Trabajo Fin de Grado
Ingeniería de Tecnologías Industriales,
Intensificación de Organización y Producción

Diseño óptimo de horarios de líneas de ferrocarril de tránsito rápido con demanda variable, integración de estrategias Stop-skipping y Short-Turning

Autor: Jose María Ruiz Sánchez de Puerta

Tutor: José David Canca Ortiz

Dpto. Organización y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado
Ingeniería de Tecnologías Industriales
Intensificación Organización y Producción

Diseño óptimo de horarios de líneas de ferrocarril de tránsito rápido con demanda variable, integración de estrategias Stop-skipping y Short-Turning

Autor:

Jose María Ruiz Sánchez de Puerta

Tutor:

Jose David Canca Ortiz

Catedrático de Universidad

Dpto. Organización y Gestión de Empresas I

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2021

A mi familia y seres queridos

Resumen

COVID-19 se ha extendido alrededor del mundo causando un tremendo cambio estructural y afectando las cadenas de suministros globales y las operaciones financieras, [19]. Por ello, es de gran importancia implementar de manera correcta las medidas sanitarias, impuestas por las autoridades sanitarias, como el distanciamiento de seguridad o la limitación de aforos.

La organización y planificación en los horarios de los transportes públicos, tales como las líneas de ferrocarril metropolitano, o metro, son muy complejas. Por ello, hay que implementar técnicas y utilizar herramientas potentes. En este estudio, a partir del trabajo de [20], se va a desarrollar un modelo de optimización para el diseño de horarios de líneas de metro con demanda variable que soporta estrategias Stop-skipping y Short-turning, cuyo propósito es el de disponer de un mayor número de servicios en los segmentos de mayor demanda de pasajeros, reduciendo el tiempo de espera en las estaciones. Además, se consideran zonas de operación, la capacidad de los trenes, las operaciones de cambio de sentido y el número de trenes disponibles.

Para ello, se desarrolla una formulación no lineal, que posteriormente se resuelve utilizando la herramienta de optimización Gurobi Optimizer. El modelo se desarrolla mediante programación en lenguaje Python 3.8. Finalmente, se realizan varios experimentos, y se discuten los resultados.

Abstract

COVID-19 has spread around the world causing tremendous structural change and affecting global supply chains and financial operations, [19]. Therefore, it is of great importance to correctly implement sanitary measures, imposed by health authorities, such as safety distancing or capacity limitation.

The organisation and planning of public transport schedules, such as metro and underground lines, are very complex. Therefore, it is necessary to implement techniques and use powerful tools. In this study, based on the work of [20], an optimisation model is developed for the design of metro line timetables with variable demand that supports stop-skipping and short-turning strategies, the purpose of which is to provide a greater number of services in the segments with the highest passenger demand, reducing waiting time at stations. In addition, operating zones, train capacity, change of direction operations and the number of available trains are considered.

For this purpose, a non-linear formulation is developed, which is subsequently solved using the optimisation tool Gurobi Optimizer. The model is developed using Python 3.8 programming language. Finally, several experiments are performed, and the results are discussed.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 OBJETIVOS

El metro permite contribuir eficazmente al logro de todos los objetivos de una política de desarrollo urbano: mejora la eficiencia de la economía de la ciudad al reducir los costos de viajar, y además genera un mayor nivel de actividades en el centro de la ciudad aprovechando las economías de aglomeración según [15]. Sin embargo, el desnivel de la falta de homogeneidad de demanda de pasajeros a lo largo del tiempo y en las diferentes estaciones de cada línea hace que se produzcan situaciones en las que excede la capacidad de transporte de los trenes, lo que provoca que los pasajeros vean aumentados sus tiempos de espera en las plataformas.

Debido a la situación en la que se encuentra el mundo desde mediados de 2019, con la expansión del virus COVID-19, declarada como "Public Health Emergency of International Concern" el 30 de enero, [9], es de vital importancia mantener adecuados niveles de ocupación de los trenes en las líneas de metro. Para mejorar el acceso a los vehículos y reducir los tiempos de espera en los andenes, se suele usar diferentes estrategias de aceleración.

Generalmente, existen tres estrategias de aceleración, que son conocidas como , Express-local, Skip-stopping y Short-turning [18]. Este estudio se centra en la implementación de las dos últimas de las estrategias nombradas en una línea de metro bidireccional, considerando una demanda de pasajeros variable con el tiempo, consiguiendo así disponer de un mayor número de servicios en los segmentos que sufren una mayor demanda. Esta estrategia se ha adoptado en grandes ciudades como Beijing, Paris y Tokio.

El objeto de este proyecto consiste pues, en el diseño e implementación de un modelo matemático de optimización y su posterior resolución a fin de determinar el número de servicios que se deben utilizar en ambos sentidos de una línea de metro, el número de trenes que se necesitan (sin exceder el número máximo de trenes disponible), las estaciones que deben ser atendidas por cada servicio, considerando explícitamente la capacidad de los trenes y una demanda de viajes variable en el tiempo..

2 INTRODUCCIÓN A LA PLANIFICACIÓN DE HORARIOS

Antes de abordar la explicación del modelo matemático exacto y la exposición de resultados y conclusiones, se va a hacer una introducción sobre los elementos que componen un sistema ferroviario, así como las estrategias que se utilizan para planificación de horarios de líneas de transporte metropolitano.

Comentar, que los sistemas a los que se va a hacer referencia en este trabajo, son los existentes dentro de España y que, por tanto, quedan dentro de la legislación consolidada recogida en la Ley 38/2015, del 29 de septiembre, en el sector ferroviario, [21].

2.1. Elementos de un Sistema Ferroviario

Los elementos de un sistema ferroviario se pueden dividir en subsistemas: infraestructura, material rodante y depósitos. Actualmente, se trabaja en todos estos factores técnicos, además de los factores de administrativos y jurídicos. En este sentido, las normas de interoperabilidad que se están desarrollando actualmente, y desde hace pocos años, pretenden ser la base del desarrollo de una futura red europea, [14].

2.1.1. Infraestructura

La infraestructura ferroviaria engloba todos los elementos que formen parte de las vías principales, de los servicios y de los ramales de desviación para particulares, a excepción de las vías situadas dentro de los talleres de reparación de material rodante y de los depósitos o garajes de máquinas de tracción (Ver elementos de la infraestructura ferroviaria)

A efectos de la citada ley [21], se entiende como infraestructura ferroviaria, todos aquellos elementos que formen parte de las vías principales y de las de servicios y ramales de desvío, con excepción de las situadas dentro de los talleres de reparación de material rodante y de los depósitos o garajes de máquinas de tracción. Entre estos elementos, se encuentran los terrenos, las estaciones de transporte de viajeros, las terminales de transporte de mercancías, las obras civiles, los pasos a nivel, los caminos de servicio, las instalaciones vinculadas a la seguridad, a las telecomunicaciones, a la electrificación, a la señalización de las líneas, al alumbrado, al almacenamiento de combustible necesario para la tracción y a la transformación y el transporte de la energía eléctrica, sus edificios anexos, los centros de control de tráfico y cualquier otro que, reglamentariamente, se determinen. [21]

Las estaciones de transporte de viajeros y las terminales de transporte de mercancías, estarán constituidas por:

a) Vías principales y de servicio, junto con los terrenos sobre los que se asientan y todos sus elementos e instalaciones auxiliares necesarias para su funcionamiento.

b) Andenes de viajeros y mercancías.

c) Las calzadas de los patiios de viajeros y mercancías, con los accesos por carretera y para pasajeros que lleguen y salgan a pie.

d) Edificios utilizados por el servicio de infraestructuras.

e) Instalaciones destinadas a la recaudación de las tarifas de transporte, así como las destinadas a atender las necesidades de los viajeros.

No se considerarán estaciones de transporte de viajeros y terminales de transporte de mercancías, las áreas

dedicadas a otras actividades, exclusivamente comerciales, logísticas o industriales, aunque se sitúen en el ámbito de aquellas. [21]

2.1.2. Material Rodante

Se conoce como material rodante a todos los tipos de vehículos dotados de ruedas, capaces de circular sobre una vía férrea, cuyo principal objetivo es transportar diferentes tipos de cargas. Los mismos, se pueden clasificar de muchas formas, aunque los criterios fundamentales para clasificar el material rodante suelen ser su capacidad tractora y su uso comercial.

En se presentan varios aspectos del material rodante tales como: tipos de trenes, ventajas y desventajas de cada uno de ellos, características y partes del material móvil ferroviario y tipos de material móvil remolcado.

Las características del material rodante son, según [1]:

- Ruedas troncocónicas: la inclinación de las generatrices es de 1/20, la misma que la de los carriles. Esto mejora el apoyo de las ruedas sobre los carriles y se ayuda a la inscripción de eje en las curvas, al permitir que cada rueda adopte un radio de contacto distinto para, de esta forma, poder girar a diferente velocidad lineal, pero a iguales revoluciones.
- Ruedas caladas: como ventaja, el calaje confiere al conjunto eje-rueda una mayor robustez, que lo hace muy apropiado para el ferrocarril, donde se mueven grandes cargas a grandes velocidades. Un inconveniente es la problemática de la inscripción de las curvas.
- Pestañas interiores: permiten el guiado del tren.
- Cargas aplicadas sobre la parte exterior de las ruedas: el eje ferroviario sobresale de las ruedas. Este saliente se denomina mangueta. Sobre estas manguetas se coloca la caja del vehículo. Con esta longitud adicional del eje, se obtienen dos ventajas: las cajas de los vehículos pueden ser más anchas y se aumenta la capacidad de los transportes.
- Peso suspendido y no suspendido: el peso suspendido de un vehículo ferroviario es aquel que pasa por la suspensión para llegar al carril, es decir, que está amortiguado. El no suspendido está sin amortiguar. Cuanto mayor sea el peso no suspendido de un vehículo, más agresivo será este con la vía, ya que las cargas dinámicas incidirán sobre ella bruscamente.
- Ruedas debajo de las cajas: permite ampliar la anchura de la caja, ya que no se ve limitada lateralmente por las ruedas, pero penaliza la altura de las mismas. Para ganar altura, las ruedas se fabrican con radio pequeño.
- Material rígido o articulado: se dice que es rígido cuando sus ejes no pueden girar respecto a un eje vertical para mejorar su inscripción en las curvas. La distancia entre 2 ejes fijos se denomina “empate”. Cuanto mayor es el empate de un vehículo, peor será su capacidad de inscripción en las curvas y, por ello, su agresión a la vía y su posibilidad de descarrillar serán mayores. Hoy en día, se utilizan los vehículos articulados, cuyos ejes pueden colocarse en una posición más o menos cercana al radio de curvatura, con lo cual, mejora su inscripción.

Las partes más importantes del material móvil de un sistema ferroviario son, según [1]:

- Caja: en su interior se sitúan los viajeros, la mercancía, los motores, etc., según el tipo de vehículo (coche, vagón o locomotora).
- Bastidor: es la estructura metálica o armazón formada por el bogie, que sirve como elemento de fijación de los ejes, las ruedas, los motores de tracción y las suspensiones, entre otras partes.
- Larguero: elemento longitudinal que forma parte de la estructura del bastidor de un vehículo.
- Traviesas extremas o cabeceros: elemento estructural situado en el extremo del bastidor de un vehículo

que une los largueros de forma perpendicular a estos y que soporta, normalmente, los aparatos de choque y tracción. Al conjunto de elementos que configuran la caja del vehículo sobre la traviesa extrema, se le denomina "Testero".

- Suspensión: la caja transmite las cargas a las ruedas a través de la suspensión. La suspensión ferroviaria es doble: primaria y secundaria.
- Cajas de grasas: las cargas de la caja pasan al bastidor, del bastidor a la suspensión y de esta a las manguetas de los ejes a través de las cajas de grasas. Son unos recipientes metálicos que contienen lubricantes y llevan encajado un rodamiento en el apoyo de las cargas sobre los ejes.
- Rodadura: permite que el vehículo se mueva sobre la vía. Puede estar formada por ejes independientes o bogies. Para el desplazamiento de los vehículos, son necesarios los órganos de rodadura que están compuestos por:
- Eje.
- En algunos vehículos, están instalados discos de freno.
- Bogie.
- Rodamientos.
- Caja de grasa.
- Ruedas.
- Aparatos de tracción y choque: los aparatos de tracción y choque transmiten la fuerza de tracción a lo largo de todo el tren. Pueden ser enganches automáticos, cadenas, barras, etc. Los elementos de choques están formados por 2 topes, situados en el testero del vehículo, tienen la misión de amortiguar las fuerzas longitudinales de compresión que se producen durante la marcha, tanto en las frenadas como en las paradas o los impactos que reciben los vagones en diferentes situaciones, protegiendo así la estructura de los vehículos y las mercancías que transportan.

2.1.3. Depósitos

Un depósito no es una especie de garaje, un lugar donde estacionar y vigilar las locomotoras, a pesar de lo que pueda sugerir su nombre a quién no esté familiarizado con las características de la tracción a vapor, [17].

Como [6] expresaba así las misiones que tenía que cumplir los depósitos, dentro del esquema general de funcionamiento de la compañía ferroviaria:

- El arreglo y distribución del trabajo de maquinistas y fogoneros.
- La regularización de la marcha de máquinas, respecto al servicio que deben prestar y sus condiciones remolcadoras tanto en relación con la sección de vía que recorren, como con las cargas máximas y mínimas que pueden remolcar.
- La conservación y entretenimiento de las máquinas y la distribución de su servicio.
- El acopio y distribución del combustible, aceite y otras materias que precisan para su funcionamiento.

2.2. Planificación de Sistemas Ferroviarios

Tras la definición de las tres partes en las que se puede dividir cualquier sistema ferroviario, se procede a explicar la planificación de estos.

Los sistemas ferroviarios tienen una diferencia con respecto a los demás tipos de transporte público, ya que necesitan de una infraestructura mayor para que se puedan llevar a cabo la prestación de los servicios de manera correcta, [17]. Las operaciones en cualquier comercio y servicio industrial son planeadas, ejecutadas y dirigidas según una jerarquía de cuatro escalones: (i) problemas estratégicos, (ii) problemas tácticos, (iii) problemas

operacionales y (iv) problemas de control a tiempo real. Esta jerarquía está basada en niveles de decisión, horizonte temporal de planificación y ejecución, accionistas, criticidad e impacto de las decisiones y amenazas, [13].

En este apartado, nos centraremos en los problemas que surgen solo en la planificación, de manera que solo se analizarán los problemas estratégicos y los problemas tácticos, referidos en concreto a los sistemas ferroviarios.

2.2.1. Planificación Estratégica

2.2.1.1. Diseño de Red (Network Design)

Según [4], este problema consiste en seleccionar nodos y puntos de encuentro de una potencial, o ya dibujada, red en la que se construirán estaciones y diseñar conexiones entre ellas.

Un trazado desde cero es un problema complicado, en cuyo caso, sería interesante estudiar en profundidad tanto los recursos disponibles para la construcción de esta, como la demanda según las zonas, como los diferentes accidentes geográficos que se pueden encontrar, (ver [17]).

2.2.1.2. Planificación de Líneas

La planificación de líneas, o Line Planning, se presenta como la determinación del origen, itinerario, destino y frecuencia de las líneas empezando desde una red obtenida a partir de un proceso de diseñado de red previo [4].

En el caso de ampliación de la red, existe el interrogante de incluir o no una nueva línea, y la decisión de incluir una nueva línea o de redefinir una ya existente. Habría que definir: longitud de la línea, identificando las estaciones inicial y final y su frecuencia, dependiendo de la tendencia del comportamiento de la demanda.

El problema puede resolverse a través de un modelo matemático, en el que es interesante incluir una matriz Origen-Destino, cuyas componentes guardan el número de pasajeros que se desplazan durante un intervalo de tiempo.

Para definir un conjunto de líneas de la red, se utilizan matrices de demanda diarias. A partir de esta, se utilizan matrices correspondientes a períodos punta y valle para establecer las frecuencias.

2.2.2. Planificación Táctica

2.2.2.1. Scheduling y Timetabling

El problema de Timetabling consiste en establecer un horario predeterminado para cada línea en la red, definida por el punto de salida y los tiempos de llegada a cada estación [16].

Con el objetivo de reducir los tiempos de espera en las zonas de intercambio, las transferencias pueden ser coordinadas, especialmente cuando los headways son amplios, según [7].

Existe, principalmente, dos variantes principales de Timetabling. Una de ellas es el Timetabling periódico, que se repite cada cierto período, por ejemplo, cada hora, con ciertas variaciones en las horas de pico de demanda y en las horas de mucha menos demanda. La otra variante, es el Timetabling no periódico, el cual permite adaptar la frecuencia de los trenes a la demanda de pasajeros. Aun así, en ambos casos los horarios se repiten cada día, aunque sí que hay diferencia entre los horarios de días laborables y los horarios de fin de semana.

Un Timetabling periódico de tipo regular (trenes equiespaciados en el tiempo) encaja más en sistemas de transporte de tránsito más rápidos, donde la frecuencia de cada vehículo es más alta y las estaciones se encuentran entre sí a distancias parecidas. En el caso de transportes con una frecuencia más baja, los horarios periódicos son mejores a la hora de que los pasajeros puedan recordar los horarios más fácilmente. Asumiendo una demanda constante, un Timetabling regular proporciona mínimos tiempos de espera [10].

2.2.2.2. Platforming

Diseñar el routing de un tren en una estación consiste en buscar el camino que tomará este desde el punto de entrada en la estación hasta el punto de salida, pasando, y generalmente parando, por un andén de esta. Este problema es denominado como "Train Platforming Problem" o "TPP" [5].

Según [5], mientras que este problema es muy fácil de resolver en pequeñas estaciones, en las que se cuenta con un número muy pequeño de alternativas para dirigir los trenes, en estaciones de mayor tamaño puede resultar un problema de mucha mayor complejidad.

[12] considera una versión simplificada en la que, para cada vehículo, la planificación de llegadas y salidas no puede cambiar y las rutas de llegada y salida se determinan únicamente por elección de las plataformas. En este caso, también se considera una lista de incompatibilidades, de manera que no cualquier vehículo puede ir a cualquier andén. La misma versión de TPP es estudiada por [2], que muestra cómo incorporar en el modelo restricciones asociadas con la lista de incompatibilidades tren-andén.

2.2.2.3 Material Rodante (Rolling Stock)

El Problema de Gestión de Material Rodante (RSCP) es un problema importante en la operación diaria. Los costes de este problema son bastante considerables, ya que abarca desde los costes de mantenimiento hasta los costes para poner los vehículos en operación con electricidad o Diesel. Todos estos gastos dependen del número de kilómetros recorridos por cada unidad. Por ello, es muy importante decidir cuidadosamente, el tipo y el número de unidades de material rodante, como se comenta en [5].

A la hora de tener en cuenta la demanda y la capacidad de los sistemas ferroviarios, hay que considerar el posible acoplamiento y desacoplamiento de vagones en los trenes, a lo largo de sus desplazamientos sobre la red. Por ello, es importante disponer de una localización para almacenar vagones y un sistema de ágil para realizar las operaciones con vagones, garantizando unos tiempos adecuados de operación. [17]

Existe varias versiones de RSCP, dependiendo del equipamiento usado y la naturaleza de la red del metro. Respecto al equipamiento, distinguimos dos casos, [5]:

i. Locomotoras y vagones independientes.
ii. Conjunto Locomotoras-Vagones predefinidas.

Respecto al segundo, estos conjuntos se componen de un número de vagones en una composición fija y dado que disponen de dos cabezas tractoras pueden moverse en ambos sentidos de una línea. Un ejemplo de este tipo de sistema es el conocido como French Train à Grande Vitesse (TGV) el cual consiste en un conjunto formado por seis a diez vagones y dos locomotoras al principio y al final del conjunto, [8].

En el primer caso, para cada trayecto dentro del horizonte temporal se debe determinar, para el tren asociado, el tipo y número de locomotora y el tipo y número de vagones. Estos no son independientes el uno del otro, sino que el número de vagones determina el número y el tipo de locomotora para proporcionar la suficiente capacidad motriz. Girar un tren con locomotora en el punto final de una línea es bastante complejo, ya que la(s) locomotora(s) debe(n) ser desacoplada(s) y conducida(s) al otro lado del tren, donde debe(n) ser acoplada(s) de nuevo. También puede ocurrir que se disponga de una locomotora de reserva para el viaje de vuelta, o que todo el tren dé la vuelta, si se dispone de las llamadas vías en Y, como observa [11].

Por otro lado, hay que tener en cuenta las situaciones según el tipo de red que pueden ser, como comenta [3]:

1. Redes de baja densidad, normalmente con distancias de trayectos largos y tiempos de viaje altos.
2. Redes de alta densidad, totalmente contrarios a los otros, de trayectos cortos.

2.2.2.4. Programación de turnos de tripulaciones (Crew Management)

Tras la planificación de servicios a través de la planificación de material rodante, plataformas, etc., se aborda el problema de planificación de los turnos de los operarios, tripulación y conductores de manera que se puedan realizar todos los desplazamientos.

[5] observa que el objetivo de la planificación de tripulaciones consiste en establecer una relación entre las necesidades de personal de los servicios a prestar y los recursos disponibles. La necesidad depende del Timetabling y de la circulación de Rolling Stock, pero también de los acuerdos de los convenios que tengan los trabajadores con la organización, y puede estar influenciada también por el desplazamiento de cierta cantidad de trabajo de un terminal a otro.

Hay que tener en cuenta la poca homogeneidad que puede haber en los turnos debido a la variabilidad de la demanda. Por tanto, la planificación de turnos conlleva dar a los trabajadores información de lugar y hora de inicio, línea, vehículo, rotaciones y lugar y hora de fin del turno de trabajo diario.

2.2.2.5. Asignación de tripulaciones

Este problema, también conocido como Crew Planing Problem (CPP), consiste en construir los calendarios de trabajo de las tripulaciones necesarias para cubrir un horario planificado. En el CPP, se obtiene un horario planificado para los servicios ferroviarios, tanto en trayectos reales con pasajeros o mercancías, como los transbordos de trenes vacíos o de material entre estaciones, que deben realizarse cada cierto tiempo dentro del horizonte de planificación. Cada servicio se ha dividido primero en una secuencia de viajes, definidos como segmentos de trayectos ferroviarios que deben ser atendidos por la misma tripulación. Cada viaje se caracteriza por una hora y una estación de salida y una hora y una estación de llegada. [5]

2.2.3. Planificación Operacional

2.2.3.1. Re-Scheduling

Muchos autores consideran “Scheduling” como una operación estratégica o táctica, y “Re-Scheduling” como un control operativo o en tiempo real, según observa [13].

“Re-scheduling” consiste en ajustar la planificación semanal realizada en la etapa anterior, en la etapa táctica, basándonos en disponibilidad o no de ciertas vías por mantenimiento, eventos deportivos o de otra índole, fiestas locales o nacionales, etc. y ajustándola a la demanda semanal, [3].

2.2.3.2. Rolling Routing

Como se puede encontrar en [17], en esta sección, se distinguen dos partes:

1. La circulación operacional.
2. El plan de movimientos operacionales.

Los elementos más importantes son el direccionado de los trenes a través de las estaciones de la red, también conocido como “Routing”, y el posicionamiento de las unidades no usadas temporalmente en zonas de estacionamiento (donde también se realizan operaciones de acoplamiento y desacoplamiento), también llamadas “Shuting Yards”.

El proceso de movimientos internos en estas zonas, “Shuting Process”, tendrá lugar en las proximidades de las estaciones principales y lo llevarán a cabo planificadores locales.

2.2.3.3. Crew Re-Rostering

Por último, falta organizar los turnos de conductores y revisores, basándose en las listas creadas en la planificación táctica, de manera que se pueda respetar el número de horas y las características de los turnos de trabajo de los tripulantes, según la antigüedad en su actividad profesional. Finalmente se puede observar que, dada su complejidad, a efectos prácticos la planificación operacional puede ser considerada muchas veces como un problema de fiabilidad [3].

3 DESCRIPCIÓN DEL CASO A DESARROLLAR

Este estudio trata una línea de metro constituida por estaciones, terminales y tramos. En la Ilustración 1 se muestra la estructura de la línea de metro de la que se parte para realizar el estudio. Como se observa, podemos encontrar 2 terminales, al principio y al final de la línea, N estaciones, con 2 andenes en cada estación y N-1 tramos, uno entre cada par de estaciones.

Dentro del conjunto de estaciones, se pueden diferenciar dos subconjuntos, que corresponden a estaciones que denominaremos mayores y menores. En las estaciones mayores y las terminales los trenes pueden realizar un cambio de sentido, de manera que, al finalizar el servicio que estaban realizando pueden ser asignados a un nuevo servicio en la dirección contraria.

En la Ilustración 1, las terminales están representadas por cuadrados, las estaciones mayores, donde también se permite realizar el cambio de sentido están representadas por triángulos con relleno y las estaciones menores por triángulos sin relleno.

Figura 1 Mapa físico de la línea

A la hora de tratar los problemas de la planificación de horarios y gestión del material rodante de una línea de metro, la condición de bidireccionalidad de la línea y las operaciones de cambio de sentido, son características muy importantes que deben ser tenidas en cuenta, ya que condicionan de forma importante la dificultad de llevar a cabo una gestión eficiente y real. Para las operaciones diarias de planificación de horarios de líneas de metro es conveniente usar mapas de operación, donde no solo se representa las distintas estaciones como en la Ilustración 1, sino que también se tienen en cuenta los andenes en ambos sentidos, como se muestra en la Ilustración 2.

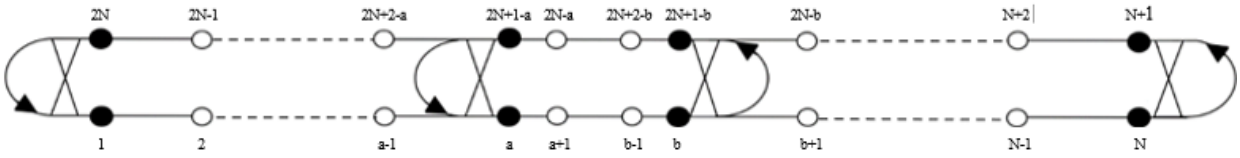

Figura 2 Mapa operacional de una línea de metro

Como podemos observar en la figura anterior, cada estación se divide en dos andenes, de manera que, por ejemplo, a la estación $i-ésima$ de la Ilustración 1 le correspondería la pareja de andenes i y $2N-i+1$ del mapa de operación de la Ilustración 2. De este modo quedan definidos los dos sentidos, es decir, del andén 1 al N se define el sentido aguas arriba, y del andén $N+1$ al $2N$ el sentido contrario, denominado aguas abajo.

Los servicios pueden ser de dos tipos, servicio de recorrido completo, si se realiza el servicio entre los andenes 1 al N o entre los andenes N + 1 al $2N$, y servicio de recorrido corto si la estación inicial o final del servicio realizado no es una de las terminales, sino alguna de las estaciones mayores intermedias (o dos estaciones mayores intermedias). La diferencia entre ambos tipos de servicio se puede ver en las figuras 3a y 3b. En la Ilustración 4 se observa cómo, algunos de los servicios (en color rojo) se realizan entre la segunda estación y la estación número 4, por el contrario, en la Figura 3b, todos los servicios transcurren entre las estaciones terminales

de la línea. Formalmente las estaciones inicial y final de la línea también serán estaciones mayores, además son las únicas que se encuentran conectadas a los depósitos, lugares de donde parten los trenes y a los que se envían los trenes una vez han terminado su actividad. Todos los servicios tienen que tener, como estaciones de comienzo y fin, una estación mayor donde se pueda realizar un cambio de sentido, independientemente del tipo de servicio, ya sea servicio de recorrido completo o servicio de recorrido incompleto.

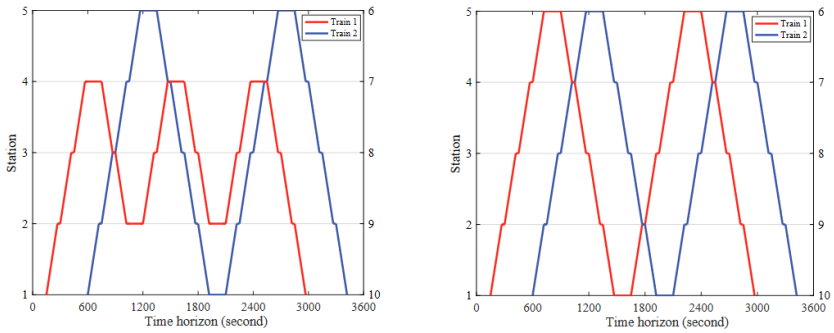

Figura 3 Con y sin Short-Turning

Si todos los servicios fueran de recorrido completo, todas las estaciones tendrían el mismo nivel de servicio, sin embargo, al utilizar servicios de recorrido incompleto, hay ciertas estaciones a las que llegarán más servicios que otras. Por ejemplo, en el caso expuesto en la ilustración 3, la estación 2 tiene una mayor frecuencia de llegada de servicios, lo que permite tender una mayor afluencia de pasajeros. Así, utilizando un diseño adecuado de servicios incompletos, la demanda desigual entre distintas estaciones se puede tratar de una forma adecuada. De esta forma, se consigue disminuir de manera considerable el tiempo total de espera de los pasajeros.

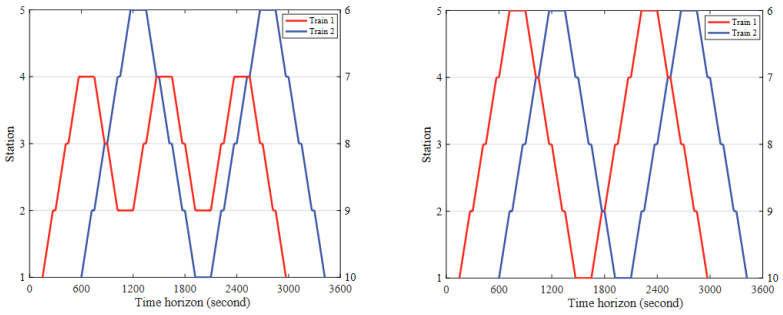

Figura 4 Con y sin usar Short-Turning

Para concluir este punto, a la hora de modelar el problema y con el objetivo de facilitar la resolución, se han asumido las siguientes 5 suposiciones previas:

1. Se consideran dos estaciones terminales, cada una situada a un extremo de la línea de metro, es decir, al principio y al final de esta.
2. No hay adelantamiento ni encuentro entre trenes, ni en las estaciones, ni a lo largo de los trayectos entre estaciones.
3. Los pasajeros obedecen el principio FIFO (First-In-First-Out), es decir, el orden de embarque de los pasajeros se supone igual al orden de llegada de éstos a las diferentes plataformas.
4. Los tiempos de recorrido entre estaciones, así como los tiempos de parada de los trenes en cada estación, se consideran datos predefinidos del problema. Resulta muy sencillo modificar esta consideración, sin embargo, dada la dificultad y tamaño de los modelos resultantes, se ha optado en este caso por fijar ambos.
5. Los pasajeros no realizan transbordos innecesarios entre servicios de la misma línea, es decir, solo

pueden usar los servicios que les lleven de forma directa a la estación a la que desean llegar. De esta forma, se eliminan comportamientos espúreos que no obedecen al principio racional de realizar el trayecto en el menor tiempo posible.

4 MODELADO DEL PROBLEMA

4.1.Variables y datos.

A continuación, vamos a definir los datos y variables del problema que serán leídas desde un libro de Microsoft Excel:

Notación	Conjuntos y parámetros
$S^{(u)}$	Conjunto de estaciones en sentido aguas arriba.
$S^{(d)}$	Conjunto de estaciones en sentido aguas abajo.
$S_M^{(u)}$	Conjunto de estaciones mayores (con cambio de sentido) en sentido aguas arriba.
$S_M^{(d)}$	Conjunto de estaciones mayores (con cambio de sentido) en sentido aguas abajo.
$K^{(u)}$	Conjunto de servicios potenciales del horario, en sentido aguas arriba.
$K^{(d)}$	Conjunto de servicios potenciales para ser elegidos, o no, en sentido aguas abajo.
i, j, m, n	Índices de estación.
k, l	Índices de servicios.
N	Número total de estaciones.
h_{max}	Headway máximo entre dos servicios consecutivos. El headway representa la separación temporal entre las salidas de dos servicios consecutivos en cualquiera de las estaciones. Valores muy altos representan frecuencias bajas y por lo tanto una menor calidad de servicio a los viajeros. Valores de headway muy pequeños, mejoran la calidad de servicio, pero exigen un mayor número de trenes para ejecutar el horario, aumentando los costes para las compañías operadoras.
h_{min}	Headway mínimo entre dos servicios consecutivos. El headway mínimo suele venir condicionado por los sistemas de señalización instalados a lo largo de la vía a fin de mantener la seguridad y evitar posibles colisiones (alcances) entre trenes.
$s_{i-1,i}$	Tiempo predeterminado de recorrido de un servicio entre las estaciones *i-1* e *i*.
e_i	Tiempo predeterminado de parada en la estación i (tiempo requerido para la bajad y embarque de pasajeros).
δ_m^{max}	Máximo tiempo predeterminado de cambio de sentido en la estación m.
δ_m^{min}	Mínimo tiempo predeterminado de cambio de sentido en la estación m.
RS	Número máximo de trenes disponibles para realización del horario en la línea de metro.
$OD_{i,j}[t]$	Elemento de la matriz de demanda de pasajeros que corresponde al origen en la estación i y destino en la estación j, en el instante t.
C	Capacidad máxima de un tren.
M	Número muy grande.

Tabla 1. Variables Definidas

En la siguiente tabla, se definen las variables de decisión que se utilizan a la hora de modelar el problema:

Notación	Definición de las Variables
τ_k	Variable binaria que vale 1 si se activa el servicio k y 0 en otro caso.
$z_{k,m,n}$	Variable binaria que vale 1 si las estaciones mayores, con cambio de sentido, m y n son el principio y final, respectivamente, del servicio k, y 0 en otro caso.
$\lambda_{k,i}$	Variable binaria que vale 1 si el servicio k para en la estación i, y 0 en otro caso.
$h_{k-1,k}$	Variable que define el headway que existe entre dos servicios consecutivos, $k-1\ y\ k$.
$a_{k,i}$	Instante de llegada del servicio k a la estación i.
$d_{k,i}$	Instante de salida del servicio k a la estación i.
$\gamma_{k,l,m}$	Variable binaria que vale 1 si un tren, tras terminar el servicio k en la estación m, hace un cambio de sentido para realizar el servicio l en el sentido contrario.
$\alpha_k^{(u)}$	Variable binaria que vale 1 si el servicio en sentido aguas arriba, k, es llevado a cabo por un tren que viene del depósito 1, y 0 en otro caso. El depósito 1 se considera situado antes de la estación inicial aguas arriba.
$\alpha_k^{(d)}$	Variable binaria que vale 1 si el servicio en sentido aguas abajo, k, es llevado a cabo por un tren que viene del depósito 2, y 0 en otro caso. El depósito 2 se considera situado detrás de la última estación aguas arriba (antes de la primera plataforma en dirección aguas abajo).
$\beta_k^{(u)}$	Variable binaria que vale 1 si un tren, tras terminar el servicio k en sentido aguas arriba, vuelve al depósito 2, y 0 en otro caso.
$\beta_k^{(d)}$	Variable binaria que vale 1 si un tren, tras terminar el servicio k en sentido aguas abajo, vuelve al depósito 1, y 0 en otro caso.
$\omega_{k,i}$	Número de pasajeros esperando al servicio k en la estación i.
$\omega_{k,i,j}$	Número de pasajeros esperando al servicio k en la estación i, con destino la estación j.
$\omega_{k,i}^b$	Número de pasajeros que se encuentran con la posibilidad de subir al tren que realiza el servicio k en la estación i. La diferencia con respecto a la variable $\omega_{k,i}$ es sutil. $\omega_{k,i}$ representan los pasajeros que están esperando, pero si el servicio k no se detiene en la estación i estos no podrán subir. $\omega_{k,i}^b$ mide los pasajeros que pueden subir al tren, que podrá coincidir con $\omega_{k,i}$ si el servicio se detiene en la estación $i-ésima$ y existe capacidad suficiente para permitir el embarque de todos los pasajeros.
$\omega_{k,i,j}^b$	Número de pasajeros que se encuentran con la posibilidad de subir al tren que realiza el servicio k en la estación i, con destino la estación j.
$c_{k,i}^b$	Número de pasajeros que realmente embarcan en el tren que hace el servicio k (en cualquier dirección) en la estación i.
$c_{k,i,j}^b$	Número de pasajeros que cogen el servicio k en la estación i, con destino la estación j.
$c_{k,i}^a$	Número de pasajeros que se bajan del servicio k en la estación i.
$c_{k,i}$	Número de pasajeros en el servicio k en la estación i.
$\omega_{k,i,j}^r$	Número de pasajeros restantes que no pueden coger el servicio k en la estación i con destino la estación j.

Tabla 2. Variables de Decisión

4.2. Restricciones

4.2.1. Selección de las zonas de operación.

Es importante hacer una buena selección de las zonas de operación de los servicios, marcadas por las estaciones inicial y final, para que la estrategia de cambio de sentido (Short turning) sea efectiva y conseguir que el horario se adapte bien a la demanda desigual de pasajeros en las diferentes estaciones.

Si un servicio, en sentido aguas arriba, $k \in K^{(u)}$ se activa, existirán dos estaciones, y solo dos estaciones, $m, n \in S_M^{(u)}$ tales que $z_{k,m,n} = \tau_k = 1$. En el caso de que ese servicio potencial no active, $z_{k,m,n} = \tau_k = 0$.

$$\sum_{m \in S_M^{(u)}} \sum_{n \in S_M^{(u)}} z_{k,m,n} = \tau_k \quad \forall k \in \mathbf{K}^{(u)} \tag{1}$$

El índice de la estación final del recorrido del servicio debe ser mayor de que el índice de la estación inicial de este, de manera que:

$$z_{k,m,n} = 0 \quad \forall k \in \mathbf{K}^{(u)}, \forall m, n \in \mathbf{S}_M^{(u)}, \ m \geq n \tag{2}$$

Estas mismas restricciones, las replicamos para los servicios en sentido aguas abajo, $k \in K^{(d)}$, y para las estaciones en este mismo sentido, $m, n \in S_M^{(u)}$.

$$\sum_{m \in S_M^{(d)}} \sum_{n \in S_M^{(d)}} z_{k,m,n} = \tau_k \quad \forall k \in \mathbf{K}^{(d)} \tag{3}$$

$$z_{k,m,n} = 0 \quad \forall k \in \mathbf{K}^{(d)}, \forall m, n \in \mathbf{S}_M^{(d)}, \ m \geq n \tag{4}$$

4.2.2. Estaciones a las que llegan los servicios.

Si el servicio en sentido aguas arriba, $k \in \mathbf{K}^{(u)}$, no se activa, no puede detenerse en la estación i, por lo que $\lambda_{k,i} = 0, \ i \in \boldsymbol{S}^{(u)}$. Para conseguir este comportamiento, se usarán las siguientes restricciones:

$$z_{k,m,n} = 0 \quad \forall k \in \mathbf{K}^{(u)}, \forall m, n \in \mathbf{S}_M^{(u)}, \ m \geq n \tag{5}$$

Si el servicio $k \in \mathbf{K}^{(u)}$ se activa y tiene inicio en la estación mayor $m \in \mathbf{S}_M^{(u)}$ y fin en la estación menor $n \in \mathbf{S}_M^{(u)}$, este servicio no puede servir a estaciones anteriores a m ni posteriores a n. Por ello:

$$\sum_{i<m} \lambda_{k,i} \leq M \cdot \left(1 - \sum_{n \in \boldsymbol{S}_M^{(u)}} z_{k,m,n}\right) \quad \forall k \in \mathbf{K}^{(u)}, \forall\, m \in \mathbf{S}_M^{(u)}, \forall i \in \boldsymbol{S}^{(u)} \tag{6}$$

$$\sum_{i>n} \lambda_{k,i} \leq M \cdot \left(1 - \sum_{m \in \boldsymbol{S}_M^{(u)}} z_{k,m,n}\right) \quad \forall k \in \mathbf{K}^{(u)}, \forall n \in \mathbf{S}_M^{(u)}, \forall i \in \boldsymbol{S}^{(u)} \tag{7}$$

Pero sí debe servir todas y cada una de las estaciones que se han seleccionado dentro de su zona de operación, por ello debemos incluir las siguientes restricciones:

$$\lambda_{k,i} \geq z_{k,m,n} \quad \forall k \in \mathbf{K}^{(u)}, \forall m,n \in \mathbf{S}_M^{(u)}, i \in \boldsymbol{S}^{(u)}, n \leq i \leq m \tag{8}$$

Del mismo modo, debemos incluir las restricciones correspondientes para el sentido contrario, es decir, aguas abajo:

$$\lambda_{k,i} \leq \tau_k \quad \forall k \in \mathbf{K}^{(d)}, i \in \boldsymbol{S}^{(d)} \tag{9}$$

$$\sum_{i<m} \lambda_{k,i} \leq M \cdot \left(1 - \sum_{n \in \boldsymbol{S}_M^{(d)}} z_{k,m,n}\right) \quad \forall k \in \mathbf{K}^{(d)}, \forall m \in \mathbf{S}_M^{(d)}, i \in \boldsymbol{S}^{(d)} \tag{10}$$

$$\sum_{i>n} \lambda_{k,i} \leq M \cdot \left(1 - \sum_{m \in \boldsymbol{S}_M^{(d)}} z_{k,m,n}\right) \quad \forall k \in \mathbf{K}^{(d)}, \forall n \in \mathbf{S}_M^{(d)}, i \in \boldsymbol{S}^{(d)} \tag{11}$$

$$\lambda_{k,i} \geq z_{k,m,n} \quad \forall k \in \mathbf{K}^{(d)}, \ \forall m,n \in \mathbf{S}_M^{(d)}, \ i \in \boldsymbol{S}^{(d)}, \ n \leq i \leq m \tag{12}$$

Las siguientes restricciones son, a priori, opcionales. Su inclusión fuerza que en todas las estaciones exista cierta frecuencia mínima, y por tanto, cierta calidad de servicio. El grupo de restricciones (13) obligan al modelo a que atienda cada estación con al menos un servicio de cada dos consecutivos, consiguiendo así que no se desplome la calidad de servicio en algunas estaciones.

$$\lambda_{k,i-1} + \lambda_{k,i} \geq 1 \quad \forall k-1, k \in \mathbf{K}^{(u)}, i \in \boldsymbol{S}^{(u)} \tag{13}$$

Claro está que será preciso modelar la restricción en ambos sentidos, de manera que se incluyen las respectivas restricciones para el sentido aguas abajo.

$$\lambda_{k,i-1} + \lambda_{k,i} \geq 1 \quad \forall k-1, k \in \mathbf{K}^{(d)}, i \in \boldsymbol{S}^{(d)} \tag{14}$$

4.2.3. Horas de llegada y salida.

Se ha supuesto que los tiempos que los trenes tardan en recorrer los segmentos que unen distintas estaciones, así como los tiempos de estancia en ellas, están predefinidos. Por ello, los instantes de salida y llegada de los servicios potenciales, deben satisfacer las siguientes condiciones:

$$a_{k,i} - d_{k,i-1} = s_{i-1,i} \quad \forall k \in \mathbf{K}^{(u)}, \forall i-1, i \in \mathbf{S}^{(u)} \tag{15}$$

$$a_{k,i} - d_{k,i-1} = s_{i-1,i} \quad \forall k \in \mathbf{K}^{(d)}, \forall i-1, i \in \mathbf{S}^{(d)} \tag{16}$$

$$d_{k,i} - a_{k,i} = e_i \quad \forall k \in \mathbf{K}^{(u)}, \forall i \in \mathbf{S}^{(u)} \tag{17}$$

$$d_{k,i} - a_{k,i} = e_i \quad \forall k \in \mathbf{K}^{(d)}, \forall i \in \mathbf{S}^{(d)} \tag{18}$$

Obviamente, es preciso modelar las restricciones en ambos sentidos, aguas arriba y aguas abajo.

Merece la pena resaltar que estas cuatro últimas restricciones tienen en cuenta las horas de salida, llegada y tiempos de estancia de los servicios potenciales en todas las estaciones, sin embargo, puede que ciertos servicios potenciales no sean seleccionados, $\tau_k = 0$, o que a un servicio potencial se le asigne un recorrido corto-reversible, de manera que, en estos casos, los servicios no llegarán a algunas de las estaciones. Dado que a priori se desconoce la asignación de zonas y si los servicios de recorrido completo se deben detener o no en algunas de las estaciones, es necesario su inclusión en el modelo. Además, si $\tau_k = 1$ pero se le asigna una zona a un servicio, las estaciones fuera de esta zona no pueden tener tiempo de parada. Como veremos esto queda modelado posteriormente.

4.2.4. Headway entre servicios consecutivos.

Con el objetivo de garantizar la seguridad de los servicios, así como un cierto nivel de calidad de servicio, se utilizarán unos tiempos predeterminados máximo y mínimo para el headway entre servicios consecutivos.

Puesto que no todos los servicios potenciales se activarán, hay que asegurar el comportamiento que se describe a continuación:

$$\begin{cases} h_{min} \leq h_{k-1,k} \leq h_{max} & si\ \tau_k = 1 \\ h_{k-1,k} = 0 & si\ \tau_k = 0 \end{cases}$$

Para conseguir dicho resultado, basta con realizar el producto entre los límites inferior y superior del headway y la variable y τ_k, de manera que, si $\tau_k = 1$, el headway queda acotado entre sus límites, pero si $\tau_k = 0$, ambos extremos valen 0, obligando así que el headway entre ambos servicios sea 0.

$$h_{min} \cdot \tau_k \leq h_{k-1,k} \quad \forall k-1, k \in \mathbf{K}^{(u)} \tag{19.1}$$

$$h_{k-1,k} \leq h_{max} \cdot \tau_k \quad \forall k-1, k \in \mathbf{K}^{(u)} \tag{19.2}$$

Y como en todas las restricciones, repetimos las mismas para el sentido contrario, es decir, aguas abajo.

$$h_{min} \cdot \tau_k \leq h_{k-1,k} \quad \forall k-1, k \in \mathbf{K}^{(d)} \tag{20.1}$$

$$h_{k-1,k} \leq h_{max} \cdot \tau_k \quad \forall k-1, k \in \mathbf{K}^{(d)} \tag{20.2}$$

Una vez definido el headway entre servicios y utilizando también la hora de partida del servicio anterior, $k-1$, de la estación i, se puede calcular el instante de salida del servicio k de la estación i, de la siguiente manera.

$$d_{k,i} = d_{k-1,i} + h_{k-1,k} \quad \forall \mathrm{k}-1, \mathrm{k} \in \mathbf{K}^{(u)}, \forall i \in \mathbf{S}^{(u)} \tag{21}$$

Como se viene repitiendo a lo largo de toda la explicación del modelado del problema, se formula la misma restricción para el sentido aguas abajo.

$$d_{k,i} = d_{k-1,i} + h_{k-1,k} \quad \forall \mathrm{k}-1, \mathrm{k} \in \mathbf{K}^{(d)}, \forall i \in \mathbf{S}^{(d)} \tag{22}$$

4.2.5. Cambio de sentido.

Los cambios de sentido se realizan en las que denominamos estaciones mayores, que son aquellas que tienen los recursos para que el tren, una vez finalizado el servicio en curso, cambie de sentido. De este modo, el inicio y final de todos los servicios se realizan en estaciones mayores.

Teniendo en cuenta lo anterior, cuando un tren termina el servicio k en un sentido, en la estación m, y realiza una operación de cambio de sentido para llevar a cabo el servicio l, en el sentido contrario, comenzará en la estación, o andén, n que es el parejo a la estación final del servicio k, es decir, es el parejo a m como se puede ver observar en la Ilustración 2.

En términos de índices, podemos decir que tenemos *N* estaciones físicas, mientras que tenemos $2N+1$ andenes, de manera que $m + n = 2N + 1$. Acorde a la definición de $\gamma_{k,l,m}$, las siguientes restricciones se deben satisfacer:

$$2 \cdot \gamma_{k,l,m} \leq \sum_{i \in S_M^{(u)}} z_{k,i,m} + \sum_{i \in S_M^{(d)}} z_{l,2N+1-m,i} \quad \forall k \in \mathbf{K}^{(u)}, \forall l \in \mathbf{K}^{(d)}, \forall m \in \mathbf{S}_M^{(u)} \qquad (23)$$

Este conjunto de restricciones obliga a que si un tren termina el servicio k en sentido aguas arriba, en la estación m, y gira para realizar el servicio l en el sentido contrario, $\gamma_{k,l,m} = 1$, de manera que la diferencia entre el instante de llegada del servicio k a la estación m y el instante de salida del servicio l de la estación $2N - m + 1$ debe quedar acotada entre unos valores razonables. (Debe ser mayor que un valor mínimo δ_m^{min} necesario para realizar la maniobra y debe ser menor a un valor máximo δ_m^{max} para no entorpecer el movimiento de otros trenes.). Estos valores se consideran datos del problema. A modo orientativo, δ_m^{min} podrá tomar valores de uno o dos minutos y δ_m^{max} no debería ser mucho mayor que el headway mínimo entre servicios.

$$a_{l,2N+1-m} - \delta_{k,m} \leq \delta_m^{min} + M \cdot (\gamma_{k,l,m} - 1) \quad \forall k \in \mathbf{K}^{(u)}, \forall l \in \mathbf{K}^{(d)}, \forall m \in \mathbf{S}_M^{(u)} \qquad (24)$$

$$a_{l,2N+1-m} - \delta_{k,m} \leq \delta_m^{max} - M \cdot (\gamma_{k,l,m} - 1) \quad \forall k \in \mathbf{K}^{(u)}, \forall l \in \mathbf{K}^{(d)}, \forall m \in \mathbf{S}_M^{(u)} \qquad (25)$$

Similarmente a lo anterior, se adecuan las anteriores restricciones al caso en el que un servicio k actúa en sentido aguas abajo.

$$2 \cdot \gamma_{k,l,m} \leq \sum_{i \in S_M^{(d)}} z_{k,i,m} + \sum_{i \in S_M^{(u)}} z_{l,2N+1-m,i} \quad \forall k \in \mathbf{K}^{(d)}, \forall l \in \mathbf{K}^{(u)}, \forall m \in \mathbf{S}_M^{(d)} \qquad (26)$$

$$a_{l,2N+1-m} - \delta_{k,m} \leq \delta_m^{min} + M \cdot (\gamma_{k,l,m} - 1) \quad \forall k \in \mathbf{K}^{(d)}, \forall l \in \mathbf{K}^{(u)}, \forall m \in \mathbf{S}_M^{(d)} \qquad (27)$$

$$a_{l,2N+1-m} - \delta_{k,m} \leq \delta_m^{max} - M \cdot (\gamma_{k,l,m} - 1) \quad \forall k \in \mathbf{K}^{(d)}, \forall l \in \mathbf{K}^{(u)}, \forall m \in \mathbf{S}_M^{(d)} \qquad (28)$$

4.2.6. Material rodante.

El modelo se ha diseñado de manera que, el servicio k, en sentido aguas arriba, se realizará por un tren que viene del depósito 1 o por un tren que acaba de terminar un servicio en sentido aguas abajo. Por ello, se define la variable $\alpha_k^{(u)}$, que vale 1 en el caso de que el servicio k se realice por un tren que viene del depósito 1, y 0 si el tren que realiza dicho servicio acaba de terminar un servicio en sentido aguas abajo. Por otro lado, definimos la variable complementaria para el caso de los servicios en sentido aguas abajo. Esto es $\alpha_k^{(d)}$, que valdrá 1 si el servicio k, en sentido aguas abajo, es realizado por un tren que viene del depósito 2, y 0 en otro caso. Por ello, estas variables deberán satisfacer las siguientes restricciones.

$$\sum_{m \in S_M^{(d)}} \sum_{l \epsilon K^{(d)}} \gamma_{k,l,m} + \alpha_k^{(u)} = \tau_k \quad \forall k \in \mathbf{K}^{(\mathrm{u})} \qquad (29)$$

$$\sum_{m \in S_M^{(u)}} \sum_{l \epsilon K^{(u)}} \gamma_{k,l,m} + \alpha_k^{(d)} = \tau_k \quad \forall k \in \mathbf{K}^{(\mathrm{d})} \qquad (30)$$

Cuando un tren termina un servicio con sentido aguas arriba, debe ir hasta el depósito 2(al final del sentido aguas arriba) o dar la vuelta para realizar un servicio en el sentido contrario. Del mismo modo, cuando un tren realiza

un servicio en sentido aguas abajo, debe ir al depósito 1(en la última estación de ese sentido) o hacer un cambio de sentido para realizar un servicio en sentido aguas arriba.

Para ello, se ha definido las variables $\beta_k^{(u)}$con $k \in \mathbf{K}^{(u)}$ y $\beta_k^{(d)}$con $k \in \mathbf{K}^{(d)}$, de manera que, en el caso del sentido aguas arriba, si el tren, tras terminar el servicio k, viaja al depósito 2, $\beta_k^{(u)} = 1$, y $\beta_k^{(u)} = 0$, en otro caso, es decir, si hace un cambio de sentido para realizar otro servicio. Lo mismo pasa con $\beta_k^{(d)}$, que vale 1 si tras terminar el servicio k, va al depósito 1, y vale 0 en otro caso. De este modo, junto con las variables $\gamma_{k,i,m}$ y τ_k, se deben satisfacer las siguientes restricciones:

$$\sum_{m \in S_M^{(u)}} \sum_{l \epsilon K^{(d)}} \gamma_{k,l,m} + \beta_k^{(u)} = \tau_k \quad \forall k \in \mathbf{K}^{(\mathrm{u})} \tag{31}$$

$$\sum_{m \in S_M^{(d)}} \sum_{l \epsilon K^{(u)}} \gamma_{k,l,m} + \beta_k^{(d)} = \tau_k \quad \forall k \in \mathbf{K}^{(\mathrm{d})} \tag{32}$$

En el modelo, como se puede observar en la ilustración 2, contamos con dos depósitos, localizados en los extremos de la línea, donde están todos los trenes que no están realizando ningún servicio. Para saber el número de trenes que están en dichos depósitos, nos apoyamos en las variables RS^1 y RS^2, para los depósitos 1 y 2 respectivamente. De acuerdo con la definición de $\alpha_k^{(u)}$, la siguiente restricción debe cumplirse:

$$RS^1 \geq \sum_{k \in \boldsymbol{K}^{(\mathrm{u})}} \alpha_k^{(u)} \quad \forall k \in \mathbf{K}^{(\mathrm{u})} \tag{33}$$

Para que el número de trenes que vienen del depósito 1 a realizar el servicio k no sea mayor que el número de trenes que hay en el depósito 1. De manera similar, tenemos una restricción para el depósito 2.

$$RS^2 \geq \sum_{k \in \boldsymbol{K}^{(\mathrm{d})}} \alpha_k^{(d)} \quad \forall k \in \mathbf{K}^{(\mathrm{d})} \tag{34}$$

Además, debemos asegurarnos de que el número de trenes en los depósitos 1 y 2, no sea mayor que el número de trenes totales.

$$RS^1 + RS^2 \leq RS \tag{35}$$

También podemos evitar que los trenes que realizan movimientos en zonas internas de la línea vuelvan vacíos a los depósitos (recorriendo estaciones fuera de la zona que les sea asignada), para lograr este comportamiento se incluyen las siguientes restricciones.

$$\alpha_k^{(u)} \geq \sum_{n \in S_M^{(u)}} z_{k,1,n} \quad \forall k \in \mathbf{K}^{(u)} \tag{36}$$

$$\beta_k^{(u)} \geq \sum_{n \in S_M^{(u)}} z_{k,n,N} \quad \forall k \in \mathbf{K}^{(u)} \tag{37}$$

De manera similar, los servicios en sentido aguas abajo deben satisfacer las siguientes restricciones.

$$\alpha_k^{(d)} \geq \sum_{n \in S_M^{(d)}} z_{k,N+1,n} \quad \forall k \in \mathbf{K}^{(d)} \tag{38}$$

$$\beta_k^{(d)} \geq \sum_{n \in S_M^{(d)}} z_{k,n,2N} \quad \forall k \in \mathbf{K}^{(d)} \tag{39}$$

4.2.7. Demanda de los pasajeros.

En esta sección, se explica la forma en que se ha modelado el comportamiento dela demanda de los pasajeros.

Debido al doble sentido, la demanda en ambos sentidos es independiente, de manera que se van a explicar las restricciones en el sentido aguas arriba, y posteriormente se mostrarán las restricciones en el sentido aguas abajo.

En primer lugar, basándonos en la definición de $\omega_{k,i}$ y $\omega_{k,i,j}$, la siguiente restricción se debe cumplir:

$$\omega_{k,i} = \sum_{j=i+1}^{N} \omega_{k,i,j} \quad \forall k \in \mathbf{K}^{(u)}, \forall i \in \mathbf{S}^{(u)} \tag{40}$$

En un principio, se asume que los pasajeros deben usar el servicio que les lleve a su destino, de manera que no se permite la transferencia de pasajeros de un servicio a otro. De este modo, para $\omega_{k,i}^b, \omega_{k,i,j}^b, c_{k,i}^b$ y $c_{k,i,j}^b$, las siguientes restricciones deben satisfacerse:

$$\omega_{k,i}^b = \sum_{j=i+1}^{N} \omega_{k,i,j}^b \quad \forall k \in \mathbf{K}^{(u)}, \forall\, i \in \mathbf{S}^{(u)} \tag{41}$$

$$c_{k,i}^b = \sum_{j=i+1}^{N} c_{k,i,j}^b \quad \forall k \in \mathbf{K}^{(u)}, \forall i \in \mathbf{S}^{(u)} \tag{42}$$

Como el número de pasajeros que esperan en cada estación según destino, $\omega_{k,i,j}$, depende del tiempo, concretamente del tiempo transcurrido entre la salida de cada pareja de servicios consecutivos, y dado que estos instantes de salida son variables del problema y por tanto desconocidos, para conseguir una representación lineal, deben incluirse las siguientes restricciones:

$$d_{k,i} = \sum_{t=1}^{T} \left(y_{k,i,t} - y_{k,i,t-1}\right) * t \quad \forall k \in \mathbf{K}^{(u)}, \forall\, i \in \mathbf{S}^{(u)} \tag{43.a}$$

$$\omega_{k,i,j} = \sum_{t=1}^{T} \left(1 - y_{k,i,t}\right) * OD_{i,j}[t] \quad k = 1, \forall i,j \in \mathbf{S}^{(u)} \tag{43.b}$$

$$\omega_{k,i,j} = \omega_{k-1,i,j} + \sum_{t=1}^{T}\left(y_{k-1,i,t} - y_{k,i,t}\right) * OD_{i,j}[t] \quad \forall k \in \mathbf{K}^{(u)}, \forall\, i,j \in \mathbf{S}^{(u)}, k \neq 1 \qquad (43.c)$$

Puesto que los pasajeros sólo pueden subir en servicios que paran en su origen y alcanzan su destino, incluimos las siguientes restricciones:

$$\lambda_{k,i} + \lambda_{k-1,i} \leq 1 + \xi_{i,j}^{k} \quad \forall k \in \mathbf{K}^{(u)}, \forall\, i,j \in \mathbf{S}^{(u)}, i \neq j \qquad (44.a)$$

$$2\xi_{i,j}^{k} \leq \lambda_{k,i} + \lambda_{k-1,i} \quad \forall k \in \mathbf{K}^{(u)}, \forall i,j \in \mathbf{S}^{(u)}, i \neq j \qquad (44.b)$$

$$\omega_{k,i,j}^{b} \leq \omega_{k,i,j} \cdot \xi_{i,j}^{k} \quad \forall k \in \mathbf{K}^{(u)}, \forall\, i,j \in \mathbf{S}^{(u)}, i \neq j \qquad (44.c)$$

Para calcular el número de pasajeros que viajan en el servicio k, tras visitar la estación i:

$$c_{k,i} \leq c_{k,i-1} - c_{k,i}^{a} + c_{k,i}^{b} \quad \forall k \in \mathbf{K}^{(u)}, \forall\, i \in \mathbf{S}^{(u)}, i \neq 1 \qquad (45)$$

Debido a la limitación de capacidad del tren, no todos los pasajeros que esperan un servicio k en la estación i pueden subir al tren, por este motivo, el número de pasajeros que suben al servicio k en la estación i se calcula como:

$$c_{k,i}^{b} = min\{C - c_{k,i-1} + c_{k,i}^{a}, \omega_{k,i}^{b}\} \quad \forall k \in \mathbf{K}^{(u)}, \forall i \in \mathbf{S}^{(u)}, i \neq 1 \qquad (46)$$

En la ecuación (46), $c_{k,i}^{a}$ representa el número de pasajeros que abandonan el servicio k en la estación i, que se puede calcular como:

$$c_{k,i}^{a} = \sum_{j=1}^{i-1} c_{k,j,i}^{b} \quad \forall k \in \mathbf{K}^{(u)}, \forall i \in \mathbf{S}^{(u)} \qquad (47)$$

A continuación, calcularemos el número de pasajeros $c_{k,i}^{b}$ que pueden subir al servicio k en la estación i con destino j. Si $c_{k,i}^{b} = c_{k,i}$, todos los pasajeros que están esperando en la estación i al servicio k pueden montarse. Sin embargo, si $c_{k,i}^{b} < c_{k,i}$, tendremos que realizar una estimación de $c_{k,i,j}^{b}$, es decir, de los pasajeros que están esperando el servicio k en la estación i con destino j y pueden subir al tren que realiza el servicio. La frecuencia de la línea de metro es alta y los pasajeros no tienen preferencia por un servicio en concreto, sino por el primero que les lleve a su destino, y lo usarán siempre que haya espacio suficiente. Es razonable pensar que los pasajeros que esperan con diferentes destinos suben al tren en proporción al destino, es decir, si por ejemplo ante la llegada de un servicio hay 100 pasajeros esperando con dos destinos (estaciones k y s), 80 para el destino k y 20 para el s, y en el tren sólo caben 50, consideraremos que suben 40 con destino a k y 10 con destino a s, Para conseguir esta proporción, las variables $c_{k,i}^{b}$ y $c_{k,i,j}^{b}$ deben satisfacer:

$$\emptyset_i = \sum_{j\in \mathbf{S}^{(u)}, i<j} \sum_{t\in T} OD_{i,j}[t] \quad \forall i \in \mathbf{S}^{(u)} \tag{48.a}$$

$$\emptyset_{i,j} = \sum_{t\in T} OD_{i,j}[t] \quad \forall i,j \in \mathbf{S}^{(u)}, i<j \tag{48.b}$$

$$c^b_{k,i,j} = \frac{\emptyset_{i,j}}{\emptyset_i} \cdot c^b_{k,i} \quad \forall k \in \mathbf{K}^{(u)}, \forall i,j \in \mathbf{S}^{(u)}, i<j \tag{48.c}$$

Finalmente, los pasajeros restantes que están esperando el servicio k con destino la estación j, y que no pueden tomar este servicio (pasajeros dejados en tierra o residuo), se calcula de la siguiente forma:

$$\omega^r_{k,i,j} = \omega_{k,i,j} - c^b_{k,i,j} \quad \forall k \in \mathbf{K}^{(u)}, \forall i,j \in \mathbf{S}^{(u)}, i<j \tag{49}$$

Como se comentó al principio del apartado, todas las restricciones deben escribirse de forma independiente para ambos sentidos, de manera que deberán modelarse las mismas en sentido aguas abajo.

$$\omega_{k,i} = \sum_{j=N+1}^{2N} \omega_{k,i,j} \quad \forall k \in \mathbf{K}^{(d)}, \forall\, i \in \mathbf{S}^{(d)} \tag{50}$$

$$\omega^b_{k,i} = \sum_{j=N+1}^{2N} \omega^b_{k,i,j} \quad \forall k \in \mathbf{K}^{(d)}, \forall\, i \in \mathbf{S}^{(d)} \tag{51}$$

$$c^b_{k,i} = \sum_{j=N+1}^{2N} c^b_{k,i,j} \quad \forall k \in \mathbf{K}^{(d)}, \forall i \in \mathbf{S}^{(d)} \tag{52}$$

$$d_{k,i} = \sum_{t=1}^{T} \left(y_{k,i,t} - y_{k,i,t-1}\right) * t \quad \forall k \in \mathbf{K}^{(d)}, \forall i \in \mathbf{S}^{(d)} \tag{53.a}$$

$$\omega_{k,i,j} = \sum_{t=1}^{T} \left(1 - y_{k,i,t}\right) * OD_{i,j}[t] \quad k=1, \forall i \in \mathbf{S}^{(u)} \tag{53.b}$$

$$\omega_{k,i,j} = \omega_{k-1,i,j} + \sum_{t=1}^{T} \left(y_{k-1,i,t} - y_{k,i,t}\right) * OD_{i,j}[t] \quad \forall k \in \mathbf{K}^{(d)}, i \in \mathbf{S}^{(d)}, k \neq 1 \tag{53.c}$$

$$\lambda_{k,i} + \lambda_{k-1,i} \leq 1 + \xi^k_{i,j} \quad \forall k \in \mathbf{K}^{(d)}, \forall i,j \in \mathbf{S}^{(d)}, i \neq j \tag{54.a}$$

$$2\xi^k_{i,j} \leq \lambda_{k,i} + \lambda_{k-1,i} \quad \forall k \in \mathbf{K}^{(d)}, \forall\, i,j \in \mathbf{S}^{(d)}, i \neq j \tag{54.b}$$

$$\omega^b_{k,i,j} \leq \omega_{k,i,j} \cdot \xi^k_{i,j} \quad \forall k \in \mathbf{K}^{(d)}, \forall\, i,j \in \mathbf{S}^{(d)}, i \neq j \tag{54.c}$$

$$c_{k,i} \leq c_{k,i-1} - c^a_{k,i} + c^b_{k,i} \quad \forall k \in \mathbf{K}^{(d)}, i \in \mathbf{S}^{(d)}, i \neq 1 \tag{55}$$

$$c^b_{k,i} = \left\{C - c_{k,i-1} + c^a_{k,i}, \omega^b_{k,i}\right\} \quad \forall k \in \mathbf{K}^{(d)}, \forall\, i \in \mathbf{S}^{(d)}, i \neq 1 \tag{56}$$

$$c^a_{k,i} = \sum_{j=1}^{i-1} c^b_{k,j,i} \quad \forall k \in \mathbf{K}^{(d)}, \forall i \in \mathbf{S}^{(d)} \tag{57}$$

$$\emptyset_i = \sum_{j \in \mathbf{S}^{(d)}} \sum_{t \in T} OD_{i,j}[t] \quad \forall i \in \mathbf{S}^{(d)} \tag{58.a}$$

$$\emptyset_{i,j} = \sum_{t \in T} OD_{i,j}[t] \quad \forall i,j \in \mathbf{S}^{(d)}, i < j \tag{58.b}$$

$$c^b_{k,i,j} = \frac{\emptyset_{i,j}}{\emptyset_i} \cdot c^b_{k,i} \quad \forall k \in \mathbf{K}^{(d)}, \forall i,j \in \mathbf{S}^{(d)}, i < j \tag{58.c}$$

$$\omega^r_{k,i,j} = \omega_{k,i,j} - c^b_{k,i,j} \quad \forall k \in \mathbf{K}^{(d)}, \forall i,j \in \mathbf{S}^{(d)}, i < j \tag{59}$$

4.3. Función Objetivo

El objetivo consiste en minimizar el tiempo total de espera de los pasajeros en las plataformas. Consideramos $f_k^{(u)}$ como el tiempo de espera de los pasajeros para el servicio k en sentido aguas arriba.

$$f_k^{(u)} = \sum_{i \in S^{(u)}} \sum_{j \in S^{(u)}, j > i} \frac{(\omega^r_{k-1,i,j} + \omega_{k,i,j}) \cdot (d_{k,i} - d_{k-1,i})}{2} \tag{60}$$

Y lo mismo para el sentido contrario.

$$f_k^{(d)} = \sum_{i \in S^{(d)}} \sum_{j \in S^{(d)}, j > i} \frac{(\omega^r_{k-1,i,j} + \omega_{k,i,j}) \cdot (d_{k,i} - d_{k-1,i})}{2} \tag{61}$$

Finalmente, definimos la función objetivo como la suma de los tiempos de espera de los pasajeros para cada servicio en sentido aguas arriba y en sentido aguas abajo.

$$F = \sum_{k \in \mathbf{K}^{(u)}} f_k^{(u)} + \sum_{k \in \mathbf{K}^{(d)}} f_k^{(d)} \tag{62}$$

Utilizando $\boldsymbol{\tau} = (\tau_k)$, $\mathbf{z} = (z_{k,m,n})$, $\boldsymbol{\lambda} = (\lambda_{k,i})$, $\mathbf{h} = (h_{k-1,k})$, $\mathbf{d} = (d_{k,i})$, $\mathbf{a} = (a_{k,i})$ y $\boldsymbol{\gamma} = (\gamma_{k,l,m})$ para simplificar, podemos formular el modelo como:

$$\begin{cases} min_{\tau,z,\lambda,h,d,a,\gamma}\ F \\ Restricciones\ (1) - (59) \end{cases}$$

5 EXPERIMENTOS COMPUTACIONALES

Partiendo del modelo matemático explicado en el punto anterior, y programado en lenguaje Python y haciendo uso del software Gurobi, se han realizado tres experimentos variando el número de estaciones y servicios para sacar conclusiones a partir de los resultados obtenidos.

5.1. Datos de entrada

En primer lugar, tomamos como dato la siguiente matriz, que nos indica el tiempo entre estación y estación:

s (min)	1	2	3	4	5	6	7	8	9	10	11	12
1	0,00	3,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
2	3,00	0,00	3,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
3	0,00	3,00	0,00	3,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
4	0,00	0,00	3,00	0,00	3,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
5	0,00	0,00	0,00	3,00	0,00	3,00	0,00	0,00	0,00	0,00	0,00	0,00
6	0,00	0,00	0,00	0,00	4,00	0,00	3,00	0,00	0,00	0,00	0,00	0,00
7	0,00	0,00	0,00	0,00	0,00	3,00	0,00	3,00	0,00	0,00	0,00	0,00
8	0,00	0,00	0,00	0,00	0,00	0,00	3,00	0,00	3,00	0,00	0,00	0,00
9	0,00	0,00	0,00	0,00	0,00	0,00	0,00	3,00	0,00	3,00	0,00	0,00
10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	3,00	0,00	3,00	0,00
11	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	3,00	0,00	3,00
12	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	3,00	0,00

Tabla 1 Tiempo entre estaciones

Como se puede observar, se ha tomado 3 y 4 como valores entre estaciones, para simplificar la matriz. He de apuntar que solo nos interesan los valores que rodean la diagonal principal de la matriz, considerando los demás valores 0. La tabla anterior, Ilustración 4, cambia en el Caso 2 ya que varía el número de estaciones operativas.

Por otro lado, se consideran los siguientes valores:

- Headway máximo: 20
- Headway mínimo: 2
- Número de trenes totales: 27
- Capacidad de los trenes: 400
- Tiempo máximo para el cambio de sentido en las estaciones mayores o ∂_{Max}. Este depende del número de estaciones mayores de manera que es igual en los casos 1 y 3, y cambia en el caso 2.

1	6,00
2	6,00
5	6,00
6	6,00
7	6,00
8	6,00
11	6,00
12	6,00

Tabla 2 ∂_{Max} de los Casos 1 y 3

1	6,00
2	6,00
5	6,00
7	6,00
8	6,00
9	6,00
10	6,00
12	6,00
15	6,00
16	6,00

Tabla 3 ∂_{Max} del Caso 2

- Tiempo mínimo para el cambio de sentido en las estaciones mayores o $\partial_{Mín}$. Al igual que ∂_{Max} en el punto anterior, estas matrices dependen del número de estaciones mayores.

1	1,00
2	1,00
5	1,00
6	1,00
7	1,00
8	1,00
11	1,00
12	1,00

Tabla 4 $\partial_{Mín}$ de los Casos 1 y 3

1	1,00
2	1,00
5	1,00
7	1,00
8	1,00
9	1,00
10	1,00
12	1,00
15	1,00
16	1,00

Tabla 5 $\partial_{Mín}$ del Caso 2

- En la siguiente tabla, se pueden observar las estaciones que de los Casos 1 y 3. Remarcar que las casillas resaltadas indican las estaciones mayores, es decir, aquellas en las que se puede hacer un cambio de sentido. Los andenes que pertenecen al sentido aguas abajo son de la 1 a la 6, y las restantes pertenecen al sentido aguas abajo.

E1	1
E2	2
E3	3
E4	4
E5	5
E6	6
E7	7
E8	8
E9	9
E10	10
E11	11
E12	12

Tabla 6 Andenes en los Casos 1 y 3

- Al igual que en Ilustración 10, en la siguiente ilustración encontramos los andenes utilizados para el Caso 2. En este caso, los andenes que pertenecen al sentido aguas arriba son del 1 al 8, y las restantes pertenecen al sentido contrario.

E1	1
E2	2
E3	3
E4	4
E5	5
E6	6
E7	7
E8	8
E9	9
E10	10
E11	11
E12	12
E13	13
E14	14
E15	15
E16	16

Tabla 7 Andenes en el Caso 2

- Las líneas que se han utilizado ahora son iguales en los Casos 1 y 2, y varían en el último caso al que se le han hecho pruebas. Estas son:

S1	1
S2	2
S3	3
S4	4
S5	5
S6	6
S7	7
S8	8
S9	9
S10	10
S11	11
S12	12
S13	13
S14	14
S15	15
S16	16
S17	17
S18	18
S19	19
S20	20

Tabla 8 Servicios en los Casos 1 y 2

- En el caso 3 se aumentaron el número de servicios para ver como respondería el modelo, por ello:

S1	1
S2	2
S3	3
S4	4
S5	5
S6	6
S7	7
S8	8
S9	9
S10	10
S11	11
S12	12
S13	13
S14	14
S15	15
S16	16
S17	17
S18	18
S19	19
S20	20
S21	21
S22	22
S23	23
S24	24

Tabla 9 Servicios en el Caso 3

- Finalmente, el último dato de entrada al modelo es la demanda que viene a partir de las siguientes tablas:

	Semillas	0	0	0	0	0	0	0	0	0	40	35	20	10	10	15	15	15	15	15	15	15	15	40	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	0	0	0	0	0	0	0	0	0
		3:30	4:00	4:30	5:00	5:30	6:00	6:30	7:00	7:30	8:00	8:30	9:00	9:30	10:00	10:30	11:00	11:30	12:00	12:30	13:00	13:30	14:00	14:30	15:00	15:30	16:00	16:30	17:00	17:30	18:00	18:30	19:00	19:30	20:00	20:30	21:00	21:30	22:00	22:30	23:00	23:30	0:00	0:30	1:00	1:30	2:00	2:30	3:00	3:30
Semillas	ODijt	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
1	1.2	0	0	0	0	0	0	0	0	0	26	27	9	5	5	15	14	9	3	7	6	13	5	28	6	1	10	11	12	8	2	1	6	13	13	11	6	10	9	5	7	0	0	0	0	0	0	0	0	0
1	1.3	0	0	0	0	0	0	0	0	0	32	24	8	4	2	15	12	11	3	5	14	10	14	29	8	14	6	1	11	15	5	9	9	8	2	10	2	14	13	6	5	0	0	0	0	0	0	0	0	0
1	1.4	0	0	0	0	0	0	0	0	0	16	33	3	10	7	12	13	9	7	14	6	4	1	18	6	11	12	12	4	6	3	13	9	8	13	9	7	8	7	6	9	0	0	0	0	0	0	0	0	0
1	1.5	0	0	0	0	0	0	0	0	0	4	25	11	8	8	5	13	14	6	5	8	2	11	12	6	9	3	14	12	1	10	4	1	5	11	5	8	10	10	3	9	0	0	0	0	0	0	0	0	0
1	1.6	0	0	0	0	0	0	0	0	0	14	12	19	8	1	9	7	7	14	7	13	4	2	8	12	11	3	6	9	10	8	12	15	5	6	2	15	12	1	15	6	0	0	0	0	0	0	0	0	0
1	2.3	0	0	0	0	0	0	0	0	0	33	33	19	4	2	12	11	5	3	11	4	3	3	35	4	13	8	7	5	1	1	13	6	12	14	15	3	9	10	12	13	0	0	0	0	0	0	0	0	0
1	2.4	0	0	0	0	0	0	0	0	0	16	29	10	8	5	10	9	9	5	10	3	5	7	13	2	1	5	7	7	3	6	4	9	10	13	5	6	2	11	4	15	0	0	0	0	0	0	0	0	0
1	2.5	0	0	0	0	0	0	0	0	0	6	33	19	5	5	1	10	7	7	4	5	4	15	38	12	1	15	13	4	6	4	8	12	13	7	1	15	2	2	5	3	0	0	0	0	0	0	0	0	0
1	2.6	0	0	0	0	0	0	0	0	0	20	2	17	1	8	12	8	3	3	2	15	8	7	31	6	10	5	3	14	15	6	11	8	12	14	11	12	8	8	12	11	0	0	0	0	0	0	0	0	0
100	3.4	0	0	0	0	0	0	0	0	0	1986	2131	744	523	611	670	192	57	909	586	114	300	1321	254	1118	932	1413	1169	1026	366	1080	929	539	1276	1118	1378	1086	123	811	713	223	0	0	0	0	0	0	0	0	0
100	3.5	0	0	0	0	0	0	0	0	0	1693	744	1064	426	576	1111	1170	882	769	544	610	24	876	2978	478	653	512	56	178	352	64	453	134	1037	962	944	56	738	867	376	992	0	0	0	0	0	0	0	0	0
100	3.6	0	0	0	0	0	0	0	0	0	1329	1587	1910	394	211	791	876	1065	26	55	83	1417	1277	3987	403	1294	792	858	413	1306	172	1109	1220	1050	54	1363	813	1243	450	167	1051	0	0	0	0	0	0	0	0	0
100	4.5	0	0	0	0	0	0	0	0	0	3769	1488	838	335	79	749	597	688	135	671	581	1026	481	2775	222	209	87	736	530	914	251	1445	918	453	555	752	1280	530	1264	439	238	0	0	0	0	0	0	0	0	0
100	4.6	0	0	0	0	0	0	0	0	0	1923	3453	1317	113	984	1248	445	334	917	197	1242	102	1430	2726	240	511	1069	436	1020	1013	1257	736	760	557	739	221	1131	1264	1280	404	723	0	0	0	0	0	0	0	0	0
1	5.6	0	0	0	0	0	0	0	0	0	39	9	18	7	1	9	13	9	11	11	13	2	12	35	12	13	1	3	4	3	15	9	3	4	4	15	2	1	9	3	9	0	0	0	0	0	0	0	0	0
1	7.8	0	0	0	0	0	0	0	0	0	37	3	10	6	10	10	10	15	6	13	12	15	7	36	12	4	12	13	2	6	5	14	3	3	5	5	5	2	5	11	14	0	0	0	0	0	0	0	0	0
100	7.9	0	0	0	0	0	0	0	0	0	1830	2649	1076	223	629	1099	451	1018	805	1159	478	1040	303	3802	1481	1361	63	242	716	1297	1454	995	600	624	618	351	1013	1335	106	1076	338	0	0	0	0	0	0	0	0	0
100	7.10	0	0	0	0	0	0	0	0	0	1591	3150	428	745	537	171	1044	203	496	67	1319	904	302	514	229	166	1371	511	956	1493	161	195	742	597	873	760	1141	1049	791	810	1009	0	0	0	0	0	0	0	0	0
1	7.11	0	0	0	0	0	0	0	0	0	27	18	17	9	1	15	13	10	8	4	3	4	2	10	8	10	13	8	1	7	2	4	9	13	11	9	2	5	10	6	12	0	0	0	0	0	0	0	0	0
1	7.12	0	0	0	0	0	0	0	0	0	35	17	19	6	7	15	1	11	8	6	1	12	1	20	7	4	6	7	12	12	15	7	7	9	6	7	2	11	9	6	8	0	0	0	0	0	0	0	0	0
100	8.9	0	0	0	0	0	0	0	0	0	1796	2486	284	35	442	1143	181	1229	1001	330	576	573	845	2223	712	1434	106	1308	737	267	1395	1251	135	302	51	542	373	701	373	50	499	0	0	0	0	0	0	0	0	0
100	8.10	0	0	0	0	0	0	0	0	0	457	469	1241	974	997	679	266	1413	1183	166	1125	87	834	2534	23	21	189	981	602	406	696	233	569	688	1495	102	1021	1279	673	1353	1112	0	0	0	0	0	0	0	0	0
1	8.11	0	0	0	0	0	0	0	0	0	22	17	20	4	6	11	3	9	1	2	8	15	10	22	10	15	10	14	15	2	10	8	13	11	15	15	3	9	11	13	4	0	0	0	0	0	0	0	0	0
1	8.12	0	0	0	0	0	0	0	0	0	6	27	11	8	10	14	14	3	15	11	10	1	1	3	12	1	13	15	8	8	3	7	1	8	1	3	10	3	5	12	10	0	0	0	0	0	0	0	0	0
100	9.10	0	0	0	0	0	0	0	0	0	2338	2692	162	904	876	35	881	1197	724	294	1475	315	527	1371	254	1235	660	696	678	1083	751	490	1116	662	1248	499	614	962	1381	871	717	0	0	0	0	0	0	0	0	0
1	9.11	0	0	0	0	0	0	0	0	0	18	14	14	1	10	14	1	1	2	7	8	4	15	10	13	4	10	4	13	9	8	3	11	7	2	15	11	10	1	11	8	0	0	0	0	0	0	0	0	0
1	9.12	0	0	0	0	0	0	0	0	0	27	24	17	5	9	11	1	4	14	8	4	5	12	10	9	8	11	5	5	7	12	14	8	13	1	5	11	9	14	13	1	0	0	0	0	0	0	0	0	0
1	10.11	0	0	0	0	0	0	0	0	0	31	34	3	3	7	12	11	3	11	11	13	7	6	34	3	2	12	13	8	4	1	2	8	3	2	4	11	14	1	10	9	0	0	0	0	0	0	0	0	0
1	10.12	0	0	0	0	0	0	0	0	0	40	1	14	8	2	13	11	15	4	1	11	2	4	1	7	14	7	6	7	5	4	12	5	14	2	2	7	9	3	15	12	0	0	0	0	0	0	0	0	0
1	11.12	0	0	0	0	0	0	0	0	0	28	5	14	1	4	8	3	11	11	7	11	5	8	36	12	10	1	1	6	6	10	11	4	6	2	13	5	13	11	10	13	0	0	0	0	0	0	0	0	0

Tabla 10 Demanda en el Caso 1

	Semillas	0	0	0	0	0	0	0	0	0	1	1	1	1	10	30	30	30	30	30	30	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	3:00	3:30	4:00	4:30	5:00	5:30	6:00	6:30	7:00	7:30	8:00	8:30	9:00	9:30	10:00	10:30	11:00	11:30	12:00	12:30	13:00	13:30	14:00	14:30	15:00	15:30	16:00	16:30	17:00	17:30	18:00	18:30	19:00	19:30	20:00	20:30	21:00	21:30	22:00	22:30	23:00	23:30	0:00	0:30	1:00	1:30	2:00	2:30	3:00	3:30
Semillas	ODijt	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
2	1.2	0	0	0	0	0	0	0	0	0	2	2	2	2	5	29	10	53	33	6	48	1	1	2	2	1	1	2	1	2	1	1	2	2	2	2	2	1	2	2	1	0	0	0	0	0	0	0	0	0
2	1.3	0	0	0	0	0	0	0	0	0	1	2	1	1	11	52	40	37	45	8	55	2	1	1	1	1	1	1	2	1	1	2	1	2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
2	1.4	0	0	0	0	0	0	0	0	0	1	2	2	2	14	42	19	30	43	35	47	1	1	1	1	1	2	1	1	1	2	1	1	2	1	1	2	2	2	2	2	0	0	0	0	0	0	0	0	0
2	1.5	0	0	0	0	0	0	0	0	0	1	2	1	1	8	3	60	34	47	30	44	2	1	1	2	1	2	1	2	2	2	1	2	1	2	2	2	1	1	1	2	0	0	0	0	0	0	0	0	0
2	1.6	0	0	0	0	0	0	0	0	0	1	1	2	1	4	31	48	39	4	47	17	1	2	1	2	2	1	2	2	1	2	2	1	1	1	1	1	2	2	1	2	0	0	0	0	0	0	0	0	0
2	1.7	0	0	0	0	0	0	0	0	0	1	2	1	1	5	2	14	56	56	22	21	2	1	2	1	1	1	2	1	1	2	2	1	1	2	1	1	2	2	2	1	0	0	0	0	0	0	0	0	0
2	1.8	0	0	0	0	0	0	0	0	0	1	2	1	1	9	43	45	4	27	31	4	1	2	2	2	2	2	2	2	2	1	1	2	2	2	1	2	2	2	1	1	0	0	0	0	0	0	0	0	0
2	2.3	0	0	0	0	0	0	0	0	0	1	2	1	2	8	46	43	38	23	45	36	1	2	1	2	2	2	1	1	2	2	2	2	1	1	1	1	2	1	2	1	0	0	0	0	0	0	0	0	0
2	2.4	0	0	0	0	0	0	0	0	0	1	1	2	1	20	8	11	49	43	50	44	1	2	2	2	1	1	2	1	1	1	2	1	1	2	2	1	2	1	2	1	0	0	0	0	0	0	0	0	0
2	2.5	0	0	0	0	0	0	0	0	0	2	1	2	2	20	21	45	33	18	15	9	2	1	1	1	2	1	1	2	1	1	1	2	2	1	1	2	1	1	1	1	0	0	0	0	0	0	0	0	0
2	2.6	0	0	0	0	0	0	0	0	0	2	2	1	1	7	14	4	29	45	55	46	2	2	1	1	2	2	2	2	1	2	2	2	2	2	1	1	2	2	2	2	0	0	0	0	0	0	0	0	0
2	2.7	0	0	0	0	0	0	0	0	0	2	2	1	2	11	33	49	58	12	36	36	2	1	2	1	1	2	1	1	2	1	2	1	1	1	1	2	1	2	2	2	0	0	0	0	0	0	0	0	0
2	2.8	0	0	0	0	0	0	0	0	0	1	2	1	1	14	56	21	54	58	7	26	2	2	2	1	2	1	2	1	2	2	2	1	2	2	1	1	1	2	1	2	0	0	0	0	0	0	0	0	0
2	3.4	0	0	0	0	0	0	0	0	0	2	1	2	2	20	59	18	56	43	37	44	1	2	2	1	1	2	2	1	1	2	1	2	1	1	1	1	2	2	2	1	0	0	0	0	0	0	0	0	0
50	3.5	0	0	0	0	0	0	0	0	0	46	32	50	43	60	938	409	776	2	1349	1437	40	36	31	8	24	5	50	39	2	17	13	20	15	47	9	12	20	29	4	43	0	0	0	0	0	0	0	0	0
50	3.6	0	0	0	0	0	0	0	0	0	13	9	49	5	301	1207	905	300	966	340	1068	30	48	4	18	14	5	10	20	38	15	12	49	28	1	34	30	14	30	5	40	0	0	0	0	0	0	0	0	0
50	3.7	0	0	0	0	0	0	0	0	0	44	42	48	48	374	403	950	650	567	209	906	8	15	38	21	47	47	15	34	35	4	45	22	13	36	33	39	40	24	17	40	0	0	0	0	0	0	0	0	0
50	3.8	0	0	0	0	0	0	0	0	0	16	2	8	14	308	1322	198	1049	1347	523	786	6	43	25	24	10	11	28	36	5	14	11	28	34	16	38	28	25	39	47	6	0	0	0	0	0	0	0	0	0
50	4.5	0	0	0	0	0	0	0	0	0	17	22	9	23	13	944	629	1074	429	1146	1164	17	44	28	42	6	10	4	36	25	7	28	24	46	15	13	8	33	4	6	36	0	0	0	0	0	0	0	0	0
50	4.6	0	0	0	0	0	0	0	0	0	8	50	5	5	191	437	150	417	87	468	641	49	32	36	41	3	41	40	47	21	13	48	26	11	1	23	8	25	40	26	35	0	0	0	0	0	0	0	0	0
50	4.7	0	0	0	0	0	0	0	0	0	30	25	42	4	61	388	689	309	477	374	804	36	32	21	48	49	2	41	7	20	17	40	31	24	32	45	30	26	10	38	31	0	0	0	0	0	0	0	0	0
50	4.8	0	0	0	0	0	0	0	0	0	23	10	48	4	110	535	830	671	652	66	558	17	7	29	44	44	9	6	21	2	48	15	12	38	21	32	30	24	35	40	36	0	0	0	0	0	0	0	0	0
2	5.6	0	0	0	0	0	0	0	0	0	1	2	2	2	13	25	28	3	53	50	26	1	1	1	1	1	2	1	1	1	1	2	2	2	1	1	1	1	1	2	2	0	0	0	0	0	0	0	0	0
2	5.7	0	0	0	0	0	0	0	0	0	2	1	2	1	13	38	51	12	59	6	23	2	2	2	2	2	2	1	2	1	1	2	1	2	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0	0
2	5.8	0	0	0	0	0	0	0	0	0	1	2	2	2	2	7	7	5	11	56	27	1	2	1	2	2	1	2	1	2	2	2	2	1	2	2	1	2	2	2	1	0	0	0	0	0	0	0	0	0
2	6.7	0	0	0	0	0	0	0	0	0	2	2	2	1	3	56	55	1	43	13	5	1	2	2	1	2	2	1	2	1	2	2	1	2	2	1	2	2	1	1	2	0	0	0	0	0	0	0	0	0
2	6.8	0	0	0	0	0	0	0	0	0	1	2	1	2	8	38	48	35	9	29	44	2	2	1	1	2	1	2	1	1	1	2	2	2	1	2	1	2	1	2	1	0	0	0	0	0	0	0	0	0
2	7.8	0	0	0	0	0	0	0	0	0	2	1	1	1	3	26	16	51	25	48	21	2	1	2	2	2	1	1	2	2	1	2	2	2	2	2	2	1	1	2	1	0	0	0	0	0	0	0	0	0
2	9.10	0	0	0	0	0	0	0	0	0	1	2	2	1	7	13	43	3	41	34	3	1	1	1	1	1	2	1	1	2	2	1	2	1	2	2	2	2	2	1	2	0	0	0	0	0	0	0	0	0
2	9.11	0	0	0	0	0	0	0	0	0	1	2	1	1	3	42	59	8	3	19	29	1	2	1	2	2	2	2	1	2	1	1	1	1	1	2	1	1	1	2	1	0	0	0	0	0	0	0	0	0
2	9.12	0	0	0	0	0	0	0	0	0	2	2	1	2	3	4	25	22	50	58	57	1	1	1	2	2	2	1	1	2	1	2	2	2	1	2	1	1	2	1	2	0	0	0	0	0	0	0	0	0
50	9.13	0	0	0	0	0	0	0	0	0	45	49	1	2	231	738	821	1182	778	201	1260	36	14	6	47	35	16	32	1	43	28	27	1	30	6	24	32	13	1	49	49	0	0	0	0	0	0	0	0	0
50	9.14	0	0	0	0	0	0	0	0	0	2	7	32	11	128	1079	128	1232	989	1472	1084	29	25	46	34	5	34	1	44	10	16	47	4	40	46	25	43	7	5	2	32	0	0	0	0	0	0	0	0	0
2	9.15	0	0	0	0	0	0	0	0	0	1	1	2	1	3	46	57	31	8	30	21	2	2	1	1	2	1	1	2	1	2	1	1	1	1	1	2	2	2	2	1	0	0	0	0	0	0	0	0	0
2	9.16	0	0	0	0	0	0	0	0	0	2	1	1	1	14	6	17	42	36	41	22	1	1	2	2	2	2	2	2	1	2	2	2	1	2	1	1	2	1	2	1	0	0	0	0	0	0	0	0	0
2	10.11	0	0	0	0	0	0	0	0	0	1	1	1	1	3	56	54	22	17	52	37	1	2	1	2	1	2	2	2	2	2	2	1	2	1	2	1	1	2	2	1	0	0	0	0	0	0	0	0	0
2	10.12	0	0	0	0	0	0	0	0	0	1	2	1	1	8	25	48	9	39	11	24	2	1	2	2	2	1	1	1	2	1	2	1	2	2	2	2	1	1	1	2	0	0	0	0	0	0	0	0	0
50	10.13	0	0	0	0	0	0	0	0	0	30	15	1	13	487	860	1426	378	370	431	357	31	41	42	15	12	46	21	43	13	24	1	33	21	50	8	24	31	37	38	23	0	0	0	0	0	0	0	0	0
50	10.14	0	0	0	0	0	0	0	0	0	16	45	34	23	393	562	591	755	907	56	1141	9	4	33	16	13	14	27	47	26	42	38	45	9	25	35	11	28	16	41	13	0	0	0	0	0	0	0	0	0
2	10.15	0	0	0	0	0	0	0	0	0	1	1	2	1	19	43	5	54	53	15	52	1	2	1	2	2	2	2	2	2	2	1	1	2	1	1	1	2	1	1	1	0	0	0	0	0	0	0	0	0
2	10.16	0	0	0	0	0	0	0	0	0	1	2	2	1	3	3	1	3	14	56	58	1	2	1	2	1	1	2	1	2	2	1	1	2	1	2	1	1	2	1	2	0	0	0	0	0	0	0	0	0
2	11.12	0	0	0	0	0	0	0	0	0	1	1	2	1	18	55	38	41	60	6	9	2	1	1	2	1	1	2	2	2	1	1	1	1	2	1	2	2	2	2	2	0	0	0	0	0	0	0	0	0
100	11.13	0	0	0	0	0	0	0	0	0	60	43	45	56	920	793	1377	1509	2722	1141	2907	38	93	78	35	34	16	7	54	30	68	27	8	42	18	92	13	97	99	12	33	0	0	0	0	0	0	0	0	0
100	11.14	0	0	0	0	0	0	0	0	0	73	88	28	8	817	123	1019	1122	969	2467	1429	20	28	7	7	90	80	63	21	74	12	87	97	33	49	79	7	10	15	4	44	0	0	0	0	0	0	0	0	0
2	11.15	0	0	0	0	0	0	0	0	0	2	1	1	2	2	25	8	20	52	7	14	1	2	2	2	1	2	1	2	1	2	2	2	2	2	1	2	1	1	1	2	0	0	0	0	0	0	0	0	0
2	11.16	0	0	0	0	0	0	0	0	0	1	2	2	1	18	20	30	16	18	58	44	2	2	1	2	1	2	2	1	2	2	2	1	2	1	1	1	2	1	2	1	0	0	0	0	0	0	0	0	0
100	12.13	0	0	0	0	0	0	0	0	0	29	36	42	26	277	2526	623	614	2026	1499	1551	63	90	21	98	52	79	93	45	43	43	78	26	98	90	39	23	71	22	76	6	0	0	0	0	0	0	0	0	0
100	12.14	0	0	0	0	0	0	0	0	0	80	87	51	9	330	2833	2392	1706	1134	1405	1731	61	3	100	7	52	6	29	5	58	21	94	40	96	60	39	55	81	68	11	16	0	0	0	0	0	0	0	0	0
2	12.15	0	0	0	0	0	0	0	0	0	1	2	2	1	9	34	4	50	44	9	51	2	1	1	1	2	2	1	1	2	1	1	2	2	1	2	2	2	2	1	1	0	0	0	0	0	0	0	0	0
2	12.16	0	0	0	0	0	0	0	0	0	1	2	2	1	9	21	39	5	35	13	34	1	2	1	2	2	1	1	1	2	1	2	1	1	2	2	2	2	2	1	2	0	0	0	0	0	0	0	0	0
2	13.14	0	0	0	0	0	0	0	0	0	1	1	2	2	14	18	8	30	3	22	28	2	2	2	2	1	1	1	2	1	2	2	2	1	2	1	1	1	2	1	1	0	0	0	0	0	0	0	0	0
2	13.15	0	0	0	0	0	0	0	0	0	1	1	1	1	14	9	15	38	52	58	7	2	2	2	1	2	1	1	2	2	1	1	2	1	1	1	2	2	2	2	1	0	0	0	0	0	0	0	0	0
2	13.16	0	0	0	0	0	0	0	0	0	1	1	1	2	20	6	42	45	59	33	2	1	1	2	2	1	1	2	2	1	2	1	2	2	2	2	1	1	1	1	1	0	0	0	0	0	0	0	0	0
2	14.15	0	0	0	0	0	0	0	0	0	2	2	2	1	14	52	32	38	49	5	6	2	2	2	2	2	1	2	1	2	1	1	2	2	1	2	2	2	1	1	1	0	0	0	0	0	0	0	0	0
2	14.16	0	0	0	0	0	0	0	0	0	2	2	1	2	15	5	45	26	17	14	44	1	2	2	2	1	2	2	1	2	2	1	1	2	1	2	1	1	1	1	1	0	0	0	0	0	0	0	0	0
2	15.16	0	0	0	0	0	0	0	0	0	2	1	2	2	15	10	26	46	52	33	49	1	2	2	2	2	1	1	2	1	2	1	1	1	1	2	2	1	1	2	2	0	0	0	0	0	0	0	0	0

Tabla 11 Demanda en el Caso 2

	Semillas	0	0	0	0	0	0	0	0	0	40	35	20	10	10	15	15	15	15	15	15	15	15	40	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	0	0	0	0	0	0	0	0	0
		3:30	4:00	4:30	5:00	5:30	6:00	6:30	7:00	7:30	8:00	8:30	9:00	9:30	10:00	10:30	11:00	11:30	12:00	12:30	13:00	13:30	14:00	14:30	15:00	15:30	16:00	16:30	17:00	17:30	18:00	18:30	19:00	19:30	20:00	20:30	21:00	21:30	22:00	22:30	23:00	23:30	0:00	0:30	1:00	1:30	2:00	2:30	3:00	3:30
Semillas	ODijt	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
1	1.2	0	0	0	0	0	0	0	0	0	34	32	18	3	6	8	10	12	15	12	12	2	14	15	10	15	15	8	3	1	13	12	9	6	14	4	12	10	13	14	13	0	0	0	0	0	0	0	0	0
1	1.3	0	0	0	0	0	0	0	0	0	38	7	16	4	1	11	4	9	7	15	3	7	3	18	3	10	6	2	2	3	11	7	15	10	6	10	1	8	7	14	10	0	0	0	0	0	0	0	0	0
1	1.4	0	0	0	0	0	0	0	0	0	8	14	13	6	10	15	9	5	9	11	10	6	12	33	11	15	13	9	8	13	11	13	8	15	9	9	6	1	3	11	12	0	0	0	0	0	0	0	0	0
1	1.5	0	0	0	0	0	0	0	0	0	26	2	17	10	6	15	4	5	15	6	8	6	7	36	11	1	12	10	7	6	14	15	11	4	6	3	7	14	6	12	13	0	0	0	0	0	0	0	0	0
1	1.6	0	0	0	0	0	0	0	0	0	29	13	16	5	7	14	13	2	8	14	5	9	11	26	4	3	12	15	3	7	12	1	8	15	5	11	1	12	12	10	11	0	0	0	0	0	0	0	0	0
1	2.3	0	0	0	0	0	0	0	0	0	19	21	5	3	6	10	13	1	6	11	8	2	10	15	4	6	15	8	9	10	14	3	5	7	2	9	9	2	8	8	13	0	0	0	0	0	0	0	0	0
1	2.4	0	0	0	0	0	0	0	0	0	24	19	4	1	5	7	11	12	10	7	11	12	9	25	6	6	4	6	5	15	8	1	9	2	8	8	7	2	6	15	4	0	0	0	0	0	0	0	0	0
1	2.5	0	0	0	0	0	0	0	0	0	39	15	3	10	1	9	12	3	7	4	4	11	14	15	8	13	6	15	3	15	10	9	14	7	15	15	9	8	3	7	12	0	0	0	0	0	0	0	0	0
1	2.6	0	0	0	0	0	0	0	0	0	5	17	13	8	1	9	6	13	10	4	13	15	9	18	7	14	8	6	7	3	14	1	8	10	12	6	10	5	15	6	8	0	0	0	0	0	0	0	0	0
1	3.4	0	0	0	0	0	0	0	0	0	28	2	12	8	9	8	13	10	13	5	2	5	5	24	2	15	2	15	4	13	5	7	3	2	4	8	5	8	14	10	7	0	0	0	0	0	0	0	0	0
15	3.5	0	0	0	0	0	0	0	0	0	205	173	59	89	42	121	117	71	164	193	12	210	148	225	197	167	3	224	117	94	184	224	64	169	131	140	194	50	47	135	93	0	0	0	0	0	0	0	0	0
15	3.6	0	0	0	0	0	0	0	0	0	519	191	150	98	71	179	158	221	101	203	162	65	52	389	117	137	179	217	119	63	166	17	96	195	14	91	55	131	23	60	37	0	0	0	0	0	0	0	0	0
15	4.5	0	0	0	0	0	0	0	0	0	568	149	36	119	83	99	98	137	33	149	130	141	1	216	4	166	25	70	16	182	120	211	21	214	82	51	211	70	4	53	17	0	0	0	0	0	0	0	0	0
15	4.6	0	0	0	0	0	0	0	0	0	361	499	197	101	18	8	9	80	225	4	120	185	102	8	97	165	16	168	97	98	172	159	183	73	214	156	139	197	156	69	40	0	0	0	0	0	0	0	0	0
1	5.6	0	0	0	0	0	0	0	0	0	26	28	8	4	4	4	11	4	10	12	5	15	13	34	2	11	4	9	13	10	12	6	14	7	12	12	11	9	13	8	6	0	0	0	0	0	0	0	0	0
1	7.8	0	0	0	0	0	0	0	0	0	17	2	14	10	3	5	6	13	6	1	11	11	5	8	4	11	15	3	4	12	10	3	7	3	6	2	15	12	15	6	13	0	0	0	0	0	0	0	0	0
1	7.9	0	0	0	0	0	0	0	0	0	10	28	6	10	8	4	5	8	4	6	1	13	14	36	3	11	11	7	15	12	2	11	15	4	13	8	12	9	15	12	15	0	0	0	0	0	0	0	0	0
1	7.10	0	0	0	0	0	0	0	0	0	2	20	19	10	8	7	6	10	3	14	2	1	8	1	1	10	3	14	12	2	15	6	1	11	10	4	14	14	10	9	14	0	0	0	0	0	0	0	0	0
1	7.11	0	0	0	0	0	0	0	0	0	8	8	8	5	5	8	1	9	8	13	7	11	13	25	6	3	10	4	9	15	6	5	9	5	10	13	7	10	10	6	12	0	0	0	0	0	0	0	0	0
1	7.12	0	0	0	0	0	0	0	0	0	19	22	12	4	2	13	8	1	5	14	5	12	15	32	12	3	7	12	9	10	14	7	2	11	9	3	6	14	12	2	8	0	0	0	0	0	0	0	0	0
1	8.9	0	0	0	0	0	0	0	0	0	38	34	14	5	4	15	11	3	9	3	7	14	9	35	2	9	10	5	10	7	8	11	2	15	4	5	7	2	12	2	12	0	0	0	0	0	0	0	0	0
1	8.10	0	0	0	0	0	0	0	0	0	29	27	14	3	6	8	7	7	11	3	13	10	4	20	5	15	1	1	9	13	13	8	2	12	6	3	4	14	6	2	3	0	0	0	0	0	0	0	0	0
1	8.11	0	0	0	0	0	0	0	0	0	3	27	18	2	10	14	7	11	11	1	10	13	15	5	2	11	13	2	3	15	15	14	8	14	11	9	4	7	12	12	15	0	0	0	0	0	0	0	0	0
1	8.12	0	0	0	0	0	0	0	0	0	7	9	5	5	6	6	1	3	3	5	11	8	3	16	2	1	3	8	6	9	9	13	4	9	5	7	2	8	5	15	4	0	0	0	0	0	0	0	0	0
1	9.10	0	0	0	0	0	0	0	0	0	29	14	5	6	1	2	3	6	8	15	10	3	12	20	2	9	7	9	15	2	1	6	15	4	13	1	15	1	6	10	2	0	0	0	0	0	0	0	0	0
1	9.11	0	0	0	0	0	0	0	0	0	30	19	12	1	3	11	3	12	13	14	12	2	15	33	7	1	2	14	10	13	1	14	10	11	7	1	3	7	5	7	10	0	0	0	0	0	0	0	0	0
1	9.12	0	0	0	0	0	0	0	0	0	4	3	20	6	3	4	4	13	14	7	10	9	9	5	5	13	8	4	6	10	13	8	6	5	9	4	8	10	8	11	8	0	0	0	0	0	0	0	0	0
1	10.11	0	0	0	0	0	0	0	0	0	25	17	3	1	2	12	8	10	1	3	10	3	15	24	11	3	14	5	7	7	15	15	2	1	7	13	9	2	6	13	15	0	0	0	0	0	0	0	0	0
1	10.12	0	0	0	0	0	0	0	0	0	31	4	1	8	6	11	12	3	15	4	12	12	1	18	12	4	5	1	12	4	10	4	4	5	2	15	15	10	11	5	11	0	0	0	0	0	0	0	0	0
1	11.12	0	0	0	0	0	0	0	0	0	9	13	17	2	8	5	13	4	1	6	5	2	10	38	10	7	8	3	11	12	12	10	4	15	10	3	11	11	15	12	11	0	0	0	0	0	0	0	0	0

Tabla 12 Demanda en el Caso 3

5.2. Resultados

En las siguientes gráficas se muestra cómo se han distribuido los diferentes servicios en cada caso. En esta, el eje x indica el tiempo, mientras que el eje y indica la estación, siendo los terminales las estaciones 0 y 7 u 8, dependiendo de si se trata del caso 1, 2 o 3. Así pues, no todos los servicios van al terminal al terminar su servicio pues el tren que ha llevado a cabo ese servicio puede dar la vuelta y para realizar otro servicio en sentido contrario.

En el Caso 1 obtenemos los siguientes movimientos de los servicios:

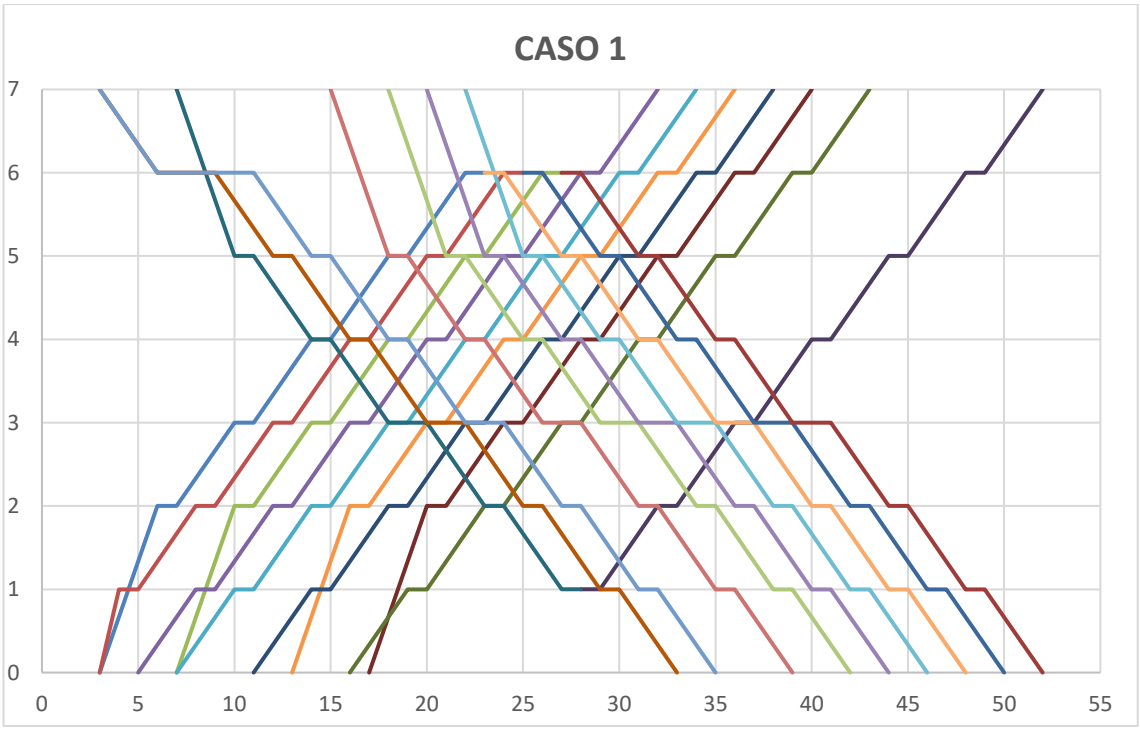

Figura 5 Movimientos de los Servicios en el Caso 1

Cómo se puede observar, la mayoría de los tramos van desde uno de los terminales hasta el otro pasando por todas o por casi todas las estaciones. Esto se observa en los segmentos en los que se encuentra parado el tren en alguna de las estaciones.

Además, podemos observar como algunos de los trenes que realizan el servicio no llegan a la terminal del final de su sentido. En estos casos, como ocurre, por ejemplo, en los servicios 1 y 18 los trenes realizan un giro en la última estación para atender un servicio en la dirección contraria. Los giros que se realizan, en este caso, son los siguientes:

Servicio 1	Servicio 18	Estación 6
Servicio 2	Servicio 19	Estación 6
Servicio 3	Servicio 20	Estación 6

Tabla 13 Giros en sentido aguas abajo Caso 1

En la tabla de arriba, indicamos que los trenes que llevan a cabo esos servicios en sentido aguas arriba, giran, en este caso todos en la estación 6, para realizar los servicios 18, 19 y 20, respectivamente.

Por otro lado, en sentido aguas abajo solo se realiza un giro:

Servicio 11	Servicio 10	Estación 1

Tabla 14 Giros en sentido aguas arriba Caso 1

Esto es que el tren que lleva a cabo el servicio 11 una vez llega al final de la línea, en vez de ir al terminal, hace un giro para realizar el servicio 10 en sentido contrario.

Tras la resolución del Caso 2 hemos obtenido los siguientes movimientos:

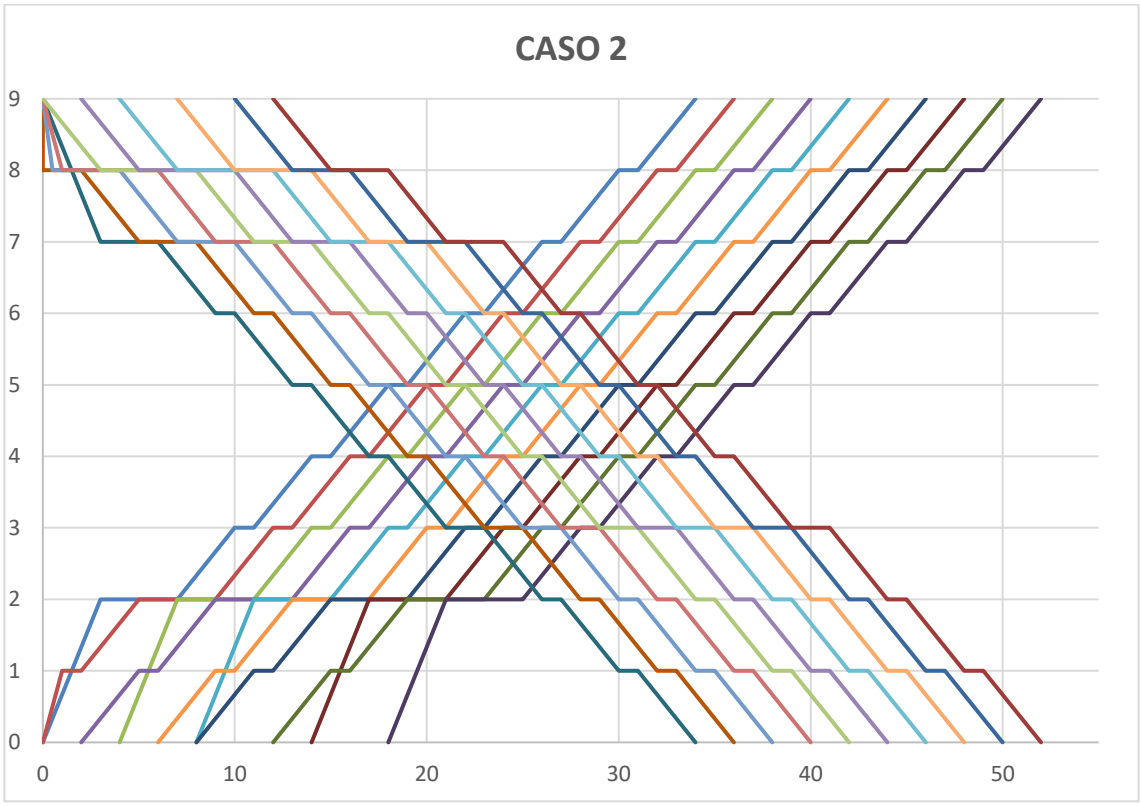

Figura 6 Movimientos de los Servicios en el Caso 2

Lo más característico del horario anterior es que, al aumentar el número de estaciones, y con ello, la demanda entre estaciones, el modelo distribuye a lo largo del tiempo todos los servicios, para abarcar el mayor número de pasajeros posibles tanto en un sentido como en el contrario.

Es interesante como, dado que el modelo trata de atender la mayor demanda posibles, , para el horizonte temporal estudiado no se realizan giros, pues se usa un mayor número de trenes al mismo tiempo para realizar los servicios. Si disminuyese la demanda y los tiempos entre estaciones fuesen menores, los servicios no necesitarían

abarcar todas las estaciones y los trenes en uso disminuirían, de forma que el modelo generará giros en algunas de las estaciones.

Los movimientos obtenidos en el caso 3 son los siguientes:

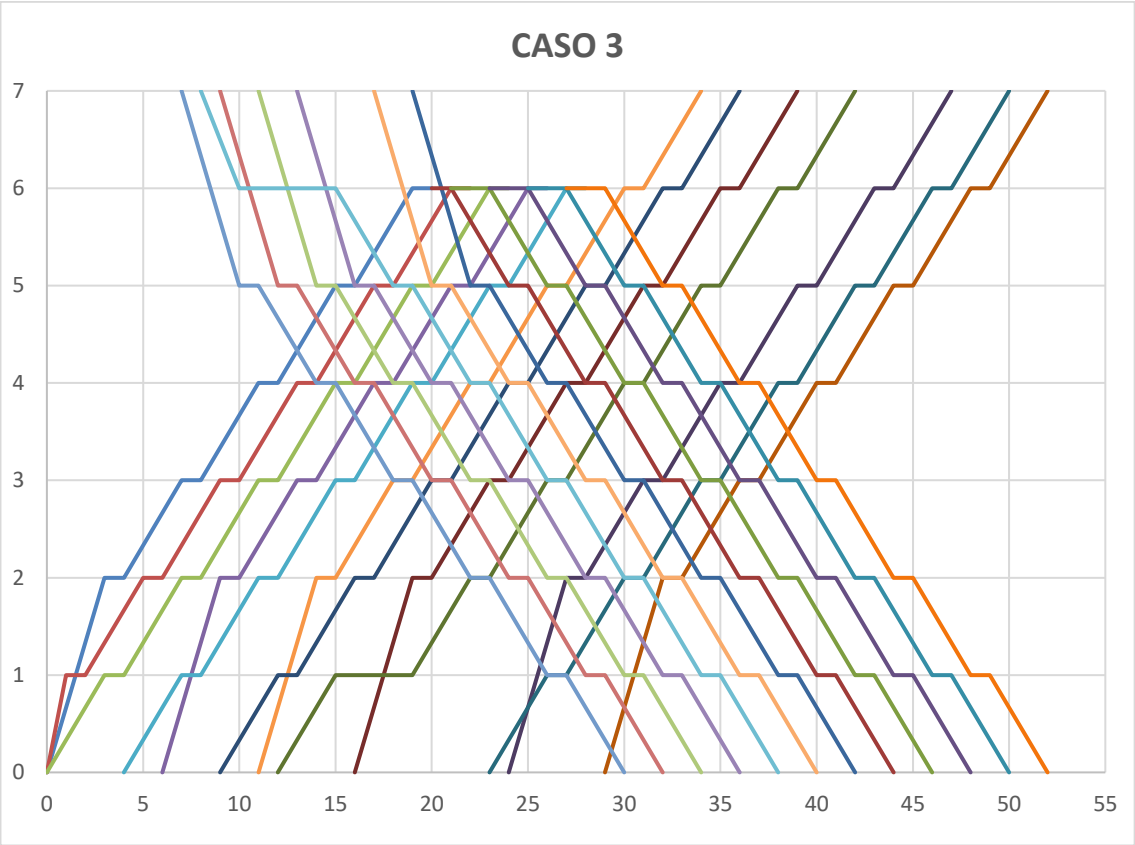

Figura 7 Movimientos de los Servicios en el Caso 3

En este caso, los servicios se vuelven a distribuir en a lo largo de la jornada con la peculiaridad de que vuelve a haber giros. En este caso, los giros son los siguientes:

Servicio 1	Servicio 20	Estación 6
Servicio 2	Servicio 21	Estación 6
Servicio 3	Servicio 22	Estación 6
Servicio 4	Servicio 23	Estación 6
Servicio 5	Servicio 24	Estación 6

Tabla 15 Giros en sentido aguas abajo Caso 3

En la tabla anterior, se pueden observar los giros que se producen al terminar un servicio en sentido aguas arriba,

para dar paso a un servicio en sentido aguas abajo. Por otro lado, tras la finalización de servicios en sentido aguas abajo, se realiza el siguiente giro:

Servicio 13	Servicio 12	Estación 1

Tabla 16 Giros en sentido aguas arriba Caso 3

En este caso se vuelve a repetir el patrón, de manera que el primer servicio en sentido aguas abajo es el único que gira para atender un servicio en la dirección aguas arriba. Esto se debe a que el modelo distribuye todos los servicios a lo largo de la línea, interesando solo los giros en aquellos servicios cuyo fin e inicio sean relativamente próximos.

Por último, es interesante comentar el tamaño del modelo que se genera en cada caso, así como el Gap final o el tiempo de computación, entre otros parámetros del modelo. Así pues, en Tabla 17 se puede observar estos parámetros.

	CASO 1	**CASO 2**	**CASO 3**
Tiempo de Computación	145,81 seconds	137,66 seconds	872,79 seconds
Gap Final	0,0004 %	0,0086 %	0,0000 %
Nº de Variables	10.302	14.732	12.650
Nº de Variables Enteras	8.122	11.192	10.034
Nº de Variables Binarias	8.020	11.050	9.912
Nº de Restricciones	11.715	16.667	14.927
Nº de Restricciones No Lineales	360	640	432
Nº de Términos No Lineales en la Función Objetivo	570	1064	690

Tabla 17 Parámetros del modelo en cada caso

6 CONCLUSIONES

A lo largo de este trabajo fin de grado se ha desarrollado un modelo de programación mixta entera para el diseño de horarios en redes de ferrocarril de tránsito rápido que permite diseñar el horario óptimo para una demanda variable con el tiempo. El modelo incorpora además las estrategias de aceleración más populares, Stop-Skipping y Short-Turning, seleccionando de forma automática la zona más adecuada de operación para los servicios de corto recorrido. De esta forma, los horarios producidos no solon adapatdan la separación (headway) entre trenes para adaptar las salidas y llegadas de las estaciones a los instantes más convenientes para la captura de los pasajeros sino que además, si resulta conveniente algunos de los trenes pueden pasar de largo por estaciones de baja demanda para llegar con mayor prontitud a estaciones con mayor demanda de pasajeros y algunos servicios se pueden mover entre estaciones que no son las estaciones terminales de la línea para de esta forma proporcionar una mayor frecuencia de paso entre las estaciones que soportan un mayor tráfico.

La formulación del modelo da lugar a una función objetivo que contiene términos no lineales y a restricciones no lineales que son convenientemente linealizadas para garantizar la obtención de soluciones óptimas.

Desde un punto de vista computacional el modelo se implementa en lenguaje Python 3.8 utilizando las API (Application Program Interface) de Python para el Solver GUROBI.

Los resultados obtenidos demuestran la capacidad del modelo para generar horarios que se adaptan a la demanda al tiempo que, cuando resulta conveniente, genera soluciones que contienen las estrategias de aceleración oportunas, Stop-skipping, Short-turning o ambas.

Desde el punto de vista teórico, el modelo responde perfectamente a las especificaciones que se plantearon como objetivo. Como desventaja, cabe citar que el tamaño del modelo extendido para instancias pequeñas como las mostradas en la sección de experimentos es importante, mirar Tabla 17, lo que obligaría a buscar un método alternativo aproximado de solución, posiblemente usando alguna metaheurística, como por ejemplo un algoritmo genético, en caso de instancias de mayor tamaño. En cualquier caso, permitiría analizar la calidad de las soluciones a los problemas obtenidas usando la mateheurística por comparación con las soluciones optimas encontradas tras resolver el modelo matemático.

REFERENCIAS

[1] Berbey-Alvarez, A. 2013. Trenes: material rodante del transporte ferroviario. *Revista Prisma Tecnológico*. 4, 1 (2013).

[2] Billionnet, A. 2003. Using integer programming to solve the train platforming problem. *Transportation Science*. 37, 2 (2003).

[3] Bolívar García, F. 2017. *Diseño de horarios de redes metropolitanas de transporte ferroviario que minimizan el consumo energético*. Trabajo fin de grado. Universidad Politécnica de Madrid.

[4] Canca, D., Santos, A. De los, Laporte, G. and Mesa, J. 2019. Integrated railway rapid transit network design and line planning problem with maximum profit. Transportation Research Part E: Logistics and Transportation Review, 127, (2019), 1–30.

[5] Caprara, A., Kroon, L., Monaci, M., Peeters, M. and Toth, P. 2007. Passenger railway optimization. Handbooks in Operations Research and Management Science, Volume 14, Chapter 3. 129-187.

[6] Casanova, J.L. 1902. Vademécum del interventor del estado en la explotación de los ferrocarriles. *Imprenta Alemana, Madrid*. (1902).

[7] Guihaire, V. and Hao, J. 2008. Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice*. 42, 10 (2008), 1251–1273.

[8] Ben-Khedher, N., Kintanar, J., Queille, C. and Stripling, W. 1998. Schedule optimization at sncf: From conception to day of departure. *INFORMS Journal on Applied Analytics*. 28, 1 (1998).

[9] Lai, C., Shih, T., Ko, W., Tang, H. and Hsueh, P. 2020. Severe acute respiratory syndrome coronavirus 2 (sars-cov-2) and: The epidemic and the challenges. *International Journal of Antimicrobial Agents*. 55, 3 (2020).

[10] Larson, R. and Odoni, A. 1981. *Urban operations research*. Prentice-Hall: New York.

[11] Lingaya, N., Cordeau, J., Desaulniers, G., Desrosiers, J. and Soumis, F. 2002. Operational car assignment at via rail canada. *Transportation Research 36*. 36, 9 (2002).

[12] De Luca Cardillo, D. and Mione, N. 1998. k l -list τ colouring of graphs. *European Journal of OperationalResearch*. 106, 1 (1998).

[13] Narayanaswami, S. and Rangaraj, N. 2011. Scheduling and rescheduling of railway operations: A review and expository analysis. *Technology Operation Management*. 2, 2 (2011).

[14] Peña, O. 2017. *Circulación optima de vehículos y localización de depósitos en redes ferroviarios de tránsito rápido. Proyecto fin de carrera.* Universidad de Sevilla.

[15] Portales, D. 2010. El gran impacto del metro. Tesis doctoral inédita. *EURE: revista latinoamericana de estudios urbano regionales*. 36, 107 (2010).

[16] Pozo Montaño, M.A. 2015. *Mathematical models for the design and planning of transportation on demand in urban logistics networks.* Tesis Doctoral. Universidad de Sevilla.

[17] Ramos Mate, M. 2016. *Cálculo óptimo de frecuencias y capacidad con mínimo consumo energético en líneas de transporte público por ferrocarril. Prroyecto fin de carrera.* Universidad de Sevilla.

[18] Vuchic, V. 2005. *Urban transit: operations, planning, and economics*.

[19] Wu, D.D. and Olson, D.L. 2020. Pandemic risk management in operations and finance modeling the impact of covid-19 (1st ed. 2020.). *Springer International Publishing*. (2020).

[20] Yuan, J., Gao, Y., Li, S., Liu, P. and Yang, L. 2020. Integrated optimization of train timetable, rolling stock assignment and short-turning strategy for a metro line. *European Jorunal of Operational Research*.

299, 1 (2020).

[21] *Ley 38/2015, de 29 de septiembre, del sector ferroviario. Jefatura del estado. B.O.E. B.O.E.. Núm. 234, de 30 de septiembre de 2015.*

ANEXO

Modelo matemático, escrito en Python y resuelto con Gurobi Optimizer.

```
        from openpyxl import *
        from IOfunctionsExcel import *
        from gurobipy import *

        name = "TFG_Septiembre.xlsx"
        excel_document = openpyxl.load_workbook(name ,data_only=True)

        "Lectura de Datos"
        sheet = excel_document['Datos']

        "Tiempo entre estación y estación predeterminado"
        s = Read_Excel_to_NesteDic(sheet, 'B58', 'N70')
        "Adelanto mínimo entre dos servicios consecutivos"
        hmin = Read_Excel_to_NesteDic(sheet, 'F46', 'G47')
        "Adelanto máximo entre dos servicios consecutivos"
        hmax = Read_Excel_to_NesteDic(sheet, 'F44', 'G45')
        "Tiempo máximo en el cambio de sentido en cada andén de alightingmbio
de sentido"
        deltamax = Read_Excel_to_NesteDic(sheet, 'Q56', 'R64')
        "Tiempo mínimo necesario para el cambio de sentido en alightingda
andén de alightingmbio de sentido"
        deltamin = Read_Excel_to_NesteDic(sheet, 'Q68', 'R76')
        "Número de estaciones totales"
        N = Read_Excel_to_NesteDic(sheet, 'K44', 'L45')
        "Número de trenes disponibles para ser utilizados"
        RS = Read_Excel_to_NesteDic(sheet, 'K50', 'L51')
        "capacidad del tren"
        CAPACITY = Read_Excel_to_NesteDic(sheet, 'K53', 'L54')
        "Número muy grande"
        M = Read_Excel_to_NesteDic(sheet, 'K47', 'L48')

        "Todos los andenes"
        S = Read_Excel_to_NesteDic(sheet, 'T38', 'T50')
        "Todos los andenes con cammbio de sentido"
        Sm = Read_Excel_to_NesteDic(sheet, 'X33', 'X37')

        "Estaciones en sentido upstream"
        Su = Read_Excel_to_NesteDic(sheet, 'X45', 'X51')
        "Estaciones en sentido downstream"
        Sd = Read_Excel_to_NesteDic(sheet, 'Z45', 'Z51')

        "Estaciones con cambio de sentido en sentido upstream"
        Smu = Read_Excel_to_NesteDic(sheet, 'AB38', 'AB42')
        "Estaciones con cambio de sentido en sentido downstream"
        Smd = Read_Excel_to_NesteDic(sheet, 'AD38', 'AD42')


        "Todos los servicios"
        Linea = Read_Excel_to_NesteDic(sheet, 'Q33', 'Q53')
        "Servicios en sentido upstream"
        ku = Read_Excel_to_NesteDic(sheet, 'Q33', 'Q43')
        "Servicios en sentido downstream"
        kd = Read_Excel_to_NesteDic(sheet, 'Q43', 'Q53')
```

```
        Scala=15.0
        Doors=4
        Tasa=0.5 # seg/persona subiendo o bajando

        sheet2 = excel_document['Demanda']
        "Instantes de tiempo (Columnas matriz OD)"
        tiempo= Read_Excel_to_NesteDic(sheet2, 'B39', 'AY40')
        "Viajes (Filas matriz OD en sentido upstream)"
        viajes_u= Read_Excel_to_NesteDic(sheet2, 'B45', 'G50')
        "Viajes (Filas matriz OD en sentido upstream)"
        viajes_d= Read_Excel_to_NesteDic(sheet2, 'L45', 'Q50')
        "Demada. Matriz ODij[t]"
        OD= Read_Excel_to_NesteDic(sheet2, 'B7', 'AY37')

        tr = Model('TFG')

        #Variables
        #Definimos las variables

        #****************************************
        dwell={}
        "Definimos la espera (dwell[i,k]) como variable, a ver si así sale
compatible"
        for k in ku:
            for i in Su:
                dwell[k,i] = tr.addVar(lb=0.0,ub=GRB.INFINITY, vtype=
GRB.CONTINUOUS, name="dwell[%s,%s]" %(k,i))

        for k in kd:
            for i in Sd:
                dwell[k,i] = tr.addVar(lb=0.0,ub=GRB.INFINITY, vtype=
GRB.CONTINUOUS, name="dwell[%s,%s]" %(k,i))

        #****************************************

        "Si el servicio k es seleccionado vale 1 y 0 en otro alightingso"

        # Línea es el conjunto de servicios tanto up como down
        tau = {}
        for i in Linea:
            tau[i] = tr.addVar(lb=0.0,ub=1.0, vtype= GRB.BINARY,
name="tau[%s]" %(i))

        #****************************************


        "Vale 1 si los andenes del principio y del final, para el servicio k,
son m y n, y vale 0 e.o.c."
        zone={}
        for j in ku:
            for m in Smu:
                for n in Smu:
                    if n > m: # Creo que debe ser n > m
                        zone[j,m,n] = tr.addVar(lb=0.0, ub=1.0, vtype=
GRB.BINARY, name="zone[%s,%s,%s]" %(j,m,n))
        for j in kd:
            for m in Sd:
                for n in Smd:
                    if n > m: # Creo que debe ser n > m
                        zone[j, m, n] = tr.addVar(lb=0.0, ub=1.0,
vtype=GRB.BINARY, name="zone[%s,%s,%s]" % (j, m, n))
```

```
        #****************************************

        "Vale 1 si el servicio k llega al andén i y 0 e.o.c."
        lambdaa={}
        for k in ku:
            for i in Su:
                lambdaa[k,i] = tr.addVar(lb=0.0, ub=1, vtype=GRB.BINARY,
name="lambdaa[%s,%s]" %(k,i))
        for k in kd:
            for i in Sd:
                lambdaa[k,i] = tr.addVar(lb=0.0, ub=1, vtype=GRB.BINARY,
name="lambdaa[%s,%s]" %(k,i))

        #****************************************
        "Headway entre el servicio k y el k-1"
        h={}
        for o,k in enumerate(ku):
            if o < len(ku):
                h[k, k + 1] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="h[%s,%s]" % (k, k+1))

        for o, k in enumerate(kd):
            if o < len(kd):
                h[k, k + 1] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="h[%s,%s]" % (k, k+1))

        #****************************************
        "Tiempo de llegada del servicio k a la estación i"
        a={}
        for k in ku:
            for i in Su:
                a[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="a[%s,%s]" %(k,i))
        for k in kd:
            for i in Sd:
                a[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="a[%s,%s]" %(k,i))

        #****************************************

        "Tiempo de salida del servicio k a la estación i"
        d={}
        for k in ku:
            for i in Su:
                d[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="d[%s,%s]" %(k,i))

        for k in kd:
            for i in Sd:
                d[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="d[%s,%s]" %(k,i))

        #****************************************

        "Vale 1 si el servicio k termina en la estación m y da la vuelta para
empezar el servicio l"
        gamma={}
        for k in ku:
            for l in kd:
                for m in Su:
                    gamma[k,l,m] = tr.addVar(lb=0.0, ub=1.1,
vtype=GRB.BINARY, name="gamma[%s,%s,%s]" %(k,l,m))
```

```
        for k in kd:
            for l in ku:
                for m in Sd:
                    gamma[k,l,m] = tr.addVar(lb=0.0, ub=1.1,
vtype=GRB.BINARY, name="gamma[%s,%s,%s]" %(k,l,m))

        #****************************************

        "Vale 1 si el servicio k en sentido upstream se realiza con un tren
que viene del depot 1 (principio de la línea upstream) y 0 e.o.c."
        from_depot1={}
        for k in ku:
            from_depot1[k] = tr.addVar(lb=0.0, ub=1.0, vtype=GRB.BINARY,
name="from_depot1[%s]" %(k))
        "Vale 1 si el  servicio k en sentido downstream viene del depot 2
(final del sentido) y 0 e.o.c."
        from_depot2={}
        for k in kd:
            from_depot2[k] = tr.addVar(lb=0.0, ub=1.0, vtype=GRB.BINARY,
name="from_depot2[%s]" %(k))
        "Vale 1 si un tren que termina en sentido upstream vuelve al depot 2
y 0 e.o.c."
        to_depot2={}
        for k in ku:
            to_depot2[k] = tr.addVar(lb=0.0, ub= 1.0, vtype=GRB.BINARY,
name="to_depot2[%s]" %(k))
        "Vale 1 si un tren que termina en sentido downstream vuelve al depot
1 y 0 e.o.c."
        to_depot1={}
        for k in kd:
            to_depot1[k] = tr.addVar(lb=0.0, ub=1.0, vtype=GRB.BINARY,
name="to_depot1[%s]" %(k))

        #****************************************

        "Número de trenes utilizados en sentido upstream"
        RS1 = tr.addVar(lb=0.0, ub=GRB.INFINITY, vtype=GRB.INTEGER,
name="RS1")
        "Número de trenes utilizados en sentido downstream"
        RS2 = tr.addVar(lb=0.0, ub=GRB.INFINITY, vtype=GRB.INTEGER,
name="RS2")

        #****************************************
        "Variables de demanda"
        "Pasajeros esperando al servicio k en la estación i"
        waiting={}
        for k in ku:
            for i in Su:
                waiting[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="waiting[%s,%s]" %(k,i))

        for k in kd:
            for i in Sd:
                waiting[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="waiting[%s,%s]" %(k,i))

        "Pasajeros esperando al servicio k en la estación i con destino j"
        waiting_to_j={}
        for k in ku:
            for i in Su:
                for j in Su:
```

```
                    if  i < j: # Solo si j es mayor que i
                        waiting_to_j[k,i,j] = tr.addVar(lb=0.0,
ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="w2_[%s,%s,%s]" %(k,i,j))

        for k in kd:
            for i in Sd:
                for j in Sd:
                    if i<j: # Solo si j es mayor que i
                        waiting_to_j[k,i,j] = tr.addVar(lb=0.0,
ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="w2_[%s,%s,%s]" %(k,i,j))

        # w1 es la suma de los w2

        "Pasajeros que potencialmente suben al servicio k en la estación i"
        waiting_pot={}
        for k in ku:
            for i in Su:
                waiting_pot[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="waiting_pot[%s,%s]" %(k,i))

        for k in kd:
            for i in Sd:
                waiting_pot[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="waiting_pot[%s,%s]" %(k,i))

        # wb1 es la suma de los wb2 (potencialmente podrían subir - cuando la
estacion i es servida por el servicio k ---
        "Pasajeros que potencialmente suben al servicio k en la estación i
con destino a la estación j"
        waiting_pot_to_j={}
        for k in ku:
            for i in Su:
                for j in Su:
                    if i<j: # Solo si j es mayor que i
                        waiting_pot_to_j[k,i,j] = tr.addVar(lb=0.0,
ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="waiting_pot_to_j[%s,%s,%s]"
%(k,i,j))

        for k in kd:
            for i in Sd:
                for j in Sd:
                    if i<j: # Solo si j es mayor que i
                        waiting_pot_to_j[k,i,j] = tr.addVar(lb=0.0,
ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="waiting_pot_to_j[%s,%s,%s]"
%(k,i,j))


        "Número de pasajeros que no pueden subir al sevicio k desde la
estadción i con destino la estación j"
        # wr son los residuos
        Residuo={}
        for k in ku:
            for i in Su:
                for j in Su:
                    if i<j: # Solo si j es mayor que i
                        Residuo[k,i,j] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="Residuo[%s,%s,%s]" %(k,i,j))

        for k in kd:
            for i in Sd:
                for j in Sd:
                    if i<j: # Solo si j es mayor que i
```

```
                        Residuo[k,i,j] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="Residuo[%s,%s,%s]" %(k,i,j))

        "Número de pasajeros que se montan en el servicio k en la estación i"
        boarding={}
        for k in ku:
            for i in Su:
                boarding[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="boarding[%s,%s]" %(k,i))

        for k in kd:
            for i in Sd:
                boarding[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="boarding[%s,%s]" %(k,i))

        # De nuevo boarding es la suma de los boarding_to_j (boarding_to_j
incluye el destino)
        "Número de pasajeros que se montan en el servicio k en la estación i
con destino j"
        boarding_to_j={}
        for k in ku:
            for i in Su:
                for j in Su:
                    if i<j: # Solo si j es mayor que i
                        boarding_to_j[k,i,j] = tr.addVar(lb=0.0,
ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="boarding_to_j[%s,%s,%s]"
%(k,i,j))

        for k in kd:
            for i in Sd:
                for j in Sd:
                    if i<j: # Solo si j es mayor que i
                        boarding_to_j[k,i,j] = tr.addVar(lb=0.0,
ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="boarding_to_j[%s,%s,%s]"
%(k,i,j))

        "Número de pasajeros que abandonan el servicio k en la estación i"
        alighting={}
        for k in ku:
            for i in Su:
                alighting[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="alighting[%s,%s]" %(k,i))

        for k in kd:
            for i in Sd:
                alighting[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="alighting[%s,%s]" %(k,i))

        "Número de pasajeros que hay en el servicio k en la estación i"
        onboard={}
        for k in ku:
            for i in Su:
                onboard[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="onboard[%s,%s]" %(k,i))

        for k in kd:
            for i in Sd:
                onboard[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY,
vtype=GRB.CONTINUOUS, name="onboard[%s,%s]" %(k,i))


#*****************************************************************************
```

```
        "Variable auxiliar para la restricción 43. Vale 1 si el tren k sale
de i en t y 0 e.o.c."
        y={}
        for k in ku:
            for i in Su:
                for t in tiempo[1]:
                    y[k,i,t]=tr.addVar(lb=0.0, ub=1.0, vtype=GRB.BINARY,
name="y[%s,%s,%s]" %(k,i,t))

        for k in kd:
            for i in Sd:
                for t in tiempo[1]:
                    y[k,i,t]=tr.addVar(lb=0.0, ub=1.0, vtype=GRB.BINARY,
name="y[%s,%s,%s]" %(k,i,t))


        "Variable auxiliar para la restricción 44. Vale 1 si lambdaa[k,i]=1 y
lambdaa[k,j]=1"
        xi={}
        for k in ku:
            for i in Su:
                for j in Su:
                    if i!=j:
                        xi[k,i,j]=tr.addVar(lb=0.0, ub=1.0, vtype=GRB.BINARY,
name="xi[%s,%s,%s]" %(k,i,j))

        for k in kd:
            for i in Sd:
                for j in Sd:
                    if i!=j:
                        xi[k,i,j]=tr.addVar(lb=0.0, ub=1.0, vtype=GRB.BINARY,
name="xi[%s,%s,%s]" %(k,i,j))

        "COMPUTO DE CANTIDADES AUXILIARES PARA PORCENTAJES DE REPARTO ENTRE
ESTACIONES" \
        "Auxiliar de python para la restricción (48)"
        O2_u={}
        for i in Su:
            for j in Su:
                if i < j:
                    O2_u[i, j] = 0
                    for t in tiempo[1]:
                        O2_u[i,j]=O2_u[i,j]+OD[viajes_u[i][j]][t]
        O1_u={}
        for i in Su:
            O1_u[i]=0
            for j in Su:
                if i<j:
                    O1_u[i]=O1_u[i]+O2_u[i,j]

        "Variables auxiliar de python para la restricción (58)"
        O2_d={}
        for i in Sd:
            for j in Sd:
                if i < j:
                    O2_d[i, j] = 0
                    for t in tiempo[1]:
                        O2_d[i,j]=O2_d[i,j]+OD[viajes_d[i][j]][t]
        O1_d={}
        for i in Sd:
            O1_d[i]=0
            for j in Sd:
```

```
                if i<j:
                    O1_d[i]=O1_d[i]+O2_d[i,j]


"Restricción (6)"
for k in ku:
    for m in Smu:
        tr.addConstr(quicksum(lambdaa[k,i] for i in Su if i < m) <= M[1][1] *
(1-quicksum(zone[k,m,n] for n in Smu if m<n)),
name="Restricción(6)[k,m][%s,%s]" %(k,m)) ;" & m<n"

"Restricción (7)"
for k in ku:
    for n in Smu:
        tr.addConstr(quicksum(lambdaa[k,i] for i in Su if i > n ) <= M[1][1]
* (1-quicksum(zone[k,m,n] for m in Smu if m<n)),
name="Restricción(7)[k,n][%s,%s]" %(k,n))
#*************Lo mismo que (5) a (7) pero en sentido contrario
************************************************
"Restricción (9)"
for k in kd:
    for i in Sd:
        tr.addConstr(lambdaa[k,i] <= tau[k],
name="Restricción(9)[k,i][%s,%s]" %(k,i))

"Restricción (10)"
for k in kd:
    for m in Smd:
        tr.addConstr(quicksum(lambdaa[k,i] for i in Sd if i < m) <= M[1][1] *
(1-quicksum(zone[k,m,n] for n in Smd if n>m)),
name="Restricción(10)[k,m][%s,%s]" %(k,m))

"Restricciones (11)"
for k in kd:
    for n in Smd:
        tr.addConstr(quicksum(lambdaa[k,i] for i in Sd if i > n) <= M[1][1] *
(1 - quicksum(zone[k,m,n] for m in Smd if m<n)),
name="Restricción(11)[k,n][%s,%s]" %(k,n))

#*************************************************************************
*************************

"Restricción (13)" # Opcional, al menos uno de cada dos servicios
for o,k in enumerate(ku):
    for i in Su:
        if o!=0:
            tr.addConstr(lambdaa[k-1,i]+lambdaa[k,i]>=1, name=
"Restricción(13)[k,i][%s,%s]" %(k,i))

"Restricción (14)"
for o,k in enumerate(kd):
    for i in Sd:
        if o!=0:
            tr.addConstr(lambdaa[k-1,i]+lambdaa[k,i]>=1, name=
"restricción(14)[k,i][%s,%s]" %(k,i))

#*************************************************************************
************************
# Llegadas y salidas
"Restricción (15)" # Ojo, es - no +

for k in ku:
```

```
    for o,i in enumerate(Su):
        if o!=0:
            tr.addConstr(a[k,i] - d[k,i-1] == s[i-1][i],
name='Restricción(15)[k,i][%s,%s]' %(k,i))

"Restricción (16)"
for k in kd:
    for o,i in enumerate(Sd):
        if o!=0:
            tr.addConstr(a[k,i] - d[k,i-1] == s[i-1][i],
name='Restricción(16)[k,i][%s,%s]' %(k,i))

#*****************************************************

"Restricción (17)"
for k in ku:
    for i in Su:
        tr.addConstr(d[k,i] - a[k,i] == dwell[k,i],
name='Restricción(17)[k,i][%s,%s]' %(k,i))

"Restricción (18)"
for k in kd:
    for i in Sd:
        tr.addConstr(d[k,i] - a[k,i] == dwell[k,i],
name='Restricción(18)[k,i][%s,%s]' %(k,i))

#*************************************************************************
**************
MinDwell=1.0
MaxDwell=5.0
"Restricción (17.b)"
for k in ku:
    for i in Su:
        tr.addConstr(dwell[k,i] <= lambdaa[k,i] * MaxDwell,
name='Restricción(17.b)[k,i][%s,%s]' %(k,i))
        tr.addConstr(dwell[k, i] >=  lambdaa[k, i] * MinDwell,
name='Restricción(17.b)[k,i][%s,%s]' % (k, i))

"Restricción (18.b)"
for k in kd:
    for i in Sd:
        tr.addConstr(dwell[k,i] <= lambdaa[k,i] * MaxDwell,
name='Restricción(18.b)[k,i][%s,%s]' %(k,i))
        tr.addConstr(dwell[k, i] >= lambdaa[k, i] *
(boarding[k,i]+alighting[k,i]) *(Tasa/(60*Doors)),
name='Restricción(17.b)[k,i][%s,%s]' % (k, i))

#*************************************************************************
**********************
"Restricción (19.1)"
for o,k in enumerate(ku):
    if o!=0:
        tr.addConstr(hmin[1][1] * tau[k] <= h[k-1,k],
name='Restricción(19.1)[k-1,k][%s]' %(k))

"Restricción (19.2)"
for o,k in enumerate(ku):
    if o!=0:
        tr.addConstr(h[k-1,k] <= hmax[1][1] * tau[k],
name='Restricción(19.2)[k-1,k][%s]' %(k))
```

```
"Restricción (20.1)"
for o,k in enumerate(kd):
    if o!=0:
        tr.addConstr(hmin[1][1] * tau[k] <= h[k-1,k],
name='Restricción(20.1)[k-1,k][%s]' %(k))

"Restricción (20.2)"
for o,k in enumerate(kd):
    if o!=0:
        tr.addConstr(h[k-1,k] <= hmax[1][1] * tau[k],
name='Restricción(20.2)[k-1,k][%s]' %(k))

#*************************************************************************
************************

"Restricciones 21 y 22"

"Restricción (21)"
for o,k in enumerate(ku):
    for i in Su:
        if o!=0:
            tr.addConstr(d[k,i] == d[k-1,i] + h[k-1,k],
name="Restricción(21)[k,i][%s,%s]" %(k,i))

"Restricción (22)"
for o,k in enumerate(kd):
    for i in Sd:
        if o!=0:
            tr.addConstr(d[k,i] == d[k-1,i] + h[k-1,k],
name="Restricción(22)[k,i][%s,%s]" %(k,i))

#************************************************

"Restricción (43a)"
for k in ku:
    for i in Su:
        tr.addConstr(d[k,i] == quicksum( (y[k,i,t] - y[k,i,t-1] )* t for t in
tiempo[1] if t >1), name="Restricción(43.a)[k,i][%s,%s]" %(k,i))


"Restricción (53a)"
for k in kd:
    for i in Sd:
        tr.addConstr(d[k,i] == quicksum( (y[k,i,t] - y[k,i,t-1] ) * t for t
in tiempo[1] if t >1), name="Restricción(53.a)[k,i][%s,%s]" %(k,i))

#*************************************************

"Restricción (43a1)"
for k in ku:
    for i in Su:
        for o,t in enumerate(tiempo[1]):
            if o>0:
                tr.addConstr(y[k,i,t-1] <= y[k,i,t],
name="Restricción(43.a1)[k,i,t][%s,%s,%s]" % (k, i,t))

"Restricción (53a1)"
for k in kd:
    for i in Sd:
        for o,t in enumerate(tiempo[1]):
            if o>0:
                tr.addConstr( y[k,i,t-1] <= y[k,i,t],
```

```
name="Restricción(53.a1)[k,i,t][%s,%s,%s]" % (k, i, t))

# ***********************************************

#*************************************************************************
*****
"Restricción (23)"
for k in ku:
    for l in kd:
        for m in Smu:
            tr.addConstr(2 * gamma[k,l,m] <= quicksum(zone[k,i,m] for i in
Smu if i < m) + quicksum(zone[l,len(S)+1-m,i] for i in Smd if (len(S)+1-m) <
i), name='Restricción(23)[k,l,m][%s,%s,%s]' %(k,l,m))

"Restricción (24)"
for k in ku:
    for l in kd:
        for m in Smu:
            tr.addConstr(a[l,len(S)+1-m]-d[k,m] >= deltamin[m][1] + M[1][1] *
(gamma[k,l,m]-1), name='Restricción(24)[k,l,m][%s,%s,%s]' %(k,l,m))


"Restricción (25)"
for k in ku:
    for l in kd:
        for m in Smu:
            tr.addConstr(a[l,len(S)+1-m] - d[k,m] <= deltamax[m][1]- M[1][1]
* (gamma[k,l,m]-1), name='Restricción(25)[k,l,m][%s,%s,%s]' %(k,l,m))

"Restricción (26)"
for k in kd:
    for l in ku:
        for m in Smd:
            tr.addConstr(2 * gamma[k,l,m] <= quicksum(zone[k,i,m] for i in
Smd if i < m) + quicksum(zone[l,len(S)+1-m,i] for i in Smu if len(S)+1-m <
i), name='Restricción(26)[k,l,m][%s,%s,%s]' %(k,l,m))

"Restricción (27)"
for k in kd:
    for l in ku:
        for m in Smd:
            tr.addConstr(a[l,len(S)+1-m] - d[k,m] >= deltamin[m][1] + M[1][1]
* (gamma[k,l,m]-1), name='Restricción(27)[k,l,m][%s,%s,%s]' %(k,l,m))

"Restricción (28)"
for k in kd:
    for l in ku:
        for m in Smd:
            tr.addConstr(a[l,len(S)+1-m] - d[k,m] <= deltamax[m][1] - M[1][1]
* (gamma[k,l,m]-1), name='Restricción(28)[k,l,m][%s,%s,%s]' %(k,l,m))


"Restricción (29)"
for k in ku:
    tr.addConstr(quicksum(quicksum(gamma[l,k,m] for l in kd) for m in Smd) +
from_depot1[k] == tau[k], name='Restricción(29)[k][%s]' %(k))


"Restricción (30)"
for k in kd:
    tr.addConstr(quicksum(quicksum(gamma[l,k,m] for l in ku) for m in Smu) +
from_depot2[k] == tau[k], name='Restricción(30)[k][%s]' %(k))
```

```
"Restricción (31)"
for k in ku:
    tr.addConstr(quicksum(quicksum(gamma[k,l,m] for l in kd) for m in Smu) +
to_depot2[k] == tau[k], name='Restricción(31)[k][%s]' %(k))

"Restricción (32)"
for k in kd:
    tr.addConstr(quicksum(quicksum(gamma[k,l,m] for l in ku) for m in Smd) +
to_depot1[k] == tau[k], name='Restricción(32)[k][%s]' %(k))

#*****************************************************************************

"Restricción (33)"
tr.addConstr(RS1 >= quicksum(from_depot1[k] for k in ku),
name='Restricción(33)')

"Restricción (34)"
tr.addConstr(RS2 >= quicksum(from_depot2[k] for k in kd),
name='Restricción(34)')

"Restricción (35)"
tr.addConstr(RS1 + RS2 <= RS[1][1], name='Restricción(35)')

#*****************************************************************************

"Restricción (36)"
for k in ku:
    tr.addConstr(from_depot1[k] <= quicksum(zone[k,1,n] for n in Smu if
1!=n), name='Restricción(36)[k][%s]' %(k))

"Restricción (37)"
for k in ku:
    tr.addConstr(to_depot2[k] <= quicksum(zone[k,n,len(Su)] for n in Smu if n
!= len(Su)), name='Restricción(37)[k][%s]' %(k))

"Restricción (38)"
for k in kd:
    tr.addConstr(from_depot2[k] <= quicksum(zone[k,len(Su)+1,n] for n in Smd
if len(Su)+1!=n), name='Restricción(38)[k][%s]' %(k))

"Restricción (39)"
for k in kd:
    tr.addConstr(to_depot1[k] <= quicksum(zone[k,n,len(S)] for n in Smd if n
!= len(S)), name='Restricción(39)[k][%s]' %(k))

#*****************************************************************************

"Restricciones de Demanda en sentido Upstream"

"Restricción (40)"
for k in ku:
    for i in Su:
        tr.addConstr(waiting[k,i] == quicksum(waiting_to_j[k,i,j] for j in Su
if j>i), name='Restricción(40)[k,i][%s,%s]' %(k,i))

"Restricción (41)"
for k in ku:
    for i in Su:
        tr.addConstr(waiting_pot[k,i] == quicksum(waiting_pot_to_j[k,i,j] for
j in Su if j>i), name='Restricción(41)[k,i][%s,%s]' %(k,i))
```

```
"Restricción (42)"
for k in ku:
    for i in Su:
        tr.addConstr(boarding[k,i] == quicksum(boarding_to_j[k,i,j] for j in
Su if j>i), name='Restricción(42)[k,i][%s,%s]' %(k,i))

#******************** Mismas en sentido DW *********************************
"Restricción (50)"
for k in kd:
    for i in Sd:
        tr.addConstr(waiting[k,i] == quicksum(waiting_to_j[k,i,j] for j in Sd
if j>i), name='Restricción(50)[k,i][%s,%s]' %(k,i))

"Restricción (51)"
for k in kd:
    for i in Sd:
        tr.addConstr(waiting_pot[k,i] == quicksum(waiting_pot_to_j[k,i,j] for
j in Sd if j>i), name='Restricción(51)[k,i][%s,%s]' %(k,i))

"Restricción (52)"
for k in kd:
    for i in Sd:
        tr.addConstr(boarding[k,i] == quicksum(boarding_to_j[k,i,j] for j in
Sd if j>i), name='Restricción(52)[k,i][%s,%s]' %(k,i))

#*********************************************************************
"Restricción (44.a)"
for k in ku:
    for i in Su:
        for j in Su:
            if i<j:
                tr.addConstr(lambdaa[k,i] + lambdaa[k,j] <= 1 + xi[k,i,j],
name="Restricción(44.a)[k,i,j][%s,%s,%s]" %(k,i,j))

"Restricción (44.b)"
for k in ku:
    for i in Su:
        for j in Su:
            if i<j:
                tr.addConstr(2 * xi[k,i,j] <= lambdaa[k,i] + lambdaa[k,j],
name="Restricción (44.b)[k,i,j] [%s,%s,%s]" %(k,i,j))
# ******************** Mismas en sentido DW *********************************

    "Restricción (54.a)"
for k in kd:
    for i in Sd:
        for j in Sd:
            if i<j:
                tr.addConstr(lambdaa[k,i] + lambdaa[k,j] <= 1 + xi[k,i,j],
name="Restricción(54.a)[k,i,j][%s,%s,%s]" %(k,i,j))

"Restricción (54.b)"
for k in kd:
    for i in Sd:
        for j in Sd:
            if i<j:
                tr.addConstr(2 * xi[k,i,j] <= lambdaa[k,i] + lambdaa[k,j],
name="Restricción (54.b)[k,i,j] [%s,%s,%s]" %(k,i,j))

#*********************************************************************
*
```

```
"Restricción (44.c)"
for k in ku:
    for i in Su:
        for j in Su:
            if i<j:
                tr.addConstr(waiting_pot_to_j[k,i,j] ==  waiting_to_j[k,i,j]
* xi[k,i,j], name="Restricción(44.c)[k,i,j][%s,%s,%s]" %(k,i,j))


"Restricción (54.c)"
for k in kd:
    for i in Sd:
        for j in Sd:
            if i<j:
                tr.addConstr(waiting_pot_to_j[k,i,j] == waiting_to_j[k,i,j] *
xi[k,i,j], name="Restricción(54.c)[k,i,j][%s,%s,%s]" %(k,i,j))

#*************************************************************************
*****

"Restricción (45)"
for k in ku:
    for o,i in enumerate(Su):
        if o>0:
            tr.addConstr(onboard[k,i] == onboard[k,i-1] - alighting[k,i] +
boarding[k,i], name="Restricción(45)[k,i][%s,%s]" %(k,i))
        else:
            tr.addConstr(onboard[k,i] == 0,
name="Restricción(45b)[k,i][%s,%s]" %(k,i))

"Restricción (46)"
x={}
for k in ku:
    for o,i in enumerate(Su):
        if o>0:
            x[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY, vtype=GRB.INTEGER,
name="x[%s,%s]" %(k,i))
for k in ku:
    for o,i in enumerate(Su):
        if o>0:
            tr.addConstr(x[k,i] == CAPACITY[1][1] - onboard[k,i-1] +
alighting[k,i], name="Subrestricción(46)[k,i][%s,%s]" %(k,i))
            #tr.addConstr(boarding[k,i] == min_(x[k,i],waiting_pot[k,i]),
name="Restricción(46)[k,i][%s,%s]" %(k,i))
            tr.addConstr(boarding[k, i] <= x[k, i],
name="Restricción(46A)[k,i][%s,%s]" % (k, i))
            tr.addConstr(boarding[k, i] <= waiting_pot[k, i],
name="Restricción(46B)[k,i][%s,%s]" % (k, i))

"Restricción (55)"
for k in kd:
    for o,i in enumerate(Sd):
        if o!=0:
            tr.addConstr(onboard[k,i] == onboard[k,i-1] - alighting[k,i] +
boarding[k,i], name="Restricción(55)[k,i][%s,%s]" %(k,i))
        else:
            tr.addConstr(onboard[k, i] == 0,
name="Restricción(55b)[k,i][%s,%s]" % (k, i))

for k in kd:
    for o,i in enumerate(Sd):
        if o!=0:
```

```
            x[k,i] = tr.addVar(lb=0.0, ub=GRB.INFINITY, vtype=GRB.INTEGER,
name='x[%s,%s]' %(k,i))
"Restricción (56)"
for k in kd:
    for o,i in enumerate(Sd):
        if o!=0:
            tr.addConstr(x[k,i] == CAPACITY[1][1] - onboard[k,i-1] +
alighting[k,i])
            #tr.addConstr(boarding[k,i] == min_(x[k,i],waiting_pot[k,i]),
name="Restricción(56)[k,i][%s,%s]" %(k,i))
            tr.addConstr(boarding[k, i] <= x[k, i],
name="Restricción(56A)[k,i][%s,%s]" % (k, i))
            tr.addConstr(boarding[k, i] <= waiting_pot[k, i],
name="Restricción(56B)[k,i][%s,%s]" % (k, i))


#*************************************************************************
**************

"Restricción (47)"
for k in ku:
    for i in Su:
        tr.addConstr(alighting[k,i] == quicksum(boarding_to_j[k,j,i] for j in
Su if j<i), name="Restricción(47)[k,i][%s,%s]" %(k,i))

"Restricción (48)"
for k in ku:
    for i in Su:
        for j in Su:
            if i<j:
                tr.addConstr(boarding_to_j[k,i,j] == (O2_u[i,j]/O1_u[i]) *
boarding[k,i], name="Restricción(48)[k,i,j][%s,%s,%s]" %(k,i,j))


"Restricción (57)"
for k in kd:
    for i in Sd:
        tr.addConstr(alighting[k,i] == quicksum(boarding_to_j[k,j,i] for j in
Sd if j<i), name="Restricción(57)[k,i][%s,%s]" %(k,i))

"Restricción (58)"
for k in kd:
    for i in Sd:
        for j in Sd:
            if i<j:
                tr.addConstr(boarding_to_j[k,i,j] == (O2_d[i,j]/O1_d[i]) *
boarding[k,i], name="Restricción(58)[k,i,j][%s,%s,%s]" %(k,i,j))




#*************************************************************************
**************
"Restricción (49)"
for k in ku:
    for i in Su:
        for j in Su:
            if i<j:
                tr.addConstr(Residuo[k,i,j] == waiting_to_j[k,i,j] -
boarding_to_j[k,i,j], name="Restricción(49)[k,i,j][%s,%s,%s]" %(k,i,j))
```

```
"Restricción (59)"
for k in kd:
    for i in Sd:
        for j in Sd:
            if i<j:
                tr.addConstr(Residuo[k,i,j] == waiting_to_j[k,i,j] -
boarding_to_j[k,i,j], name="Restricción(59)[k,i,j][%s,%s,%s]" %(k,i,j))

#*************************************************************************
************************

"Restricción (43b)"
for o,k in enumerate(ku):
    for i in Su:
        for j in Su:
            if i < j:
                if o == 0:
                    tr.addConstr(
                        waiting_to_j[k, i, j] == quicksum((1 - y[k, i, t]) *
Scala* OD[viajes_u[i][j]][t] for t in tiempo[1]),
                        name="Restricción(43.b)[k,i,j][%s,%s,%s]" % (k, i,
j))
                else:
                    tr.addConstr( waiting_to_j[k, i, j] == Residuo[k - 1, i,
j] +
                                    quicksum( (y[k - 1, i, t] - y[k, i, t]) *
Scala * OD[viajes_u[i][j]][t]  for t in tiempo[1] ),
                        name="Restricción(43.b)[k,i,j][%s,%s,%s]" % (k, i,
j))

"Restricción (53b)"
for o, k in enumerate(kd):
    for i in Sd:
        for j in Sd:
            if i < j:
                if o == 0:
                    tr.addConstr(
                        waiting_to_j[k, i, j] == quicksum( (1 - y[k, i, t]) *
Scala * OD[viajes_d[i][j]][t] for t in tiempo[1]),
                        name="Restricción(53.b)[k,i,j][%s,%s,%s]" % (k, i,
j))
                else:
                    tr.addConstr(waiting_to_j[k, i, j] == Residuo[k - 1, i,
j] +
                                    quicksum( (y[k - 1, i, t] - y[k, i, t]) *
Scala * OD[viajes_d[i][j]][t] for t in tiempo[1]),
                                    name="Restricción(53.b)[k,i,j][%s,%s,%s]" %
(k, i, j))

#*************************************************************************
*********************************************
#*************************************************************************
*********************************************
#*************************************************************************
*********************************************
Alpha=10.0 # Peso para los residuos al final del ultimo servicio
Beta=40.0 # Pso para las llegadas despues del ultimo servicio
Delta=5.0 # Peso para los que no quieren subir pudiendo hacerlo
Omega=5.0 # peso para viajes a y desde depots
Delta_u=[len(ku)-o for o,i in enumerate(ku)]
Delta_d=[len(kd)-o for o,i in enumerate(kd)]
Suma_up=[(k,i,j) for k in ku for i in Su for j in Su if (i<j and k > 1)]
```

```
Suma_dw=[(k,i,j) for k in kd for i in Sd for j in Sd if (i<j and k >
len(ku)+1)]
Suma_up_comp=[(k,i,j) for k in ku for i in Su for j in Su if j > i ]
Suma_dw_comp=[(k,i,j) for k in kd for i in Sd for j in Sd if j > i ]
# De momento falta lo que pasa antes del 1 y despues del último
pesou={i:(len(Su)-(i-1))**3 for i in Su}
pesod={i:(len(Sd)-o)**3 for o,i in enumerate(Sd)}


tr.setObjective(quicksum(pesou[i] * waiting_to_j[k,i,j] * (d[k,i]-d[k-1,i])/2
for (k,i,j) in Suma_up) +
                quicksum(pesod[i] * waiting_to_j[k,i,j] * (d[k,i]-d[k-1,i])/2
for (k,i,j) in Suma_dw ) + \
                Alpha * (quicksum( pesou[i] * Residuo[len(ku),i,j] *
(len(tiempo[1])-d[len(ku), i])/2 for i in Su for j in Su if j > i) +
                quicksum(pesod[i] *
Residuo[len(ku)+len(kd),i,j]*(len(tiempo[1])-d[len(ku)+len(kd), i])/2 for i
in Sd for j in Sd if j > i)) + \
                Beta * (quicksum(Scala * pesou[i] *OD[viajes_u[i][j]][t] *
y[len(ku), i, t] / 2 for i in Su for j in Su if j > i for t in tiempo[1]) +
                quicksum(Scala * pesod[i] * OD[viajes_d[i][j]][t] *
y[len(ku)+len(kd), i, t] / 2 for i in Sd for j in Sd if j > i for t in
tiempo[1])) + \
                Omega * ( quicksum(to_depot2[k] for k in ku) +
quicksum(to_depot1[k] for k in kd)  +
                quicksum(from_depot1[k] for k in ku) +
quicksum(from_depot2[k] for k in kd))-\
                1000* (quicksum( pesou[i] * boarding_to_j[k,i,j] for (k, i,
j) in Suma_up_comp ) + quicksum( pesod[i] * boarding_to_j[k,i,j] for (k, i,
j) in Suma_dw_comp))+ \
                -1000000*M[1][1]*quicksum( quicksum((boarding[k,i])  for i in
Sd) for k in kd)+\
                1000*M[1][1]*quicksum( quicksum( quicksum( (1-gamma[k,l,m])
for m in Smu if m>2 or m<5) for l in kd) for k in ku)+\
                1000*M[1][1]*quicksum( quicksum( quicksum( (1-gamma[k,l,m])
for m in Smd if m >8 or m < 10) for l in ku) for k in kd))


tr.write('modelo.lp')
tr.modelSense=GRB.MINIMIZE
setParam("NonConvex",2)
setParam('TimeLimit',300)
tr.optimize()
#tr.computeIIS()
#tr.write("model.ilp")
# para ver cómo actúa el modelo, he puesto cómo actúa alightingda servicio en
alightingda estación
print("_____Sentido UP_____")
for k in ku:
    print(" \nServicio %s tau[k]: %s" % (k,tau[k].x))
    for i in Su:
        print("Estación: %s lambda: %s T.llegada: %s T.salida: %s DwellTime
%s Esperando %s Pueden subir %s Suben %s Bajan %s Viajan %s" %(i,
lambdaa[k,i].x,a[k,i].x,d[k,i].x, dwell[k,i].x,
waiting[k,i].x,waiting_pot[k,i].x,boarding[k,i].x,alighting[k,i].x,onboard[k,
i].x))
print("\n")
print("_____Sentido DW_____")
# Aquí comprobamos los resultados en sentido downstream
for k in kd:
    print(" \nservicio %s tau[k]: %s" % (k,tau[k].x))
    for i in Sd:
```

```
        print("Estación: %s lambda: %s T.llegada: %s T.salida: %s DwellTime
%s Esperando %s Pueden subir %s Suben %s Bajan %s Viajan %s" %(i,
lambdaa[k,i].x,a[k,i].x,d[k,i].x, dwell[k,i].x,
waiting[k,i].x,waiting_pot[k,i].x,boarding[k,i].x,alighting[k,i].x,onboard[k,
i].x))

table={}
tablezone={}
for k in ku:
    if tau[k].x > 0.9:
        for i in Su:
            aux=[]
            aux.append(a[k,i].x)
            aux.append(d[k,i].x)
            table[k,i]=aux
for k in kd:
    if tau[k].x > 0.9:
        for i in Sd:
            aux=[]
            aux.append(a[k,i].x)
            aux.append(d[k,i].x)
            table[k,i]=aux
for k in ku:
    if tau[k].x > 0.9:
        for i in Smu:
            for j in Smu:
                if j > i:
                    if zone[k,i,j].x > 0.9:
                        aux=[]
                        aux.append(i)
                        aux.append(j)
                        tablezone[k]=aux
for k in kd:
    if tau[k].x > 0.9:
        for i in Smd:
            for j in Smd:
                if j > i:
                    if zone[k,i,j].x > 0.9:
                        aux=[]
                        aux.append(i)
                        aux.append(j)
                        tablezone[k]=aux

print("*************** TIMETABLE *****************")
for (k,i) in table:
    print("Servicio %s Estacion: %s %s " %( k,i, table[(k,i)][0]))
    print("Servicio %s Estacion: %s %s " % (k,i, table[(k, i)][1]))

print("*************** ZONAS ***************")
for k in tablezone:
    print("Servicio: %s  Origen: %s  Destino: %s"
%(k,tablezone[k][0],tablezone[k][1]))

print("*************** GAMMAS UP ***************")
for k in ku:
    for l in kd:
        for m in Su:
            if gamma[k,l,m].x > 0.9:
                print("Servicio Up: %s Servicio Down: %s Estación Up: %s
Gamma: %s" %(k,l,m,gamma[k,l,m].x))

print("*************** GAMMAS DOWN ***************")
```

```
for k in kd:
    for l in ku:
        for m in Sd:
            if gamma[k,l,m].x > 0.9:
                print("Servicio Down: %s Servicio Up: %s Estación Up: %s
Gamma: %s" %(k,l,m, gamma[k,l,m].x))
```