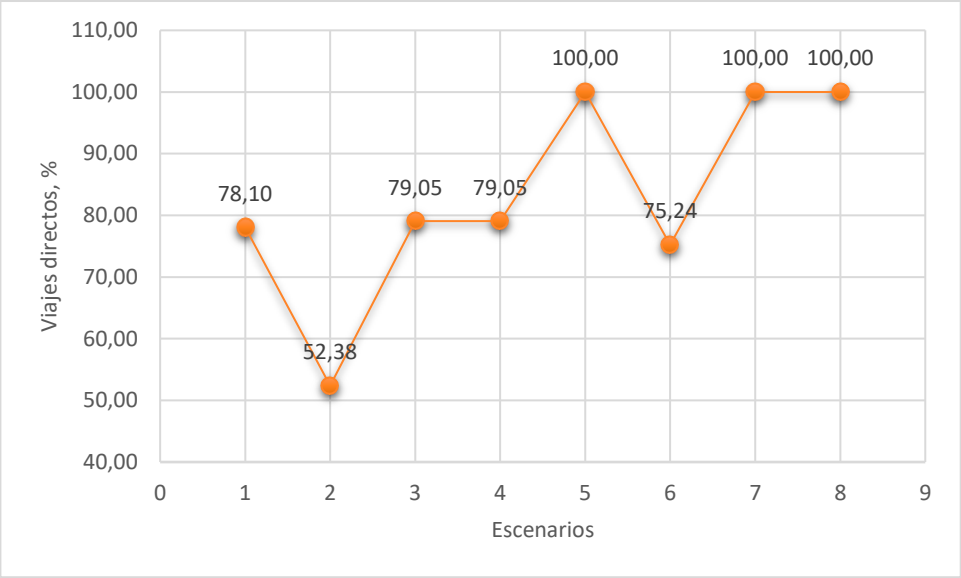
Esc	Num max de líneas	Frec media	% Vj direc	% Transb	Long media	% T bus	% T andando	% T transb	% T espera	% T cambio	Valor de la f.o.
1	3	17,33	78,10	21,90	2,2	7,54	84,14	2,89	5,42	0	140005,5
2	3	17,33	52,38	47,62	4	9,74	77,23	6,52	6,18	0,33	132229
3	3	6	79,05	20,95	5,27	7,51	70,52	7,82	14,15	0	157259
4	3	6	79,05	20,95	5,67	7,20	71,46	7,82	13,52	0	157195
5	5	17,33	100,00	0,00	2,67	5,74	88,31	0,00	5,95	0	148576
6	5	16,8	75,24	24,76	2,22	8,28	81,22	4,61	5,89	0	137862
7	5	5,2	100,00	0,00	3,46	5,87	78,36	0,00	15,77	0	161419
8	5	6	100,00	0,00	4,5	5,87	78,36	0,00	15,77	0	161419

Cada escenario tiene un número máximo de líneas que varía entre las 3 y las 5 líneas de autobús, como se muestra en la segunda columna de la **Tabla 5**. En la tercera columna se muestra la media de las frecuencias de las líneas activas de cada escenario, siendo el escenario 1, 2 y 5 las de mayor frecuencia (**Gráfica 15**).



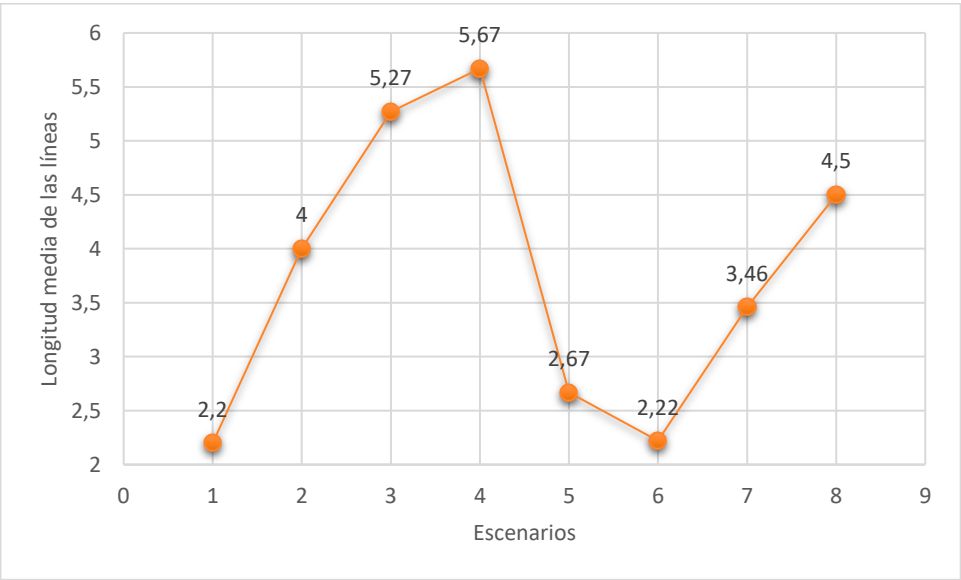
Gráfica 15: Frecuencia media por escenario.

Se realizan un total de 210 viajes en cada escenario, en la cuarta columna se muestra el porcentaje de los viajes directos y en la quinta columna de la **Tabla 5** el porcentaje de los viajes con transbordo. Los escenarios con mayores viajes directos son el 5, el 7 y el 8 (**Gráfica 16**).



Gráfica 16: Porcentaje de viajes directos por escenario

La sexta columna de la **Tabla 5** muestra la longitud media de todas las líneas de cada escenario. El escenario 4 tiene la mayor media (**Gráfica 17**).



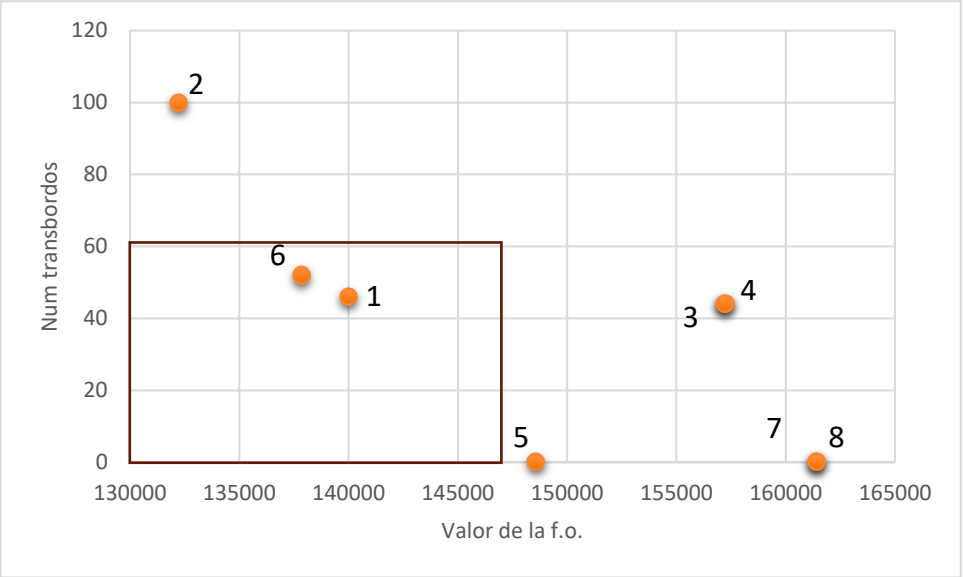
Gráfica 17: Longitud media de las líneas por escenario

La séptima, octava, novena, decima y décima primera columna de la **Tabla 5** muestran el porcentaje del tiempo en autobús, andando, de transbordo, de espera y de cambio de modo respectivamente sobre el total de la función objetivo (ver **Gráfica 1**), que se muestra en la décima segunda columna de la **Tabla 5.**

8. Conclusiones

El objetivo principal del trabajo es diseñar una red de transporte de autobuses atractiva para los pasajeros. Esto se consigue reduciendo el tiempo total de viaje del pasajero que se mueve dentro de la red. Las frecuencias más altas también son más atrayentes para los pasajeros ya que los tiempos de viaje son mejores.

De los escenarios planteados, la mejor función objetivo es la del escenario 2 pero implica un gran número de transbordos. Los escenarios 5, 7 y 8 no tienen transbordos, pero su función objetivo es bastante superior. Si se busca conseguir un equilibrio entre la función objetivo y el número de transbordos, los escenarios más competitivos son el 6 y el 1 (ver **Gráfica 18**).



Gráfica 18: Número de transbordos frente al valor de la función objetivo

En cuanto a las conclusiones del proyecto, se ha planteado la integración de las etapas de diseño de rutas y determinación de frecuencias y flota, obteniendo una serie de resultados para respaldar su eficiencia. Se han diseñado las líneas de transporte público de autobuses desde cero, determinando cada línea, su itinerario, las paradas terminales, la frecuencia y la flota. Por último, se han establecido las rutas que seguirán los pasajeros en función de minimizar el tiempo total de viaje, teniendo en cuenta las limitaciones de parte del operario.

8.1. Mejoras futuras

Como se nombró previamente, estos resultados corresponden a un modelo sin restricciones de antibucle. Por lo tanto, pueden existir subtours. Este paso sería interesante estudiarlo en un futuro y ver cómo cambian los resultados al añadir las nuevas restricciones para eliminar estos bucles en los escenarios planteados.

El modelo tarda poco tiempo en resolver la red de Mandl´s, lo que ayudará a poder utilizar el mismo modelo en una red más grande, compatible con el tamaño de una ciudad o, incluso, una red interurbana.

BIBLIOGRAFÍA

[1] C. E. Mandl, «Evaluation and optimization of urban public transportation networks», *Eur. J. Oper. Res.*, vol. 5, n.º 6, pp. 396-404, dic. 1980, doi: 10.1016/0377-2217(80)90126-5.

[2] V. Guihaire y J.-K. Hao, «Transit network design and scheduling: A global review», *Transp. Res. Part Policy Pract.*, vol. 42, pp. 1251-1273, dic. 2008, doi: 10.1016/j.tra.2008.03.011.

[3] C. Iliopoulou, K. Kepaptsoglou, y E. Vlahogianni, «Metaheuristics for the transit route network design problem: a review and comparative analysis», *Public Transp.*, vol. 11, n.º 3, pp. 487-521, oct. 2019, doi: 10.1007/s12469-019-00211-2.

[4] J. S. C. Chew, L. S. Lee, y H. V. Seow, «Genetic Algorithm for Biobjective Urban Transit Routing Problem», *J. Appl. Math.*, vol. 2013, p. e698645, dic. 2013, doi: 10.1155/2013/698645.

[5] S. B. Pattnaik, S. Mohan, y V. M. Tom, «Urban Bus Transit Route Network Design Using Genetic Algorithm», *J. Transp. Eng.*, vol. 124, n.º 4, pp. 368-375, jul. 1998, doi: 10.1061/(ASCE)0733-947X(1998)124:4(368).

[6] M. Bielli, M. Caramia, y P. Carotenuto, «Genetic algorithms in bus network optimization», *Transp. Res. Part C Emerg. Technol.*, vol. 10, n.º 1, pp. 19-34, feb. 2002, doi: 10.1016/S0968-090X(00)00048-6.

[7] P. Chakroborty, «Genetic Algorithms for Optimal Urban Transit Network Design», *Comput.-Aided Civ. Infrastruct. Eng.*, vol. 18, n.º 3, pp. 184-200, 2003, doi: 10.1111/1467-8667.00309.

[8] V. M. Tom y S. Mohan, «Transit Route Network Design Using Frequency Coded Genetic Algorithm», *J. Transp. Eng.*, vol. 129, n.º 2, pp. 186-195, mar. 2003, doi: 10.1061/(ASCE)0733-947X(2003)129:2(186).

[9] Y.-J. Lee y V. R. Vuchic, «Transit Network Design with Variable Demand», *J. Transp. Eng.*, vol. 131, n.º 1, pp. 1-10, ene. 2005, doi: 10.1061/(ASCE)0733-947X(2005)131:1(1).

[10] F. Zhao y X. Zeng, «Simulated Annealing–Genetic Algorithm for Transit Network Optimization», *J. Comput. Civ. Eng.*, vol. 20, n.º 1, pp. 57-68, ene. 2006, doi: 10.1061/(ASCE)0887-3801(2006)20:1(57).

[11] F. Zhao y X. Zeng, «Optimization of User and Operator Cost for Large-Scale Transit Network», *J. Transp. Eng.*, vol. 133, n.º 4, pp. 240-251, abr. 2007, doi: 10.1061/(ASCE)0733-947X(2007)133:4(240).

[12] B. R. Marwah, F. S. Umrigar, y S. B. Patnaik, «OPTIMAL DESIGN OF BUS ROUTES AND FREQUENCIES FOR AHMEDABAD», *Transp. Res. Rec.*, n.º 994, 1984, Accedido: 10 de junio de 2023. [En línea]. Disponible en: https://trid.trb.org/view/269877

[13] J. F. Guan, H. Yang, y S. C. Wirasinghe, «Simultaneous optimization of transit line configuration and passenger line assignment», *Transp. Res. Part B Methodol.*, vol. 40, n.º 10, pp. 885-902, dic. 2006, doi: 10.1016/j.trb.2005.12.003.

[14] R. van Nes, R. Hamerslag, y B. H. Immers, «DESIGN OF PUBLIC TRANSPORT NETWORKS», *Transp. Res. Rec.*, n.º 1202, 1988, Accedido: 10 de junio de 2023. [En línea]. Disponible en: https://trid.trb.org/view/302174

[15] R. Z. Farahani, E. Miandoabchi, W. Y. Szeto, y H. Rashidi, «A review of urban transportation network design problems», *Eur. J. Oper. Res.*, vol. 229, n.º 2, pp. 281-302, sep. 2013, doi: 10.1016/j.ejor.2013.01.001.

[16] R. Borndörfer, M. Grötschel, y M. E. Pfetsch, «A Path-Based Model for Line Planning in Public Transport».

[17] R. Borndörfer, M. Grötschel, y M. E. Pfetsch, «A Column-Generation Approach to Line Planning in Public Transport», *Transp. Sci.*, vol. 41, n.º 1, pp. 123-132, feb. 2007, doi: 10.1287/trsc.1060.0161.

[18] C. Winston y V. Maheshri, «On the social desirability of urban rail transit systems», *J. Urban Econ.*, vol. 62, n.º 2, pp. 362-382, sep. 2007, doi: 10.1016/j.jue.2006.07.002.

[19] L. Zhang, H. Yang, D. Wu, y D. Wang, «Solving a discrete multimodal transportation network design problem», *Transp. Res. Part C Emerg. Technol.*, vol. 49, pp. 73-86, dic. 2014, doi: 10.1016/j.trc.2014.10.008.

[20] R. Camporeale, L. Caggiani, A. Fonzone, y M. Ottomanelli, «Better for Everyone: An Approach to Multimodal Network Design Considering Equity», *Transp. Res. Procedia*, vol. 19, pp. 303-315, ene. 2016, doi: 10.1016/j.trpro.2016.12.090.

[21] A. De-Los-Santos, D. Canca, y E. Barrena, «Mathematical formulations for the bimodal bus-pedestrian social welfare network design problem», *Transp. Res. Part B Methodol.*, vol. 145, pp. 302-323, mar. 2021, doi: 10.1016/j.trb.2021.01.010.

[22] E. W. Dijkstra, «A note on two problems in connexion with graphs», *Numer. Math.*, vol. 1, n.º 1, pp. 269-271, dic. 1959, doi: 10.1007/BF01386390.

[23] R. O. Arbex y C. B. Da Cunha, «Efficient transit network design and frequencies setting multi-objective optimization by alternating objective genetic algorithm», *Transp. Res. Part B Methodol.*, vol. 81, pp. 355-376, nov. 2015, doi: 10.1016/j.trb.2015.06.014.