Proyecto Fin de Grado
Grado en Ingeniería de Organización Industrial

Asignación de pasajeros y determinación de frecuencias en redes ferroviarias de tránsito rápido

Autor: Marta Elvira Madrid Naz

Tutor: José David Canca Ortiz

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

Sevilla, 2018

Proyecto Fin de Grado
Ingeniería de Organización Industrial

Asignación de pasajeros y determinación de frecuencias en redes ferroviarias de tránsito rápido

Autor:

Marta Elvira Madrid Naz

Tutor:

José David Canca Ortiz

Profesor titular

Dep. Organización Industrial y Gestión de Empresas I

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2018

Proyecto Fin de Grado: Asignación de pasajeros y determinación de frecuencias en redes ferroviarias de tránsito rápido

Autor: Marta Elvira Madrid Naz

Tutor: José David Canca Ortiz

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

Agradecimientos

Quisiera agradecer a todas aquellas personas que, ya sea directa o indirectamente, han hecho posible la realización de este trabajo.

A mi tutor, José David Canca, por su gran dedicación, paciencia y disponibilidad a lo largo de todos estos meses, y por supuesto, por haberme brindado la oportunidad de realizar este trabajo junto a él, ha sido un orgullo para mí.

A mis padres, por darme la oportunidad de estudiar en Sevilla. Gracias por confiar siempre en mí, por vuestro cariño y apoyo en todo momento, y por animarme en los momentos más difíciles. Sin vosotros, nada de esto hubiese sido posible.

A mis amigas de clase y compañeras de piso, por hacerme vivir tan buenos momentos en estos cuatro años. Esta ciudad ha sido mucho más bonita gracias a vosotras.

Marta Elvira Madrid Naz

Sevilla, 2018

Resumen

Durante las últimas décadas, la necesidad de movilidad de las personas ha sufrido un increíble aumento, lo que ha provocado una gran demanda diaria de los medios de transporte público, y como consecuencia, del transporte ferroviario. Además, tal demanda sigue aumentando día a día en número y en exigencia, por lo que resulta fundamental lograr una planificación adecuada del servicio. El presente proyecto, se centra en una de las etapas necesarias para su consecución, concretamente, tiene el propósito de desarrollar e implementar un modelo de programación lineal mixta-entera que permita determinar las frecuencias óptimas junto con el resto de variables necesarias para programar cada una de las líneas que componen una red de cercanías, atendiendo a las características y exigencias que supone un sistema de transporte como el que se presenta. Adicionalmente, debido a que el modelo se propone para una red compuesta por diferentes líneas que cuentan con varias estaciones en común, los pasajeros tendrán en la mayoría de las ocasiones diferentes opciones o rutas para llegar a un mismo destino, y por ello, surge la necesidad de realizar una asignación de tránsito que nos permita ver cómo se comportan los usuarios dada la red de transporte desarrollada, es decir, que estrategias de viaje seleccionan para llegar al destino deseado. Los dos problemas presentados guardan una estrecha relación, ya que, en base a las frecuencias establecidas, los pasajeros decidirán escoger una ruta u otra, pero al mismo tiempo, en función del número de viajeros que utilicen una línea, se establecerá una determinada frecuencia de paso. Por lo tanto, se busca alcanzar un equilibrio entre ambos problemas.

Así mismo, con el objetivo de satisfacer al mismo tiempo tanto a operadores como a pasajeros, se persigue en todo momento la minimización de los gastos de tripulación y de los costes de operación de los diferentes modelos de tren puestos en marcha, y, junto con ello, minimizar también el valor monetario del tiempo total de viaje empleado por los pasajeros para llegar a sus correspondientes destinos, incluyendo no sólo el tiempo a bordo del tren, sino también tiempos de transferencia y esperas. No obstante, debido a que ambos objetivos son contrapuestos, antes de escoger una solución final, se muestreará la Frontera de Pareto mediante la resolución de distintos escenarios, a lo largo de los cuales, se irá variando la ponderación de cada objetivo.

Finalmente, con el propósito de evaluar el desempeño del modelo de optimización propuesto, se ha realizado su aplicación práctica en una parte de la red de cercanías de Valencia. Dicha implementación, ha sido llevada a cabo mediante el lenguaje de programación Python y, usando a su vez como software de optimización, el solver Gurobi.

Abstract

Over the last decades, the people´s need for mobility has suffered an incredible increase, which has caused a great daily demand for public transport, and consequently, for rail transport. In addition, such demand continues to increase day by day in number and in request, so it is essential to achieve an adequate planning of the service. The present project focuses on one of the necessary stages for its achievement, specifically, it has the purpose of developing a linear programming model that allows determining the optimal frequencies along with the rest of necessary variables to schedule each of the lines that make up a commuter rail network, considering the characteristics and requests of a transport system such as the one presented. Additionally, due to the model is proposed for a network composed of different lines that have in common several stations, in most cases, passengers will have different options or routes to reach the same destination and, because of that, the need arises to make a transit assignment that allows us to see how the users behave given the transport network developed, that is, which travel strategies they select to reach the desired destination. The two problems presented are closely related, since, based on the established frequencies, passengers will choose one route or another, but at the same time, depending on the number of passengers using a line, a certain frequency will be established. Therefore, it is sought to find a balance between both problems.

Likewise, with the objective of satisfying both operators and passengers at the same time, it is pursued the minimization of crew costs and operation costs of the different train models put in place, and, along with this, also minimize the monetary value of the total trip time used by passengers to reach their corresponding destinations, including not only on-board travel times, but also transfer times and waiting times. However, since both objectives are opposed, before choosing a final solution, different points of the Pareto Border will be obtained through the resolution of different scenarios in which the weighting of each objective will vary.

Finally, with the purpose of evaluating the performance of the proposed linear programming model, it has been carried out its practical application in a part of the Valencia commuter network. This implementation has been carried out using the Python programming language and, using as optimization software, the Gurobi solver.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

En las últimas décadas se ha ido produciendo una gran expansión de las ciudades, lo que ha provocado un aumento en la necesidad de movilidad de las personas. Como consecuencia, los medios de transporte público se han convertido en un servicio básico en la mayoría de las ciudades (especialmente en aquellas más grandes), siendo utilizados diariamente por millones de personas para ir tanto a núcleos urbanos como a las zonas más periféricas, de tal forma que hoy en día nos resultaría muy complicado concebir una sociedad exenta de dichos medios.

Son múltiples las ventajas que estos nos ofrecen, desde la enorme reducción en la emisión de gases contaminantes a la atmósfera, hasta el gran ahorro económico que este supone en comparación con otros medios de transporte terrestres.

Concretamente, en España, uno de los medios más característicos de los que disponemos es el sistema ferroviario, el cual, en el caso de la alta velocidad, nos ha hecho posicionarnos como el país de Europa con mejor red ferroviaria. Merece la pena destacar también la alta utilización de las redes de Metro y cercanías en las grandes ciudades. Esto es debido, entre otros aspectos, a la gran operatividad y eficiencia tecnológica que se ha conseguido alcanzar a lo largo de los últimos años. Además de las ventajas mencionadas anteriormente, cabe destacar la alta velocidad que este medio es capaz de alcanzar, junto con la gran seguridad que nos proporciona. Por otro lado, hay que tener en cuenta que este tiene una demanda que aumenta día a día, no sólo en número sino también en exigencia, por lo que, para satisfacerla, el sistema de transporte debe estar en constante desarrollo y renovación. Aquí, la labor de investigadores y planificadores ferroviarios adquiere una gran importancia.

Antes de comenzar con la definición del problema y la metodología propuesta para su resolución, se hace necesario realizar una pequeña introducción, de tal forma que se aborden ciertos conceptos que resultan de interés para un mejor entendimiento de este trabajo.

1.1 Introducción a la planificación ferroviaria

La planificación de un sistema de transporte público abarca un área de investigación muy extensa en la sociedad actual. El primer estudio (o al menos que tengamos constancia de ello) relacionado con la planificación de redes de transporte, se le atribuye a (Patz 1925).

Concretamente, este propone un procedimiento iterativo para abordar el problema de diseño de redes, el cual será explicado más adelante. Desde entonces, numerosos estudios han sido realizados sobre el tema, los cuales, han ido evolucionando desde diferentes perspectivas: tanto la de pasajeros como la de operadores del sistema; y, buscando a su vez, lograr distintos objetivos tales como satisfacer la demanda, minimizar los costes operacionales o los tiempos de espera en las estaciones, convirtiéndose así el diseño y la planificación del transporte público, en un tema crucial hoy en día.

Centrándonos ahora en la planificación del sector ferroviario, se podría decir que esta no posee un único objetivo, sino que pretende atender a un conjunto de propósitos que resultan fundamentales para lograr el correcto funcionamiento del sistema y, por lo tanto, para ofrecer un servicio adecuado a sus usuarios. Entre los más destacables se encuentran la calidad, el coste operacional y el impacto al medio ambiente.

Podemos entender por un servicio de transporte de calidad, aquel en el cual los trenes y todo el sistema inherente a ellos funcionan correctamente, aunque en realidad el término calidad va más allá, englobando diversos ámbitos. Por un lado, consiste en que el servicio garantice el mayor grado de seguridad posible a sus usuarios, contando con planes emergencia y actuación en caso de imprevistos. Por otro lado, que este tenga suficiente capacidad para satisfacer toda la demanda de pasajeros, ofreciéndoles a su vez el mayor grado de comodidad posible y teniendo en cuenta también, la incorporación de los elementos necesarios que hagan que toda la sociedad, sea cual sea su condición (por ejemplo, personas con movilidad reducida), tenga posibilidad de acceder al sistema. Al mismo tiempo, la puntualidad y disponibilidad del servicio también son factores que afectan enormemente a la calidad del servicio.

Cuando hablamos de coste operacional nos referimos, entre otros costes, al consumo de electricidad, combustible y mantenimiento. Una eficiente planificación minimiza dichos costes, haciendo el sistema económicamente viable. Hay que tener en cuenta que debe existir un equilibrio entre los gastos asociados al sistema ferroviario y la calidad del servicio ofertado, de tal manera que se genere un beneficio razonable.

Ni que decir tiene que la preservación del medio ambiente es un asunto que cada día concierne más a la sociedad, por ello, es inevitable abordarlo a la hora de planificar el sistema. Se trata de buscar en todo momento causar el menor impacto posible en el entorno que nos rodea, tanto a la hora de la construcción de infraestructuras como en el empleo del combustible.

Por otro lado, son múltiples los factores que provocan que la planificación de redes y servicios de transporte ferroviario sea una actividad de enorme complejidad. Uno de los más destacables se encuentra en que, a diferencia del resto de transportes públicos, este sistema requiere desarrollar una infraestructura específica que permita el desplazamiento de los

diferentes equipos, lo que condiciona de manera importante la gestión y despliegue del servicio en cuestión. Además, tal y como se argumenta en (Orro 2006), no debe olvidarse que la mayoría de la inversión realizada para dichas infraestructuras no podrá ser recuperada para otros usos, ya que el único fin de una vía férrea es atender los servicios que en ella se prestan. Por lo tanto, será crucial una adecuada planificación que esté en constante desarrollo y evolución.

1.2 Proceso de planificación

La planificación de un completo sistema de transporte ferroviario conlleva una serie de problemas de alta complejidad. Tradicionalmente, para facilitar su estudio, estos han sido organizados de manera jerárquica en función del horizonte temporal abarcado siguiendo el esquema habitual de niveles de planificación (estratégica, táctica, operacional), asegurando así, un correcto funcionamiento de todo el sistema tanto en el día a día como también a medio y largo plazo. A pesar de la división establecida, dichos problemas guardan una estrecha relación entre sí, de tal forma que las decisiones tomadas para resolver cada uno de ellos, condicionarán a su vez, al resto de problemas, por lo que habitualmente son resueltos secuencialmente.

Siguiendo este enfoque, se distinguen fundamentalmente tres niveles de planificación:

- La planificación estratégica se realiza con una visión a largo plazo, tratando con horizontes temporales de varios años o incluso décadas. Su finalidad es responder con solvencia a las situaciones de demanda que se presentarán en un futuro, fundamentalmente en términos de conexiones directas, frecuencias y fiabilidad (Caprara et al. 2007).

- La planificación táctica está orientada a un futuro más cercano, manejando generalmente, horizontes temporales de varios meses. En especial, esta planificación busca ofrecer una buena calidad en el servicio a sus usuarios.

- La planificación operacional se encarga de un horizonte temporal mucho más reducido, normalmente días o como mucho un mes. Principalmente, buscar ajustar los planes tácticos a las características especiales que puedan presentarse a corto plazo.

A continuación, se introduce un esquema a modo de resumen que muestra el total de los problemas que conlleva la planificación ferroviaria, clasificados en función de los tres niveles mencionados anteriormente.

Posteriormente, se procederá a realizar una descripción de los mismos, junto con algunos más que también resultan de interés.

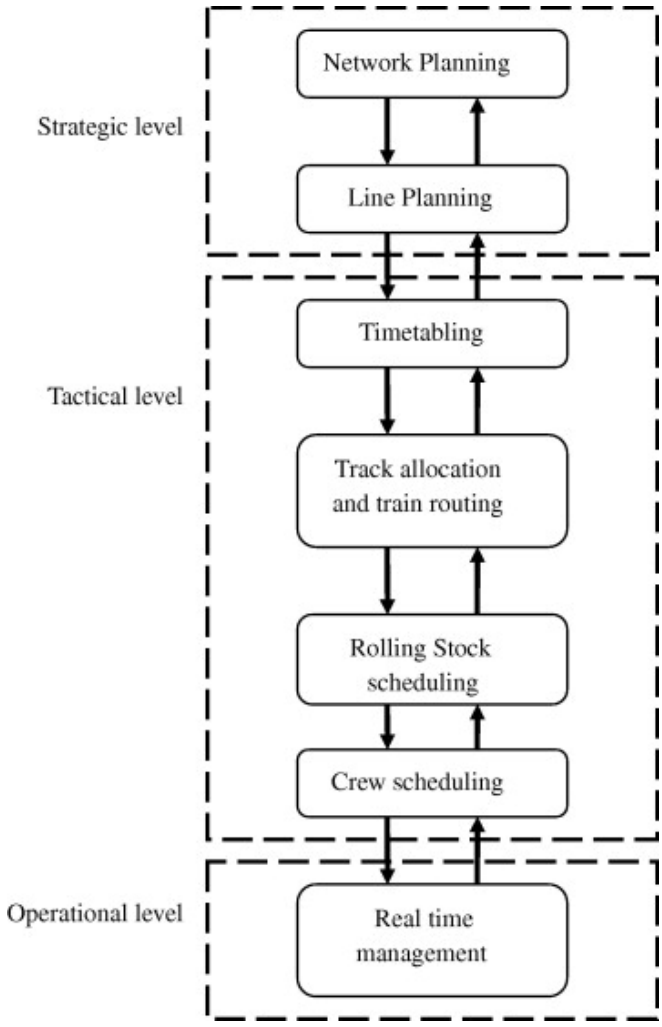


Figura 1. Esquema general de problemas en la planificación ferroviaria (Heydar et al. 2013)

1.2.1 Problemas estratégicos

Diseño de redes (Network Design)

El primer paso a la hora de realizar la planificación del sistema ferroviario consiste en diseñar la red en la cual operará el sistema.

La mayoría de las veces, este problema se aborda mediante la formulación de un grafo con sus correspondientes nodos y aristas, tal que G = (N, A), donde los nodos representan las distintas estaciones de la red, y las aristas se corresponden con las vías por las que circularán los trenes.

Para el diseño de la red, será necesario conocer previamente las necesidades de infraestructuras existentes, es decir, conocer la demanda futura del servicio. Habitualmente para el tratamiento de este aspecto, se tiene como dato de entrada una o varias matrices Origen-Destino, que representan la demanda de pasajeros entre cada par de nodos (estaciones) de la red, ya sea diariamente o para un período concreto del día.

El estudio y análisis de este tipo de problemas debe tener en cuenta numerosos aspectos. No solo los meramente económicos, debido al hecho de que se están construyendo nuevas infraestructuras y esto lleva un alto coste asociado, sino también aspectos de tipo técnico, sociales y políticos propios de la zona, que influirán en gran medida en la selección de la solución más adecuada. De hecho, este último aspecto tendrá especial relevancia, debido a que generalmente, la toma de decisiones en esta etapa es llevada a cabo por los organismos públicos del gobierno competentes en transporte, en España, el Ministerio de Fomento.

Son varios los estudios que han abordado el problema de diseño de redes ferroviarias. La ubicación de nuevas líneas ha sido tratada en los trabajos de (Bruno et al. 1998) donde en un primer artículo se busca minimizar la suma de los costes asociados a la construcción junto con el tiempo de viaje de los pasajeros, y posteriormente, en un segundo artículo, (Bruno 2002) buscará maximizar la cobertura de la población susceptible de utilizar la nueva línea. A su vez, (Dufourd et al. 1996) también tratarán de maximizar la cobertura de la población aplicando la Búsqueda Tabú (Glover 1989; Glover 1990).

Sin embargo, actualmente resulta bastante complicado encontrar posibilidades para diseñar una red partiendo de cero. Es por ello por lo que, habitualmente, los problemas de diseño de redes suelen buscar la expansión de una red ya existente, planteando por ejemplo la inclusión de nuevas estaciones de parada o la creación de nuevas líneas, respetando la topología de red ya existente. Tal situación se puede encontrar en (Xiong & Schneider 1992), quienes presentan un método basado en la implementación de algoritmos genéticos para añadir nuevas rutas a una red ya existente. (López-Ramos 2017) también tratan la expansión de una red, buscando en este caso, minimizar los costes de las nuevas infraestructuras.

Planificación de líneas (Line Planning)

De igual forma que la etapa anterior, la planificación de las líneas se encuentra dentro de la fase de planificación estratégica.

Para abordar este apartado, será necesario definir previamente el concepto de "línea de tren", el cual, es definido por (Caprara et al. 2007) como un conjunto de trenes dentro de la red establecida que se caracteriza por realizar siempre el mismo recorrido, parte del mismo inicio y llega al mismo destino, realizando a su vez las paradas en las mismas estaciones, de tal modo que la única diferencia existente entre los trenes pertenecientes a una misma línea, es la hora de llegada y salida a cada estación. Teniendo en cuenta todo ello, el propósito principal de esta etapa consiste en encontrar tanto el origen como el destino final de cada línea, y el camino sobre la red que debe seguir cada una de ellas, de forma que se satisfagan ciertos objetivos. Los principales, que entran en conflicto en la mayoría de ocasiones, consisten en maximizar el servicio proporcionado a los pasajeros, al mismo tiempo que se minimizan los costes operacionales del sistema (Caprara et al. 2007).

Una posible solución a este problema, bastante tratada en la literatura científica, consiste en plantear un modelo de optimización con el propósito de desplazar al mayor número posible de pasajeros desde sus puntos de origen hacia sus destinos habituales (reflejado en la Matriz Origen-Destino ya mencionada con anterioridad) mediante trayectos directos, o en el caso de que no sea posible, minimizando el número de trasbordos realizados. Su consecución implicaría, por lo general, la construcción de líneas de gran longitud, sin embargo, esto daría lugar a grandes retrasos propagados a lo largo de toda la línea. Para que esto no ocurra y el sistema sea robusto, las líneas deben ser relativamente cortas, lo que implica que los pasajeros tengan que transbordar con bastante frecuencia, por lo que se trata de conseguir un equilibrio entre ambos criterios (Caprara et al. 2007).

Por lo general, el establecimiento de las diferentes líneas que conformarán la red ha sido planteado en la literatura científica como un problema de selección de líneas, a escoger de entre un conjunto definido previamente, por lo que resulta fundamental definir adecuadamente el conjunto potencial de líneas candidatas.

Durante dicha fase de planificación de líneas, se suele abordar simultáneamente el problema del establecimiento de frecuencias para cada línea de la red, entendiéndose por frecuencia, la cantidad de trenes que pasan por cada estación de la red en un intervalo de tiempo dado (normalmente la frecuencia viene referida a intervalos de una hora). La demanda máxima en cada tramo de la línea es la que va a determinar la frecuencia de servicio de la misma, lo que da lugar generalmente a frecuencias bastante altas, especialmente cuando se consideran matrices de demanda correspondientes a períodos de tiempo largos.

Por ello, frecuentemente lo que se hace es utilizar la matriz de demanda diaria para definir el conjunto de líneas de la red y, posteriormente, para determinar la frecuencia de cada una de ellas, se consideran matrices específicas correspondientes a horas punta del día y a aquellas horas con menos demanda.

Respecto a los estudios realizados sobre el tema, cabe decir que no es muy elevado el número de artículos que tratan el problema de planificación de líneas. (Dienst 1978) utiliza la técnica de Branch and Bound para diseñar un sistema de líneas que busca maximizar el número de pasajeros que realizan trayectos sin necesidad de transbordar hacia otras líneas del sistema definido. Más tarde, (Bussieck et al. 1997) también buscarán el mismo objetivo. Para ello, en este caso utilizan variables de decisión que representan la frecuencia de cada línea y, además, asumen que todos los trenes tienen la misma capacidad, la cual se considera conocida a priori. Posteriormente, (Bussieck 1998) amplía el modelo propuesto. (Lindner & Zimmermann 2000) al igual que (Goossens & Hoesel 2004), se centran en diseñar un sistema de líneas buscando el mínimo coste posible, además, ambos proponen un modelo de optimización que resuelven mediante Branch and Bound. (Scholl 2005) tratará también el problema, considerando en su caso, el objetivo de minimizar el número de transbordos realizados entre diferentes líneas.

1.2.2 Problemas tácticos

Programación de trenes (Train Scheduling)

El siguiente nivel en la cadena jerárquica de problemas se establece para un horizonte temporal menor que los anteriores, es el denominado Scheduling.

Antes de comenzar con este punto, resulta conveniente resaltar las diferencias entre los conceptos Scheduling y Timetabling, debido a que son usados indistintamente en múltiples ocasiones.

Por un lado, el término Scheduling incluye el establecimiento de horarios de salida y llegada de cada uno de los trenes a las diferentes estaciones de la red, pero, además, también trata de identificar la secuencia (orden de paso) que siguen los trenes por las estaciones a las que prestan servicio. Esto último cobra especial importancia cuando tratamos con estaciones que se encuentran conectadas mediante una única vía. Mientras que, por otro lado, el concepto de Timetabling hace referencia solamente a la determinación de los horarios. Debido a la relevancia que tiene este último aspecto en el presente trabajo, más adelante se dedicará un apartado para profundizar exclusivamente en la determinación de los horarios, por lo tanto, en este apartado centraremos la atención en los problemas de Scheduling.

Tal y como ocurre en otros campos de investigación (por ejemplo, la producción), los problemas de Scheduling pueden llegar a ser realmente complejos, resultando muy difícil y costoso en tiempo el llegar a una solución factible. Concretando en el ámbito que nos interesa, esta dificultad se debe entre otros aspectos, a que tales problemas incorporan decisiones adicionales como pueden ser la coordinación de trenes en tramos compartidos por varias líneas, las detenciones en apartaderos, los cruces, necesidad de incorporar largos tiempos de parada en alguna de las estaciones debido a la gran cantidad de pasajeros que suben y/o bajan en ella, o, por el contrario, tomar decisiones sobre el paso por alguna estación sin detenerse.

Los problemas de Scheduling han sido muy tratados en la literatura. (Szpigel 1973) utiliza variables continuas para representar los tiempos de llegada a las estaciones y mediante restricciones modelará el orden de salida de los diferentes trenes de cada estación de la red. Años más tarde, (Jovanovic & Harker 1991) resuelven una variante de estos modelos buscando soluciones que sean factibles, dejando a un lado la búsqueda de un objetivo concreto. Por otro lado, (Carey & Lockwood 1995) utilizan una heurística que va trabajando con los trenes de uno en uno, y para cada uno de ellos, resuelve un problema MILP (Programación Lineal Entera Mixta). Más recientemente, nos encontramos por ejemplo a (Pashchenko et al. 2015), quienes modelan el problema de Scheduling como un problema de asignación. A su vez, para su resolución usan tanto el método de subasta como el método húngaro, siendo los resultados obtenidos en ambos finalmente comparados.

Asignación de plataformas (Train Platforming)

Tras haber realizado la programación de los trenes, el siguiente paso consiste en asignar una plataforma o andén determinado a cada tren que llega a una estación. Ambos problemas están estrechamente relacionados resolviéndose de forma secuencial, de hecho, habitualmente se itera con las dos fases hasta que ambas soluciones son aceptadas por el Planificador Central. Cabe destacar que mientras el primero se resuelve para toda la red, el segundo se realiza para cada estación por separado.

Este segundo problema, más conocido por sus siglas en ingles TPP (Train Platforming Problem), se ocupa del establecimiento de la ruta que cada tren programado debe seguir al llegar, detenerse y por último abandonar una estación, lo cual contempla la asignación de una determinada plataforma (ver Figura 2). En particular, cada tren tiene que realizar un camino desde el punto donde ingresa a la estación hasta el punto donde la abandona.

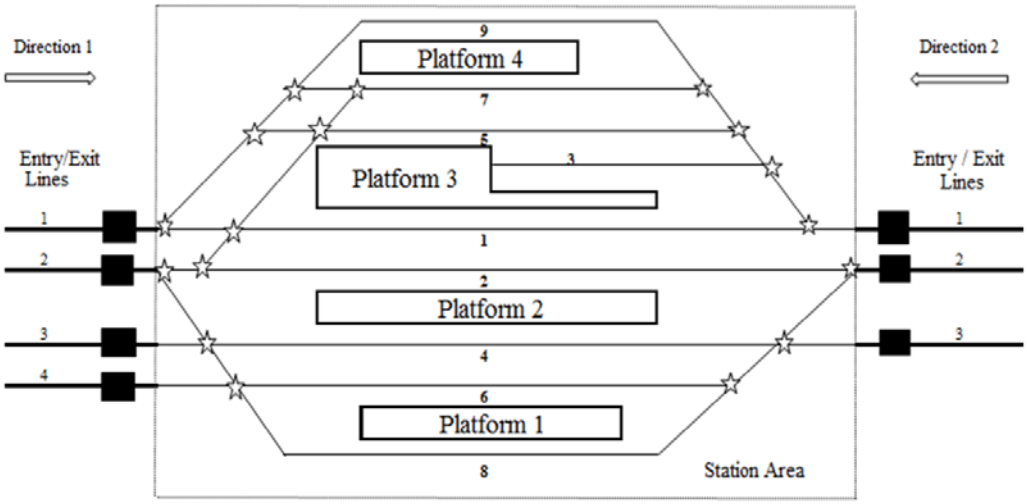


Figura 2. Ejemplo de la topología de una estación (Akyol 2017)

Dado un conjunto de instrucciones para llegadas y salidas de trenes, un conjunto de plataformas y un conjunto de trenes, con el horario correspondiente, direcciones y conjunto de plataformas donde el tren puede detenerse, el problema determina la ruta óptima de los trenes, evitando rutas incompatibles y buscando, en la mayoría de las ocasiones, minimizar el tiempo necesario para completar los movimientos entre llegadas y salidas de la estación (Caprara et al. 2007).

Hay dos tipos principales de restricciones que se establecen para evitar incompatibilidades (Cacchiani 2008):

- Imponer un intervalo de tiempo mínimo entre la salida de un tren de una plataforma y la llegada del próximo tren a la misma.

- Prohibir la superposición entre rutas, es decir, la ocupación de dos rutas incompatibles por dos trenes en el mismo instante de tiempo.

Dependiendo del contexto particular, pueden existir más restricciones relacionadas con el diseño de la estación, condiciones de seguridad o aspectos operacionales.

En relación con la literatura desarrollada sobre el tema, no son demasiados los autores que lo han abordado. (De Luca Cardillo & Mione 1998) desarrollan una versión simplificada, en la cual, los horarios de llegada y salida programados inicialmente para cada tren no pueden ser alterados, y las rutas se establecen únicamente mediante la elección de una determinada plataforma. Por otro lado, (Kroon et al. 1997) y (Zwaneveld et al. 1996) ofrecen una visión más general del problema, en la cual, los instantes de llegada y salida no

están fijados previamente. A pesar de que en la práctica a cada tren hay que asignarle una determinada ruta de llegada, salida y paso por una estación, los autores consideran la opción de que pueda no encontrarse alguna de ellas. Esta versión la modelan mediante un grafo, donde cada nodo representa una posible elección para un tren con un determinado coste asociado y las aristas unen los diferentes nodos. (Carey & Carville 2003) realizan una versión intermedia a las dos mencionadas, en la cual, se pueden modificar los horarios de llegada y salida, no obstante, la ruta que siguen los trenes dentro de una estación se ve definida por la asignación a una plataforma determinada. Estos autores describen un procedimiento heurístico que posteriormente es aplicado al caso real de una compañía británica.

Gestión del Material Rodante (Rolling Stock Circulation)

Antes de comenzar a abordar este apartado, resulta interesante definir qué se entiende por material rodante. Según (Berbey et al. 2013), se denomina de tal forma a todos los equipos y vehículos dotados de ruedas que se desplazan sobre las vías formando trenes o unidades de trenes, típicamente hablamos de:

- Locomotoras y vagones.
- Módulos agregados formados por locomotoras y vagones, llamados unidades.

Figura 3. Unidades de 3 y 4 módulos (Maróti 2006)

En dicho problema, más conocido por sus siglas en inglés RSCP, los operadores de servicios de transporte ferroviario tienen que decidir el tipo y el número de unidades de material rodante que se asignarán a los trenes ya programados con anterioridad, con el objetivo de satisfacer la demanda de los pasajeros. A su vez, también forma parte de esta fase el establecimiento de las políticas de mantenimiento de dicho material.

Se trata de una decisión muy compleja, la cual juega un papel fundamental dentro de la planificación, dado que su adquisición supone inversiones muy elevadas con un gran impacto en las futuras décadas.

Con el propósito de satisfacer la demanda de pasajeros (esta difiere entre trayectos y en función de la hora del día considerada), la composición de los trenes puede variar a lo largo de su recorrido en determinadas estaciones, surgiendo la necesidad de añadir o eliminar vagones de los diferentes trenes. Hay que tener en cuenta que dichos cambios llevan asociados unos tiempos de operación para efectuar los acoplamientos o desacoplamientos necesarios, los cuales deben respetar ciertas condiciones tales como el tiempo disponible entre paso de trenes consecutivos. Como consecuencia de ello, este tipo de operaciones suponen también ciertos costes que deben ser considerados en el proceso. Por otro lado, también obliga a disponer de cierto espacio destinado al almacenamiento de dichos vagones removidos.

Según (Cacchiani 2008), si nos fijamos en la naturaleza de la red de transporte, se puede establecer una división muy importante para el mantenimiento del material rodante. Por un lado, tenemos las redes de baja densidad, caracterizadas por recorridos de largas distancias, para las cuales es necesario tener en cuenta las operaciones de mantenimiento y construir un calendario para ellas. Por otro lado, cuando hablamos de redes de alta densidad, caracterizadas por cortos recorridos, dicho mantenimiento puede llegar a ser bastante más sencillo, debido a la posibilidad que existe de intercambiar unidades que son idénticas en los intervalos de tiempo que se producen durante las paradas realizadas.

En cuanto a la literatura desarrollada para este problema, se puede destacar el trabajo de (Schrijver 1993), quien propone un modelo para determinar el mínimo número de unidades de tren que son necesarias para satisfacer la demanda existente, en una línea perteneciente a una red de alta densidad. Dicho modelo permite que la composición del tren se pueda modificar en las distintas estaciones de parada. Posteriormente, (Kroon & Peeters 2003) describen un modelo que busca minimizar la falta de plazas y otros costes asociados, teniendo además en cuenta el número de cambios realizados en las composiciones de los trenes. (Lingaya et al. 2002) formulan un modelo que pretende maximizar el beneficio esperado, teniendo que cumplir a su vez varias restricciones, como son las establecidas para cumplir con los requisitos de mantenimiento y con el tiempo mínimo necesario para acoplar o desacoplar vagones al tren.

Desvío de las unidades de tren (Train Units Shunting)

Durante de las horas punta del día, normalmente todo el material rodante está operando en el calendario establecido o se encuentra en mantenimiento. Sin embargo, fuera de este horario (especialmente durante la noche), los operadores disponen de un exceso de material rodante. Tales unidades de tren excedentes pueden estacionarse en una zona aislada denominada apartadero, para así poder explotar la infraestructura principal.

La opción de estacionar una unidad de tren en el apartadero tiene varias implicaciones (Caprara et al. 2007). Primero, hay que tener en cuenta que, si las unidades estacionadas son de distinto tipo, es importante el orden en el que se haga, para que de esta forma ninguna unidad de tren bloquee la llegada o salida de otra. Además, el tiempo es también un recurso restrictivo para el desvío. Por ejemplo, por razones de seguridad, es obligatorio respetar un cierto tiempo mínimo de avance entre dos movimientos ferroviarios en la misma pista. Otros aspectos que juegan un papel importante y por ello deben ser estudiados son, entre otros, la ruta que siguen las unidades desde la vía principal hasta el aparcadero, las preferencias para ocupar estas zonas, la limpieza y mantenimiento de las unidades de tren y la disponibilidad de la tripulación para llevar a cabo las actividades necesarias para realizar la operación.

Por otro lado, por la noche, también se busca estacionar las unidades de tal manera que las operaciones en la mañana siguiente puedan iniciarse lo más fácil y rápido posible.

Este proceso depende altamente de los cambios que se produzcan en la programación de los trenes, ya que cuando esta se ve alterada, la planificación de los desvíos deberá modificarse también.

En la siguiente imagen se puede apreciar las dimensiones que llegan a alcanzar dichas áreas, lo que permite hacernos una idea de la complejidad que conlleva el problema de “Shunting”.

Figura 4. Vista satélite del apartadero Inman Yard en Atlanta

Con relación a la literatura escrita sobre el tema, (Tomii et al. 1999) proponen un algoritmo genético para resolver el problema teniendo en cuenta tanto la ruta que deben seguir los trenes hasta el aparcadero, como los turnos del personal y el mantenimiento de las

unidades estacionadas. Es un modelo bastante sencillo, ya que considera que como máximo sólo una unidad de tren pueda estar estacionada en cada apartadero. (Di Stefano & Love Koci 2004) estudian de qué forma estacionar los trenes para evitar movimientos a la mañana siguiente. Además, asumen que cada vía es lo suficientemente larga como para albergar todos los trenes que se le asignan. Una elaborada introducción, junto con un modelo y los resultados obtenidos tras aplicarse a una estación real, se pueden encontrar en (Freling et al. 2005). Más adelante, (Haijema & Dijk 2006) proponen una heurística basada en la programación dinámica, obteniendo unos resultados muy prometedores.

Programación de la tripulación (Crew scheduling)

Una vez programados los trenes y habiendo establecido un plan para la gestión del material rodante, la siguiente fase consiste en programar a la tripulación, es decir, establecer los diferentes turnos de trabajo (carga de trabajo habitualmente diaria) para el personal (tanto conductores como revisores), de forma que se cubran todos los desplazamientos programados para cada día y, al mismo tiempo, se busque minimizar la cantidad de turnos necesarios. Una vez resuelto este problema, se procederá a la asignación de los turnos a cada miembro individual de la tripulación, pero esta segunda parte será tratada en el siguiente apartado.

Tal y como se indica en (Caprara et al. 2007), lo primero que se debe hacer es dividir cada servicio en una secuencia de viajes, definidos como un conjunto de segmentos de viajes en tren que debe ser atendido por la misma tripulación sin descanso. Cada viaje se caracteriza por una hora de salida, una estación de salida, un horario de llegada y una estación de llegada, y este, debe ser realizado por una misma tripulación.

La determinación de los turnos va sujeta a una serie de requisitos, como por ejemplo serían el que cada tren deba tener un conductor y un número dado de revisores. También habrá que tener en cuenta el tipo de servicio, si se trata de una línea de metro, un tren de larga distancia, etc. Otras restricciones son aquellas derivadas de los convenios laborales específicos firmados entre empresa y trabajadores, tales como que cada turno no sea superior a un número determinado de horas y que incluya una parada mínima para el tiempo de comida. Además, cabe destacar que la programación táctica de la tripulación toma como entrada la planificación del material rodante, debido entre otras razones a que la cantidad de personal necesario depende de la longitud del tren.

El resultado obtenido tras esta etapa, y que se utilizará como entrada a la siguiente, será un conjunto de turnos diarios (aún no asignados a una persona física) dependientes del tipo de día, los cuales recogen tanto la hora de inicio y fin, como demás información

necesaria sobre el lugar de comienzo y finalización, la línea asignada, rotaciones a realizar durante el turno.

Con relación a la literatura destinada a tratar este problema, son números los estudios realizados, generalmente resueltos mediante la generación de un subconjunto adecuado de todos los turnos de trabajo y la posterior selección de ellos mediante la resolución de un problema de cobertura de conjuntos, ver (Balas & Carrera 1996) para más información. (Caprara et al. 1997) diseñaron un método heurístico basado en relajación Lagrangiana aplicado al ferrocarril italiano que resultó bastante eficaz. (Abbink et al. 2005) tratan un problema de Crew Scheduling aplicado al sector ferroviario de los Países Bajos, donde el objetivo fundamental consiste en asignar toda la carga de trabajo de la forma más equilibrada posible entre los puntos de origen y destino de los diferentes empleados.

Asignación de turnos a la tripulación (Crew Rostering)

Como se mencionó anteriormente, tras haber obtenido un calendario de turnos que ofrezca la cobertura necesaria para cumplir con todos los servicios programados, se procederá con la asignación individual del personal disponible a los diferentes turnos.

Se trata de un proceso muy complejo debido a que hay que tener en cuenta diversos factores como pueden ser las características propias de cada trabajador (categoría/nivel, años de antigüedad, preferencias individuales, etc.) y condiciones de seguridad necesarias. Otro aspecto a tener en cuenta es el establecimiento de días de descanso (habitualmente 2 o 3, en función del convenio) tras un determinado número de días consecutivos trabajando.

A la hora de encontrar una solución para este problema, se siguen dos enfoques principalmente:

- **Cíclico o rotatorio**, en el cual, teóricamente, todos los trabajadores realizarían los mismos turnos, ver (Caprara et al. 1998).

- **No-cíclico o personal**, el cual considera las preferencias personales de los trabajadores por lo que es bastante más flexible, ver (Bianco & Spadoni 1992).

La mayor parte de los estudios realizados sobre el tema, se centran en aplicaciones al caso de autobuses urbanos (ver (Carraresi & Gallo 1984) donde el número de grupos de trabajo necesarios para realizar las tareas se puede determinar fácilmente, y se busca distribuir la carga de trabajo de forma equitativa entre los distintos grupos. Aunque también existen autores que lo han aplicado al ámbito ferroviario, un buen ejemplo de ello lo encontramos en (Ernst & Dowling 1998), quienes presentan un método heurístico basado en

el recocido simulado (Kirkpatrick et al. 1983) aplicado al sistema nacional de ferrocarril australiano.

Dentro de la literatura, también existen autores que han tratado de integrar conjuntamente ambos problemas (Crew Scheduling y Rostering), tales como (Ernst et al. 2001) y (Freling et al. 2004).

1.2.3 Problemas operacionales

En este nivel, se busca ajustar la planificación táctica ya descrita anteriormente, a las características específicas de cada semana, tratando de conseguir un buen funcionamiento del sistema a corto plazo y en tiempo real.

Tal y como describe (Maróti 2006), en este nivel no existen grandes cargas de trabajo con relación al problema de establecimiento de frecuencias y horarios de los trenes, debido a que la mayoría fueron ya fijados durante los niveles superiores de planificación estratégica y táctica. Fundamentalmente, lo que se hace ahora es reajustar los horarios y frecuencias para adaptarlos a la demanda específica de cada semana. Dichas modificaciones se deben a circunstancias especiales como pueden ser eventos culturales, competiciones deportivas, conciertos, congresos, fiestas locales o nacionales, etc. También puede ocurrir que se produzcan algún tipo de fallo, avería o retraso que obligue a modificar los horarios establecidos.

Cuando ocurren situaciones como las mencionadas anteriormente, puede que sea necesario el tener que añadir o quitar trenes, así como modificar los horarios de los ya programados, de modo que el sistema pueda adaptarse a dichas circunstancias o imprevistos sin alteraciones importantes en su funcionamiento habitual.

Entre los autores que han tratado este problema, encontramos a (Hirai et al. 2006), quienes proponen un algoritmo para la reprogramación automática de los trenes en caso de que se produzca algún accidente, avería o desastre natural que altere gravemente el tráfico habitual. Recientemente, (Shakibayifar et al. 2017) restructuran la programación establecida en busca de minimizar los retrasos que se producirán debido a un bloqueo en una línea de la red.

Al igual que ocurre con la determinación de horarios, ya existe una planificación establecida a nivel táctico para la programación de la tripulación. Sin embargo, cuando ocurren alteraciones en los turnos ya establecidos a causa de enfermedades, retrasos,

cancelación de servicios u otro tipo de imprevistos, es necesaria una restructuración del calendario determinado con anterioridad.

No debemos restarle importancia a este tipo de problema, ya que, si un tren no cuenta con los empleados necesarios, no podrá realizar su salida programada y deberá ser retrasado o incluso cancelado, teniendo que reubicar a los pasajeros en otros trenes.

Además de las ya mencionadas con anterioridad, pueden darse situaciones extraordinarias que impliquen una mayor restructuración de todo el sistema tanto a nivel de horarios, como personal y material rodante, y que obligan a reaccionar muy rápidamente, provocando en muchas ocasiones que el sistema no proporcione el nivel de servicio adecuado. Se da por ejemplo en situaciones de climatología extremas o huelgas del personal.

2. ESTABLECIMIENTO DE HORARIOS

Habiendo realizado un repaso general de cada uno de los diferentes problemas que componen el proceso de planificación de un sistema ferroviario, a continuación, pasamos a profundizar un poco más en el proceso de establecimiento de horarios, problema que tendrá especial relevancia en el presente trabajo. Tal y como vimos anteriormente, el establecimiento de los horarios se trataba en el problema de programación de trenes, correspondiente, a su vez, a la fase de planificación táctica.

La decisión final para este tipo de problemas, en el caso de redes de larga distancia, la tendrá el planificador o administrador central, Adif en el caso de España. Tras haber recibido las preferencias de horarios de las diferentes compañías de transporte, este deberá diseñar y establecer los horarios que mejor se ajusten a dichas preferencias. En el caso de redes de Metro o cercanías, las decisones dependen fundamentalmente de los operadores del servicio de transporte, influidos en muchos casos por las administraciones encargadas de velar por el servicio público y, en su caso, aplicando ciertas reglas de prioridad en favor de los servicios de larga distancia en el caso de que se compartan infraestructuras.

En este problema, el objetivo es establecer un calendario para determinar los horarios de salida y llegada de los trenes en las diferentes estaciones o puntos relevantes dentro de la red ferroviaria. La elaboración de este calendario se realiza para responder a toda la demanda del servicio, pero también debe respetar una serie de requisitos. Entre ellos, (Kinder 2008) destaca:

- **Tiempo de espera (dwell time)**. Cuando un tren llega una estación, tiene que parar un tiempo mínimo para que los pasajeros dispongan del tiempo suficiente para subirse, colocar equipaje, etc.

- **Limitaciones de seguridad** como, por ejemplo, respetar la distancia mínima de seguridad entre dos trenes que comparten una misma vía para evitar adelantamientos y posibles colisiones.

- **Limitaciones de capacidad** asociadas a la red ferroviaria, el material rodante y la legislación.

- **Planificación jerárquica**: los sistemas de transporte normalmente se programan siguiendo unas preferencias, por ejemplo, los trenes nacionales tienen prioridad sobre los locales. De esta forma, primero serán programados los trenes con alta prioridad, y posteriormente, se pasa a programar los restantes sin alterar los horarios de los ya programados.

Tal y como se presenta en (Arenas et al. 2015), este tipo de restricciones no pueden ser violadas (hard-contraints), sin embargo, existen otro tipo de restricciones (soft-constraints) que pueden no cumplirse, las cuales están más bien relacionadas con la calidad del servicio ofertado, y que suelen llevar asociadas ineficiencia y pérdidas económicas. La combinación de estos requisitos y limitaciones, así como la cantidad de trenes y horarios que se deben planificar, hace que la preparación de un calendario factible sea un proceso complejo y lento, que normalmente tarda varios meses en completarse, de ahí que este problema se aborde en la fase de planificación táctica.

Teniendo en cuenta lo anterior, se pueden distinguir cuatro tipos de problemas: programación cíclica, programación no-cíclica, un híbrido entre ambos, y finalmente, programación regular.

2.1 Programación cíclica

En la programación cíclica (también conocida como clock-faced) los trenes se agrupan en conjuntos, de forma que los trenes de cada conjunto (línea) tienen la misma ruta, y cumplen con un horario que detalla la hora de entrada y salida en cada estación, el cual, contempla además el tiempo de parada en cada una de ellas, de tal forma que dicho horario se repite de manera periódica formando lo que se denomina un ciclo, que se va repitiendo a lo largo del tiempo (Caprara et al. 2007). En el caso del servicio ferroviario de pasajeros, un tipo especial de horarios cíclicos es el horario de reloj, donde el ciclo es de una hora.

Un ejemplo de este tipo de horario puede verse a continuación en la Tabla 1. Esta nos muestra el primer horario cíclico creado en la historia, establecido para un tren que opera entre dos ciudades de los Países Bajos, La Haya y Venlo (ver figura 5) en 1931. Se puede observar como la conexión se establece exactamente en los mismos minutos de cada hora del día, exceptuando el primer tren de la mañana.

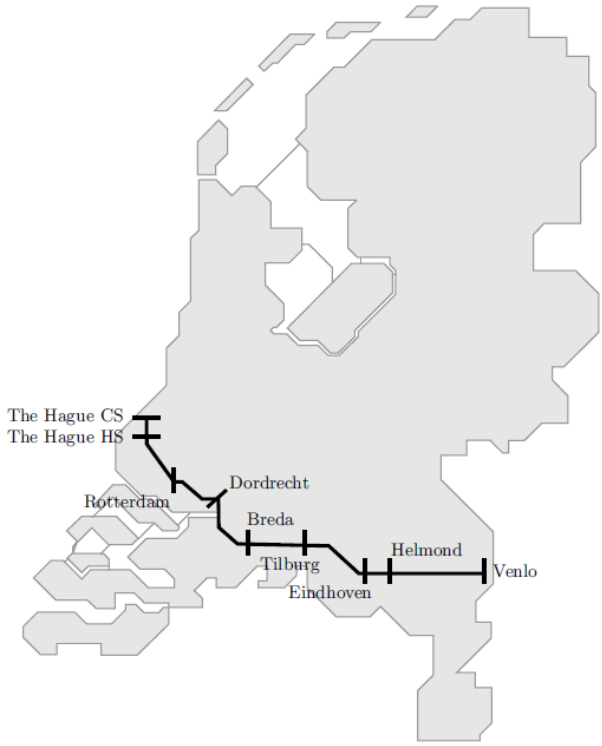


Figura 5. Recorrido del tren Intercity 1900 (Peeters 2003)

The Hague CS	-	06:21	07:21	...	20:21	21:21	22:21
The Hague HS	-	06:25	07:25	...	20:25	21:25	22:25
Rotterdam	05:41	06:45	07:45	...	20:45	21:45	22:45
Dordrecht	06:01	07:01	08:01	...	21:01	22:01	23:01
Breda	06:19	07:19	08:19	...	21:19	22:19	23:19
Tilburg	06:33	07:33	08:33	...	21:33	22:33	23:33
Eindhoven	07:01	08:01	09:01	...	22:01	23:01	23:58*
Helmond	07:11	08:11	09:11	...	22:11	23:11	-
Venlo	07:32*	08:32*	09:32*	...	22:32*	23:32*	-

Tabla 1. Horario del Intercity 1900 desde La Haya a Venlo (Peeters 2003)

Los horarios cíclicos tienen principalmente tres ventajas: la primera, descrita en trabajos como los de (Cardaso & Marín 2012), está destinada a los usuarios del servicio, ya que de esta forma no tienen la necesidad de memorizar horarios complejos para sus conexiones normales, les basta con memorizar los minutos de la hora a la que salen los trenes (suponiendo que el ciclo sea de una hora). De hecho, algunos estudios confirman que la regularidad de un calendario conduce a un aumento en la demanda de pasajeros (Wardman et al. 2004)**,** aunque actualmente con el uso tan extendido del internet en los teléfonos móviles, esta ventaja está perdiendo cada vez más fuerza. Otra ventaja es que, si se implementa el concepto cíclico de forma estricta, los pasajeros no se enfrentan a carencias en los servicios de trenes en las partes del día en que la demanda de transporte es baja.

Desde un punto de vista de la planificación, los horarios cíclicos tienen la ventaja de que sólo se tiene en cuenta un período de un ciclo, es decir, el horizonte de planificación es pequeño y por tanto la construcción del horario es más sencilla, de tal forma que el calendario de un día completo se puede obtener mediante la copia del horario cíclico para todas las horas pertinentes del día.

Sin embargo, como principal desventaja hay que señalar que este sistema es mucho más caro de operar, llegando a resultar en ocasiones ineficiente. Esto es debido a que, al no realizar ninguna distinción entre los diferentes periodos del día, operan el mismo número de trenes en las horas puntas del día que en aquellas donde la demanda es baja (Caprara et al. 2007). Además, resulta complicado planificar dicho tipo de calendario en un mercado competitivo, donde son muchos los operadores de transporte que acceden a la misma infraestructura ferroviaria.

La mayoría de los autores que estudian la programación cíclica se basan en los modelos utilizados para el Problema de Programación de Eventos Periódicos (más conocido por sus siglas en inglés, PESP), definidos por primera vez por (Serafini & Ukovich 1989). En dicho problema, un conjunto dado de eventos se programa en intervalos igualmente espaciados. Más adelante, (Nachtigall 1994) también utilizará las restricciones PESP para crear su propio modelo de formulación de periodicidad cíclica, teniendo como objetivo la minimización del tiempo de espera de los pasajeros. Este se basa en la construcción de un gráfico auxiliar, cuyos nodos se corresponden con los eventos (salidas y llegadas de trenes) y cuyos arcos modelan las restricciones en el tiempo que separan los eventos (por ejemplo, para satisfacer los requisitos de seguridad). (Nachtigall & Voget 1996) buscarán la minimización del tiempo de espera de los pasajeros para este tipo de problemas cíclicos, generando primero un horario de forma heurística y posteriormente mejorándolo mediante el uso de algoritmos genéticos. (Odijk 1996) también utiliza el modelo PESP, pero no con el objetivo de construir un calendario, sino más bien, su propósito es establecer especificaciones para la ampliación de la infraestructura dentro y alrededor de la estación.

2.2 Programación no-cíclica

En el calendario no cíclico, por otro lado, no se impone ninguna regla especial en el horario de salida de los trenes, se puede formular de manera flexible (ver (Caprara & Toth 2002), por ejemplo).

Cuando se trata con un mercado competitivo en el que se deben compaginar multitud de servicios de diferentes operadores en infraestructuras con unas limitaciones dadas, este tipo de programación del se vuelve más apropiado, ya que varios operadores de trenes solicitan la programación de un conjunto de trenes de acuerdo con un horario preferido y se hace más difícil respetar un período de tiempo fijo al mismo tiempo que se satisfacen las preferencias de cada operador.

Otra ventaja es la disminución de coste que supone para la empresa, ya que, al trabajar en función de la demanda, se puede reducir el tamaño de la flota optimizando la asignación de recursos. Es por ello por lo que a este tipo de horarios se le considera como un calendario dirigido por el mercado y por la demanda. Como consecuencia, un horario que se guie estrictamente por la demanda puede dar lugar a un calendario complejo que es difícil de consultar y memorizar. Por otra parte, el concepto de horario no cíclico requiere la planificación de un horario para cada día de la semana. Esto genera problemas de planificación muy grandes, de ahí que su uso quede restringido a sistemas de transporte de mercancías y trenes de media o larga distancia con frecuencias deseadas ni muy altas ni muy bajas.

Según (Caprara et al. 2007), las características esenciales del proceso pueden ser resumidas de la siguiente forma:

- Se trabaja de forma que cada operador asocia un beneficio a cada tren y propone su horario ideal, definiendo los horarios de salida y llegada de las estaciones visitadas, y las tolerancias dentro de las cuales se pueden realizar cambios.

- El administrador de infraestructura construye el horario óptimo maximizando el beneficio total, dado por la prioridad global de los trenes programados y viéndose penalizado por una función que tiene en cuenta las modificaciones del horario final con respecto al horario ideal.

En relación con los trabajos más relevantes realizados sobre el tema, (Szpigel 1973) fue uno de los primeros que trató el problema de los horarios no cíclicos, proponiendo una formulación para la planificación del problema de los horarios de los trenes en una única vía de ferrocarril, con el objetivo de minimizar los tiempos de viaje en tren. (Jovanovic & Harker 1991) resolvieron mediante Branch and Bound una versión del modelo anterior, con

el objetivo de diseñar un horario fiable de trenes más que de satisfacer una determinada función objetivo. Poco más tarde, (Cai & Goh 1994) desarrollaron un algoritmo heurístico constructivo para la programación de una sola línea de trenes. Suponiendo que todos los trenes que viajan en la misma dirección tienen la misma velocidad, el algoritmo determinaba los grupos de trenes en conflicto y los costos asociados a retrasos. (Mistry & Kwan 2003) también abordan el tema, presentando un algoritmo para la generación automática de los horarios.

2.3 Programación híbrida

Además, (Robenek et al. 2017) proponen en su trabajo otro tipo de programación a la que denominan híbrida, la cual combina los beneficios de las dos anteriores: cíclica y no cíclica.

Con este tipo de horario, buscan combinar la regularidad de los horarios cíclicos, con la flexibilidad que ofrecen los no cíclicos. La regularidad se incorpora en su diseño y la flexibilidad se evalúa comprobando la satisfacción de los pasajeros (en unidades monetarias). Este horario consta del 75% de los trenes cíclicos (asegurando la regularidad del servicio) y del 25% de trenes no cíclicos (desplegados como trenes suplementarios durante las horas punta y capturando la fluctuación de la demanda).

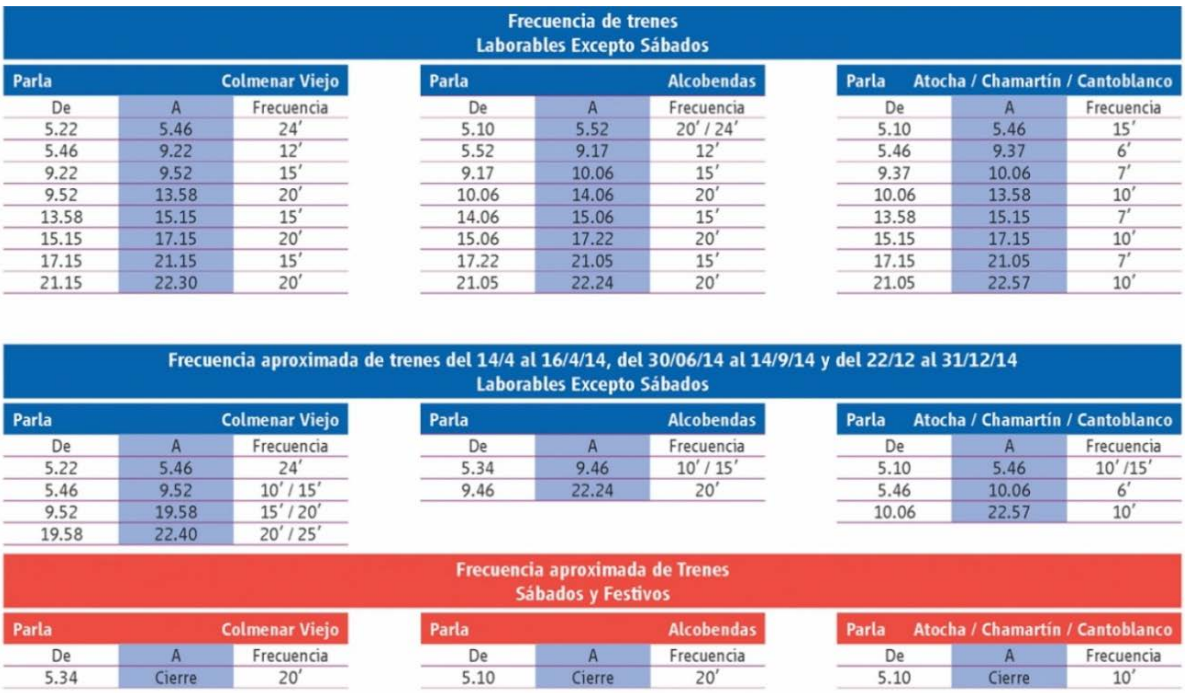
Los horarios híbridos los obtienen resolviendo lo que ellos denominan “Problema de horario de trenes centrado en el pasajero (Passenger Centric Train Timetabling Problem, PCTTP)”, utilizando una heurística de búsqueda de vecindad combinada con recocido simulado. La eficacia de cada uno de los nuevos horarios propuestos se evalúa y se compara con los horarios cíclicos y no cíclicos reales existentes en la red de ferrocarriles de Israel, con el propósito de comprobar el impacto que tiene en la satisfacción de los pasajeros, obteniendo, según los autores, unos resultados bastante prometedores.

2.4 Programación regular

Cuando la frecuencia del servicio es lo suficientemente alta, resulta más importante especificar dicha frecuencia, que detallar las horas exactas de llegada y partida de los trenes a cada estación (Higgins & Kozan 1998).

En estos casos se suele usar la programación regular, en la cual, a diferencia de las anteriores, se da como información el tiempo que transcurre entre dos trenes consecutivos (headway time), el cual se obtiene simplemente realizando el inverso de la frecuencia de

servicio deseada, es decir, si por ejemplo se quiere tener una frecuencia de 5 trenes por hora, el tiempo entre dos trenes consecutivos que llegan a una misma estación debería ser de 12 minutos. Cabe resaltar que este tipo de programación nos permite tener en cuenta las diferentes horas del día, ya que la frecuencia deseada no será la misma en hora punta que en aquellas horas con baja demanda. Esta situación podemos observarla en la siguiente imagen, en la cual se muestran las frecuencias detallas del servicio en función del día de la semana y de la hora del día para la línea 4 de la red de cercanías de Madrid.



Frecuencia de trenes
Laborables Excepto Sábados

Parla		Colmenar Viejo
De	A	Frecuencia
5.22	5.46	24′
5.46	9.22	12′
9.22	9.52	15′
9.52	13.58	20′
13.58	15.15	15′
15.15	17.15	20′
17.15	21.15	15′
21.15	22.30	20′

Parla		Alcobendas
De	A	Frecuencia
5.10	5.52	20′ / 24′
5.52	9.17	12′
9.17	10.06	15′
10.06	14.06	20′
14.06	15.06	15′
15.06	17.22	20′
17.22	21.05	15′
21.05	22.24	20′

Parla		Atocha / Chamartín / Cantoblanco
De	A	Frecuencia
5.10	5.46	15′
5.46	9.37	6′
9.37	10.06	7′
10.06	13.58	10′
13.58	15.15	7′
15.15	17.15	10′
17.15	21.05	7′
21.05	22.57	10′

Frecuencia aproximada de trenes del 14/4 al 16/4/14, del 30/06/14 al 14/9/14 y del 22/12 al 31/12/14
Laborables Excepto Sábados

Parla		Colmenar Viejo
De	A	Frecuencia
5.22	5.46	24′
5.46	9.52	10′ / 15′
9.52	19.58	15′ / 20′
19.58	22.40	20′ / 25′

Parla		Alcobendas
De	A	Frecuencia
5.34	9.46	10′ / 15′
9.46	22.24	20′

Parla		Atocha / Chamartín / Cantoblanco
De	A	Frecuencia
5.10	5.46	10′ /15′
5.46	10.06	6′
10.06	22.57	10′

Frecuencia aproximada de Trenes
Sábados y Festivos

Parla		Colmenar Viejo
De	A	Frecuencia
5.34	Cierre	20′

Parla		Alcobendas
De	A	Frecuencia
5.10	Cierre	20′

Parla		Atocha / Chamartín / Cantoblanco
De	A	Frecuencia
5.10	Cierre	10′

Figura 6. Frecuencia de trenes para la línea C-4 de Madrid.
(Recuperado de https://en.redtransporte.com)

La situación mencionada tiene lugar especialmente en los sistemas de tránsito rápido, tales como redes de metro y cercanías. Debido a que el modelo que se desarrollará posteriormente en este trabajo tratará sobre este último tipo de sistemas (cercanías), más adelante se profundizará sobre este tema.

3. PROBLEMA PROPUESTO

3.1 Introducción al problema

Este trabajo tiene como objeto de estudio un sistema de tránsito rápido por ferrocarril (como es el caso de las redes de cercanías y metro) más conocido por su terminología en inglés Railway Rapid Transit Systems (RRT Systems). Este tipo de sistemas de transporte masivo está electrificado y, a diferencia de otros medios de transporte urbanos, se caracteriza por operar en raíles de uso exclusivo, a los cuales no tienen acceso vehículos ni peatones. De forma general, suelen circular subterráneamente, aunque también pueden hacerlo a nivel del suelo o sobre vías elevadas (información extraída de https://www.britannica.com/technology/rapid-transit). Principalmente, funcionan en las áreas metropolitanas de ciudades que cuentan con una población elevada, permitiendo a sus usuarios viajar tanto a zonas urbanas como hasta a las áreas más periféricas de la ciudad.

Concretamente, la principal contribución de este trabajo va a consistir en el desarrollo de un modelo que permita determinar las frecuencias de paso óptimas para las diferentes líneas que componen una red de cercanías, junto con el resto de variables necesarias para programar en el tiempo cada una de ellas, de tal forma que se satisfaga al mismo tiempo tanto a operadores como a pasajeros.

El diseño de frecuencias en una red ferroviaria ha sido tratado en numerosos estudios, resultando tradicionalmente producto de la fase de planificación de líneas (Bussieck et al. 2004). El enfoque más común, consiste en diseñar las frecuencias de paso, de tal forma que se consiga atender la máxima demanda sin exceder la capacidad (Ceder 1984). Otros trabajos como el de (Scheele 1980), buscan minimizar el tiempo total de viaje de los pasajeros. Para ello, formula un problema no lineal donde introduce variables de decisión para determinar las frecuencias, restricciones de capacidad y de tamaño de flota; resolviendo simultáneamente el problema de asignación de pasajeros, del cual se hablará posteriormente. (Han & Wilson 1982) formulan un problema similar, pero buscando minimizar el segmento con mayor carga. Otros trabajos también destacables, han sido publicados más recientemente, como es el caso de (Hu & Liu 2014), quienes desarrollan un modelo de cálculo de “headways” para un sistema de tránsito rápido, el cual, busca evaluar los efectos de la frecuencia del servicio tanto en los costes operacionales, como en el tiempo de espera de los pasajeros evaluado en términos monetarios.

Con relación al problema que se va a estudiar en este trabajo, merece la pena resaltar que la red de transporte escogida está compuesta por varias líneas. Cuando solamente se cuenta con una línea en la red, el pasajero simplemente espera hasta que llegue el siguiente

tren, sin embargo, cuando se tienen varias líneas, en la mayoría de los casos los pasajeros tendrán más de una opción disponible para llegar al destino deseado, es decir, múltiples rutas a escoger, a las cuales se hará referencia más adelante como estrategias de viaje. Este concepto de estrategia fue introducido por (Spiess & Florian 1989), buscando hacer referencia a la elección de un conjunto de trayectorias usando diferentes líneas, cuya combinación permite al usuario alcanzar el destino deseado. Por lo tanto, el desarrollo de un modelo de asignación de tránsito será muy útil para predecir cómo se comportan los distintos usuarios dada una red de transporte. Este nos permitirá asignar rutas a los distintos pasajeros, y a su vez, también ayudará a identificar aquellos tramos de viaje con mayor congestión, o aquellos que, por el contrario, tienen un nivel de uso muy bajo (Hamdouch & Lawphongpanich 2008).

Durante décadas, los planificadores e investigadores del transporte han ideado diferentes modelos de asignación de tránsito, ver (Nuzzolo 2003) para una buena recopilación de estos. En este sentido, tal y como nos muestran (Friedrich & Wekeck 2004) en su trabajo, el problema ha sido modelado desde dos puntos de vista:

- **Frequency-based o line-based:** simplifica la búsqueda mediante el uso de valores medios para los tiempos de transferencia entre líneas, como consecuencia los resultados son agregados. Cada línea de tránsito se modela a través de una sucesión de paradas, de tiempos de recorrido entre paradas y del “headway” de la línea (el “headway” hace referencia a la separación temporal entre trenes consecutivos). Los procedimientos utilizados en este enfoque no calculan explícitamente los tiempos de transferencia, sino que asumen que estos dependen del “headway” entre trenes. Normalmente, suele asumirse que los tiempos de espera en la estación de inicio del viaje y las de transferencia son iguales a la mitad del “headway” de la línea (o mitad del inverso de la frecuencia). Este tipo de modelo proporciona buenos resultados cuando se aplica en áreas urbanas con una red de tráfico densa.

- **Timetable-based**: considera la construcción de un horario para cada línea de la red con unas horas de partida y llegada exactas (Friedrich 1994). La mayoría de los modelos propuestos, aplican algún tipo de algoritmo para el cálculo del camino más corto que, dado un horario de partida, determina la mejor conexión entre dos zonas. Para diferentes horarios de partida, puede haber diferentes conexiones dependiendo de las líneas de tránsito y/o de las paradas para trasbordar utilizadas. Este tipo de asignación es muy adecuada cuando se tienen valores precisos de la frecuencia de servicio, tiempo de transferencia y cargas de los diferentes vehículos y suele aplicarse en zonas rurales donde la frecuencia del servicio es baja.

Concretamente, en el contexto de los ferrocarriles, la programación de los trenes a lo largo del tiempo se suele realizar determinando los horarios de salida y llegada para cada estación, no obstante, cuando tratamos con redes de tránsito rápido (como es el caso de este

proyecto), son muchos los servicios que se tienen que programar a lo largo de todo el día para poder atender unos altos volúmenes de demanda, especialmente en horas punta, por lo que se suele realizar un diseño de horarios basado en frecuencias que permita asegurar el paso de un tren cada un cierto periodo de tiempo constante y usualmente pequeño. Este razonamiento puede constatarse en trabajos como el de (Schmöcker & Bell 2009), según el cual, para seleccionar un modelo u otro de asignación, se puede tomar como límite una diferencia de "headways" de 15 min aproximadamente. Si el servicio opera con un "headway" mayor, "schedule-based" será el modelo adecuado, sin embargo, tal y como ocurre en el caso que se considera en este proyecto, el "headway" es menor, y por lo tanto el modelo que mejor refleja la realidad será del tipo "frequency-based". De hecho, de acuerdo con el Teorema de Incidencia Aleatoria propuesto por (Larson & Odoni 1981), los horarios regulares se convierten en óptimos cuando se trata con patrones uniformes de llegada de pasajeros.

Finalmente, para el desarrollo del problema que se plantea, de forma similar a (Poon et al. 2004), se asume que todos los pasajeros tienen información acerca del funcionamiento de la red de transporte, la cual ha sido obtenida a partir de experiencia previa, por lo tanto, no escogerán rutas aleatorias para llegar a su destino, sino que la elección de una estrategia u otra dependerá del tiempo total de viaje, incluyendo el tiempo de transferencia y espera. De tal forma, que un pasajero probablemente rechazaría coger un camino que contenga una línea con una frecuencia de paso muy baja, ya que esto implicaría mayor tiempo de espera. Es por esto, que la frecuencia de paso va a determinar en gran medida el comportamiento de los distintos usuarios, como consecuencia será necesario fijar las frecuencias más adecuadas. Sin embargo, el diseño de las frecuencias va a depender a su vez de la demanda de pasajeros existente para cada línea de la red, existiendo obviamente una mayor frecuencia de paso en aquellas líneas con mayor demanda. Por lo tanto, el problema de asignación y el de determinación de frecuencias van a estar completamente relacionados, siendo ambos resueltos de forma secuencial, ver (Szeto & Jiang 2014), de tal forma que se realizarán sucesivas iteraciones hasta que se consiga alcanzar un equilibrio entre ambos.

3.2 Descripción general del problema

Antes de comenzar con la explicación detallada del problema propuesto y el procedimiento planteado para su resolución, se hace necesario realizar una pequeña explicación de todo el problema en su conjunto, de tal forma que se aborden ciertos conceptos que resultan fundamentales para facilitar el entendimiento del problema que posteriormente se planteará.

Supongamos que tenemos una red de tránsito rápido compuesta por un conjunto de líneas, cada una de ellas con sus correspondientes estaciones y vías. A su vez, habrá estaciones que pertenezcan a más de una línea, por lo tanto, se podrán realizar trasbordos en la red. Por otro lado, la red cuenta con una demanda determinada que viene dada en forma de Matriz Origen-Destino para un horizonte temporal de una hora. En base a dicha demanda, se busca realizar la mejor programación posible de las diferentes líneas que componen la red, de tal forma que se satisfaga a los operadores minimizando costes y, al mismo tiempo, se ofrezca a los usuarios una calidad del servicio adecuada medida en términos del tiempo de viaje. Para realizar la programación de cada línea, se recurrirá a la resolución de un modelo matemático de optimización que nos permitirá determinar, además de la frecuencia de paso, otra serie de variables necesarias, como son el tamaño de la flota, los tiempos de recorrido en cada vía, el modelo de tren y los tiempos de parada en cada estación. Sin embargo, antes de proceder a resolver el modelo, será necesario conocer cómo se comporta la demanda, ya que, como se mencionó anteriormente, debido a que existen estaciones compartidas por varias líneas, los pasajeros tendrán diferentes opciones para llegar a su destino. La elección por parte de los usuarios de tomar un camino u otro, dependerá del tiempo total de viaje que un determinado camino suponga en comparación con el resto de las opciones disponibles, no obstante, como inicialmente se realiza una asignación de tránsito sin haber resuelto el modelo de optimización (no se conocen por lo tanto frecuencias, tiempos de parada, etc.), esta primera asignación se realizará con la longitud total del viaje (suma de las longitudes de cada vía que compone el trayecto). Una vez se ha realizado la asignación, se conocerá el número de pasajeros que sigue cada estrategia (secuencia de líneas), y como consecuencia, la cantidad de personas que sube y baja en cada estación y la carga de pasajeros en cada vía de la red, todo ello para el horizonte temporal considerado en la matriz OD. Obtenidos estos datos, ya se podrá llevar a cabo la programación deseada.

A continuación, podemos observar una figura en la que se representan las diferentes variables necesarias para la programación de una línea a lo largo del tiempo en su paso por las diferentes plataformas que la componen.

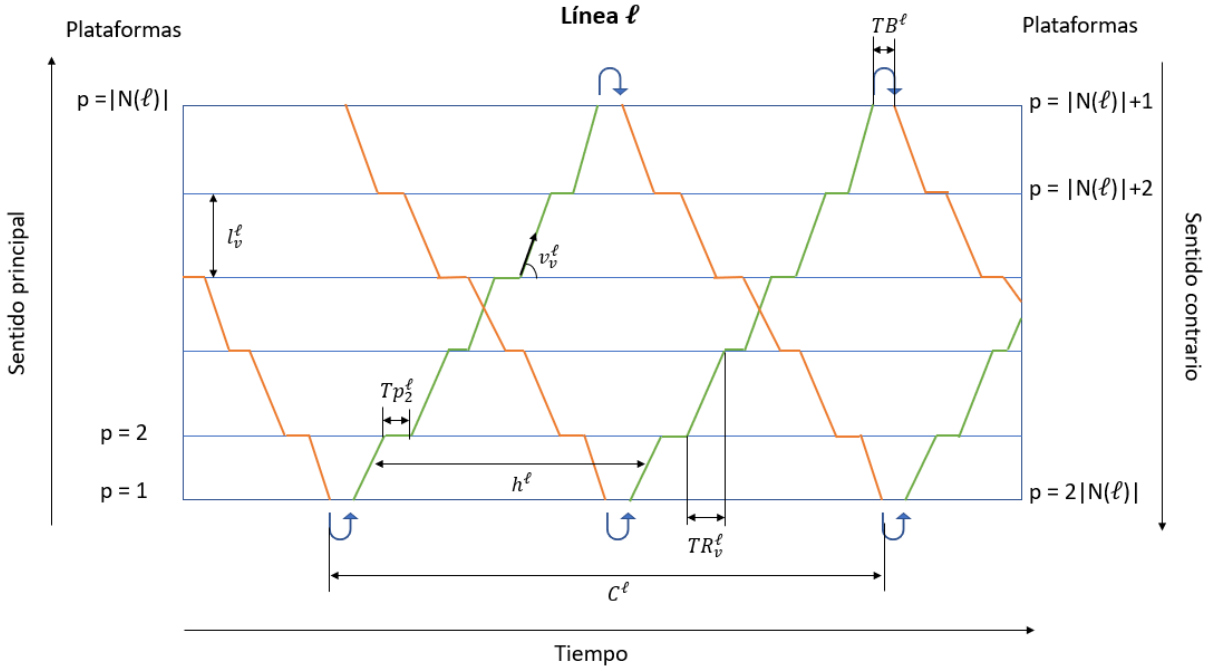


Figura 7. Representación de una línea a lo largo del tiempo.
Adaptación de (Canca et al. (Forthcoming))

De la figura anterior podemos obtener varias ideas importantes. Por un lado, se puede observar como en color verde se representa el sentido de ida de la línea ℓ, y, cuando se llega a la plataforma final de esta, tras un período de tiempo necesario para realizar el cambio de sentido denotado por TB^{ℓ}, cada tren continua su recorrido en sentido de vuelta, representado en color naranja. Por otro lado, cuando el tren llega a una plataforma, se detiene en esta un tiempo específico representado en la imagen mediante Tp_p^{ℓ}, para permitir la subida y bajada de todos los pasajeros que lo deseen (dato obtenido tras la asignación). Además, se observa que cada un cierto tiempo h^{ℓ}, parte un tren de la primera estación de la línea. Esta variable representa el intervalo de tiempo entre dos trenes consecutivos que, como se mencionó anteriormente, se denomina "headway". Este a su vez se corresponde con el inverso de la frecuencia de paso, la cual nos indica cuántos trenes partirán de la primera estación en el intervalo de tiempo considerado. Así mismo, en función de la frecuencia de paso que se establezca, se obtendrá un tamaño de flota (número de trenes que se necesitan para cubrir todos los servicios) u otro. Para determinarlo, habrá que tener en cuenta también el tiempo de ciclo de la línea (tiempo necesario para que un tren complete todo el recorrido en los dos sentidos), representado en la imagen mediante C^{ℓ}, ya que, cuando un tren haya completado todo su recorrido, volverá a estar disponible para partir de la primera estación. Como puede observarse, entran en juego también las velocidades en cada vía del recorrido v_v^l, debido a que cuanto mayores sean estas, menor será el tiempo de recorrido TR_v^l, lo que implica un menor tiempo de ciclo, y como consecuencia, se necesitará un tamaño de flota menor para una determinada frecuencia. Si, por el contrario, se mantuviese constante el tiempo de ciclo

y lo que modificamos es la frecuencia, se tendría de manera obvia que, a mayor frecuencia deseada, mayor tamaño de flota.

Junto con estas variables, también resulta necesario seleccionar un determinado modelo de tren (entre los disponibles) para cada línea, de tal forma que el modelo escogido, debe ser capaz de atender toda la demanda que tenga la línea en el horizonte temporal considerado. Para una determinada demanda, cuanto mayor sea la capacidad del tren, menor será la frecuencia de paso necesaria. Además, a mayor capacidad del tren (tamaño), mayor número de vagones, lo que implica un mayor número de puertas, y como consecuencia, se podrá establecer un menor tiempo de parada en cada estación a fin de permitir la subida y bajada de pasajeros.

Una vez se han obtenido tras la optimización del modelo todas las variables que acaban de ser mencionadas, estas son utilizadas para realizar una nueva asignación, que se llevará a cabo en esta segunda iteración, en base al tiempo total de viaje para cada par origen-destino. Con los nuevos datos de la asignación, se vuelve a resolver el modelo de optimización, y así sucesivamente, hasta que se consiga alcanzar una situación de equilibrio, controlada mediante cierto criterio de aceptación establecido.

Ahora ya sí, una vez se ha contextualizado todo el problema, pasa a exponerse en profundidad toda la información y notación necesaria para llevar a cabo la formulación del modelo matemático que nos permita determinar nuestro objetivo. Dentro de este apartado, también se explica detalladamente como se desarrolla el proceso de asignación de tránsito, el cual, a su vez, será necesario para poder resolver el modelo. Cabe resaltar que el problema que se muestra a continuación es una adaptación de los propuestos por (Canca & Zarzo 2017) y (Canca et al. (Forthcoming)).

3.3 Explicación detallada del problema

Se considera una red de cercanías representada por un grafo dirigido $G = \{N, A\}$, donde $N = \{1, 2, \ldots, N\}$ se corresponde con el conjunto de estaciones (representadas mediante nodos) y $A = \{(i,j) : i,j \in N\}$ es el conjunto de arcos dirigidos que unen cada par de estaciones de la red.

Sea L el conjunto de líneas de la red, cada línea $\ell \in L$ está compuesta de un subconjunto de nodos $N(\ell) \subset N$. Se define un sentido principal de forma arbitraria para cada línea $\ell \in L$, de tal forma que, el orden de los elementos en N(ℓ) coincide con el orden que siguen las estaciones para dicha línea. Por conveniencia, denotaremos al conjunto de

arcos que conectan cada par de estaciones en $N(\ell)$ como A(ℓ), conteniendo este conjunto tanto los arcos del sentido principal, como los del contrario.

Se denota por $L_{(i)}$ al conjunto de líneas que circulan por la estación $i \in N$. Por lo tanto, el conjunto de nodos de transferencia, es decir, aquellos en los cuales se puede trasbordar hacia otras líneas de la red, es $T = \{i \in N : |L_{(i)}| > 1\}$. A su vez, se expresa mediante $L_{(i,j)}$ al conjunto de líneas que atraviesan el arco $(i,j) \in A$. Para este caso concreto, cuando dos o más líneas circulan por un mismo arco $(i,j) \in A$, se considerará que cada línea dispone de su propia infraestructura (podría darse el caso en el que las líneas compartiesen vía, sin embargo, esta situación queda fuera del alcance de este trabajo por la complejidad añadida que supone).

3.3.1 Plataformas y vías

Como las líneas son bidireccionales, se consideran dos plataformas para cada estación, una correspondiente al sentido principal y otra, al contrario. Para una línea $\ell \in$ L, se tiene que $P^{\ell} = \{1, 2, \ldots, 2|N(\ell)|\}$ es el conjunto de plataformas, de forma que las primeras $|N(\ell)|$ plataformas se corresponden con el sentido principal de la línea, y aquellas que van desde $|N(\ell)| + 1$ hasta $2|N(\ell)|$, con el sentido inverso. En el modelo propuesto, se considera que cada línea tiene dos plataformas propias para cada estación por la que circule, por ello, como consecuencia de la definición de la red, nunca coincidirán dos líneas en una misma plataforma.

Por ejemplo, si considerásemos una línea ℓ_1 con tres estaciones, $N(\ell_1) = \{4, 9, 14\}$, el conjunto de plataformas correspondientes sería $\mathrm{P}^{\ell_1} = \{1, 2, 3, 4, 5, 6\}$, donde las plataformas 1, 3 y 5 se corresponden con el sentido principal de la línea, y las plataformas 2, 4 y 6 con el sentido inverso a este. Concretamente, la estación 4 al ser la primera de la línea, contendría las plataformas 1 y 2, la estación 9 al ocupar la segunda posición, las plataformas 3 y 4, y, por último, la estación 14 estaría formada por las plataformas 5 y 6.

Para el posterior cálculo de subidas y bajadas en las diferentes plataformas de la red, será necesario definir un operador que devuelva la estación a la que una plataforma pertenece. Este operador se representará como $s^{\ell}(p)$. Por ejemplo, para el caso anterior, tendríamos que $s^{\ell_1}(1) = s^{\ell_1}(4) = 4,\ s^{\ell_1}(2) = s^{\ell_1}(5) = 9\ y\ s^{\ell_1}(3) = s^{\ell_1}(6) = 14$.

Por otro lado, dada una línea ℓ de la red, $V^{\ell} = \{1, 2, \ldots, |N(\ell)| - 1, |N(\ell)| + 1, \ldots, 2|N(\ell)| - 1\}$ es el conjunto de vías unidireccionales que unen cada par de plataformas, de tal forma que las primeras $|N(\ell)| - 1$ vías se corresponden con el sentido principal, y las siguientes desde $|N(\ell)| + 1$ hasta $2|N(\ell)| - 1$ con el sentido inverso. Cabe

resaltar que los números $|N(\ell)|$ y $2|N(\ell)|$ no están incluidos en el conjunto V^{ℓ} (podrían ser incluidos dentro del conjunto anterior para denotar las infraestructuras necesarias para el cambio de sentido en las estaciones $|N(\ell)|$ y 1 respectivamente, sin embargo, para el caso que estamos considerando no será necesario).

A continuación, se muestra una imagen que permite una mejor comprensión de lo descrito.

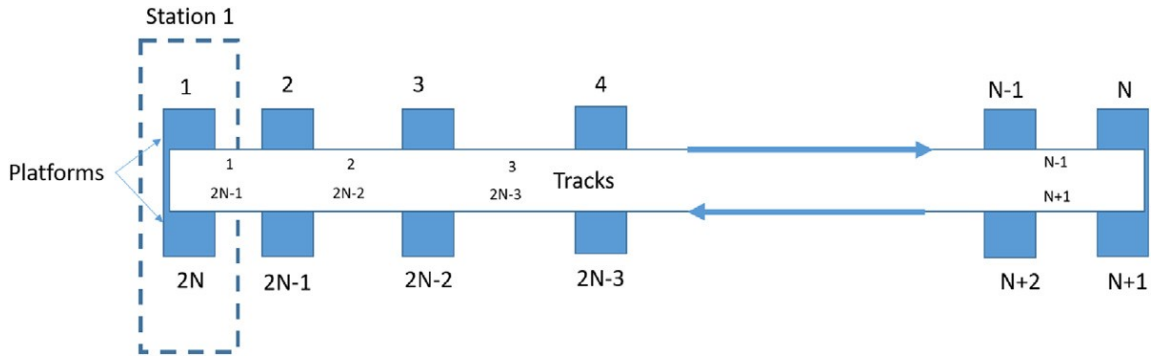


Figura 8. Esquema general de una línea (Canca & Zarzo 2017)

A su vez, cada vía $v \in V^{\ell}$ estará caracterizada por una longitud l_v^{ℓ} (metros). Se considera que esta tiene el mismo valor para el sentido de ida que para el de vuelta, es decir, para una línea ℓ, la vía que une la plataforma 1 con la 2, tiene la misma longitud que la que une 2 con 1. Al mismo tiempo, cada vía también tendrá asociada unos límites de velocidad $V_v^{\ell,min}$ y $V_v^{\ell,max}$, impuestos por temas tanto técnicos como de seguridad.

Finalmente, se va a denotar por $v^{\ell}(i,j)$ un operador que permite obtener la vía correspondiente a cada uno de los arcos que componen las líneas.

3.3.2 Demanda

Sea $W = \{(r,s) : \forall\, r,s \in N : r \neq s\}$ el conjunto de pares origen-destino (OD) de la red de cercanías considerada, de tamaño $|N|\ x\ |N-1|$. Para cada par OD $w = (r,s) \in W$, se denotan como $w_o = r$ y $w_d = s$, las estaciones origen y destino del par, respectivamente. La demanda de los pasajeros está representada por una matriz OD cuadrada, que tiene tantas filas y columnas como estaciones hay en la red, $|N|$. El elemento $O_{i,j}$ correspondiente al par OD $w = (i,j)$, representa el número de pasajeros que viaja durante un cierto intervalo de tiempo desde la estación $w_o = i$ a la estación $w_d = j$. En este caso concreto se ha escogido un intervalo de una hora, aunque podría haberse escogido

cualquier otro intervalo horario. De manera obvia, la diagonal de la matriz estará compuesta por ceros. Además, cuando se necesite $O_{i,j}$ se denotará también como g_w.

A continuación, se muestra una tabla donde se resumen los parámetros y conjuntos usados en relación con la definición de la red.

$\boldsymbol{N}$	*Conjunto de estaciones de la red de cercanías*
$\boldsymbol{N(\ell)}$	*Conjunto de estaciones de la línea* ℓ
$\boldsymbol{A}$	*Conjunto de arcos de la red de cercanías*
$\boldsymbol{A(\ell)}$	*Conjunto de arcos de la línea* ℓ
$\boldsymbol{L}$	*Conjunto de líneas de la red de cercanías*
$\boldsymbol{T}$	*Conjunto de estaciones de transferencia de la red de cercanías*
$\boldsymbol{L_{(i,j)}}$	*Conjunto de líneas que atraviesan el segmento* $(i,j) \in A$
$\boldsymbol{L_{(i)}}$	*Conjunto de líneas que circulan por la estación* $i \in N$
$\boldsymbol{P^{\ell}}$	*Conjunto de plataformas de la línea* ℓ, $P^{\ell} = 1, 2, \ldots, 2\|N(\ell)\|\}$
$\boldsymbol{s^{\ell}(p)}$	*Operador que devuelve la estación a la que pertenece la plataforma* $p \in P^{\ell}$
$\boldsymbol{V^{\ell}}$	*Conjunto ordenado de las vías de la línea* ℓ $V^{\ell} = \{1, 2, \ldots, \|N(\ell)\| - 1, \|N(\ell)\| + 1, \ldots, 2\|N(\ell)\| - 1\}$
$\boldsymbol{v^{\ell}(i,j)}$	*Operador que aplicado a un arco* $(i,j) \in A(\ell)$, *devuelve la vía correspondiente* $v \in V^{\ell}$
$\boldsymbol{l_v^{\ell}}$	*Longitud de la vía* $v \in V^{\ell}$
$\boldsymbol{V_v^{\ell,min}}$	*Velocidad mínima permitida para la vía* $v \in V^{\ell}$
$\boldsymbol{V_v^{\ell,max}}$	*Velocidad máxima permitida para la vía* $v \in V^{\ell}$
$\boldsymbol{W}$	*Conjunto de pares Origen-Destino de la red de cercanías*
$\boldsymbol{w_o}$	*Estación origen del par OD* $w \in W$
$\boldsymbol{w_d}$	*Estación destino del par OD* $w \in W$
$\boldsymbol{O_{ij}}$	*Demanda entre las estaciones* $i, j \in W$
$\boldsymbol{g_w}$	*Demanda del par OD* $w \in W$ *Si* $w_o = i$ *y* $w_d = j$, $g_w = \boldsymbol{O_{ij}}$

Tabla 2. Principales conjuntos y datos de entrada

3.3.3 Caminos y estrategias

A la hora de realizar la asignación de tránsito, será necesario conocer cuáles son los posibles caminos y estrategias que los pasajeros pueden tomar, y, además, cual es la probabilidad de que escoger cada uno de ellos.

Dado un par OD específico $w = (w_o,\ w_d)$, se representa por P^w al conjunto de caminos que conectan las estaciones w_o y w_d, definidos sobre el grafo dirigido G. De tal forma que, cada camino $p \in P^w$ se define como una sucesión de arcos en A que conectan el origen y el destino del par w.

Para un determinado camino $p \in P^w$, se define π_p^w como el conjunto de estrategias que los pasajeros pueden seguir en base a la combinación de las diferentes líneas de la red que atraviesan los arcos que componen el camino p. Para entenderlo mejor, imaginemos que tenemos un par OD $w = (2,\ 5)$, y uno de sus posibles caminos $p \in P^w$, está compuesto por los arcos (2, 3), (3, 4), (4, 5). Por el primero de ellos pasa solamente la línea 1, sin embargo, por el segundo y el tercer arco pasan las líneas 1 y 2, y las líneas 2 y 3 respectivamente. Por lo tanto, combinando las diferentes posibilidades, tendríamos 4 posibles estrategias a seguir, las cuales son $\pi_p^w = \{\ (L1, L1, L2),\ (L1, L2, L2),\ (L1, L1, L3),\ (L1, L2, L3)\ \}$.

Habitualmente, dado un camino $p \in P^w$, algunas de las posibles estrategias dominarán al resto, debido a que su elección supondrá a los pasajeros la realización de un menor número de trasbordos o a que están compuestas por un menor número de estaciones, lo que normalmente implicaría un menor tiempo de viaje. Por ejemplo, para el caso mencionado anteriormente, sería más razonable tomar la estrategia $(L1, L1, L2)$ que consta solamente de un cambio de línea, que tomar la formada por $(L1, L2, L3)$ que implica trasbordar en dos ocasiones. Para referirnos a las mejores estrategias, se denotará por π^w al conjunto de estrategias factibles y "razonables" correspondientes a todos los caminos que conectan el par OD $w = (w_o,\ w_d)$, evitando de esta forma en lo sucesivo, a efectos de la formulación del modelo, cualquier referencia a un camino específico. Cada elemento en π^w se denotará como π_j^w, que se corresponde con una serie de tripletas ordenadas $\pi_j^w = \{(w_o, i, l_1), \dots, (k, w_d, l_m)\}$. Anteriormente, cada camino p contenía la serie de arcos ordenados que llevaban de un nodo origen a uno destino, ahora, se ha añadido además la línea utilizada para atravesar cada arco, obteniendo de esta forma una sucesión de tripletas que identifican unívocamente cada estrategia. Si consideramos la primera estrategia mencionada para el caso anterior, quedaría ahora descrita de la siguiente forma: $\pi_1^w = \{\ (2, 3, L1), (3, 4, L1), (4, 5, L2)\ \}$, siendo $w = (2,\ 5)$.

A su vez, los conjuntos de líneas, arcos y nodos involucrados en una determinada estrategia π_j^w, se van a denotar por $L(\pi_j^w),\ A(\pi_j^w)\ y\ N(\pi_j^w)$ respectivamente. Finalmente, $T(\pi_j^w)$ será el conjunto de nodos de transferencia contenidos en la estrategia π_j^w. Para cálculos posteriores, será de utilidad definir también un operador $T_{\pi_j^w}(i)$, que aplicado a una determinada estación $i \in T(\pi_j^w)$, devuelve la pareja de líneas involucradas en la transferencia del nodo considerado.

Por otro lado, la probabilidad de escoger cada una de las estrategias "razonables" $\pi_j^w \in \pi^w$ se expresará mediante $p_{\pi_j^w}$, resultando de manera obvia $\sum_{j \in \pi^w} p_{\pi_j^w} = 1 \; \forall \, w \in W$, es decir, la suma de las probabilidades de todas las posibles estrategias para cada par OD w, tiene que ser igual a la unidad.

En la primera asignación de tránsito que se realice, los únicos datos conocidos de cada línea son los mencionados, por lo tanto, la probabilidad inicial va a depender de la longitud total que tenga el viaje siguiendo la estrategia π_j^w en comparación con el resto de las estrategias existentes para π^w. La longitud total se obtiene a partir de la suma de todos los arcos que componen el viaje, que se representará mediante $l(\pi_j^w)$. Imaginemos, por ejemplo, que tenemos una estrategia π_j^w, siendo $w = (1, 4)$ tal que $\pi_j^w = \{ (1, 2, l_1), (2, 3, l_2), (3, 4, l_3) \}$. Se conoce que la longitud del arco (1, 2) es de 1560 metros, la del arco (2, 3) es de 1830 metros y por último la del arco (3, 4) de 1750 metros, por lo tanto, tendríamos que $l(\pi_j^w)$ es igual a 5140 metros. Es importante señalar que este cálculo basado en longitudes solo se usará a la hora de calcular los valores iniciales de las probabilidades en un proceso iterativo para alcanzar el óptimo del problema. A partir de la segunda iteración, las probabilidades de elección de las estrategias se calcularán usando los tiempos de viaje asociados a las mismas.

Teniendo en cuenta todo lo anterior, la probabilidad de que un pasajero escoja la estrategia π_j^w viene dada por la siguiente ecuación:

$$p_{\pi_j^w} = \frac{\sum_{e \in \pi^w} l(\pi_e^w) - l(\pi_j^w)}{(|\pi_j^w| - 1) \cdot \sum_{e \in \pi^w} l(\pi_e^w)}$$

La cual implica que:

- La suma de las probabilidades de todas las estrategias existentes para cada par OD $w \in W$ tiene que ser igual a 1.

- Para un determinado par OD $w \in W$, la estrategia que tendrá una mayor probabilidad de ser escogida será aquella con menor longitud de viaje.

Sin embargo, en las siguientes asignaciones, ya se habrá resuelto el modelo de optimización y se dispondrán de todos los datos necesarios para poder calcular el tiempo total de viaje, por lo tanto, como se comentó anteriormente, el cálculo de la probabilidad se verá modificado realizándose en base al tiempo de viaje que suponga la estrategia π_j^w en comparación con el resto de las estrategias existentes para π^w, quedando finalmente expresada de la siguiente forma:

$$p_{\pi_j^w} = \frac{\sum_{e \in \pi^w} t(\pi_e^w) - t(\pi_j^w)}{(|\pi_j^w| - 1) \cdot \sum_{e \in \pi^w} t(\pi_e^w)}$$

Esta fórmula tiene las mismas implicaciones que las mencionadas anteriormente.

Con relación al tiempo de viaje $t(\pi_j^w)$ de una determinada estrategia π_j^w, este va a depender de:

- El tiempo esperando la llegada del primer tren, el cual, se considera la mitad del "headway" de la línea.

- La velocidad que deben tener los trenes en cada tramo (arco) involucrado en la estrategia, ya que como se mencionó anteriormente, los límites de velocidad varían de un tramo a otro por condiciones técnicas y de seguridad.

- El número de número de trasbordos a realizar, el cual dependerá de las líneas involucradas en la estrategia. Se considerará como tiempo de transferencia el valor correspondiente a la mitad del "headway" (intervalo de tiempo entre dos trenes consecutivos) de la línea a la que se transborda.

- El tiempo de parada (más conocido como "dwell time") de los trenes en las diferentes estaciones contenidas en la estrategia, el cual dependerá de la demanda de pasajeros, de la frecuencia de las líneas involucradas (a menor frecuencia de la línea, mayor será la cantidad de personas que bajen y embarquen en una determinada estación) y del tipo de tren (cuanto más largo sea el tren, mayor número de puertas tendrá y, como consecuencia, las subidas y bajadas se producirán en un tiempo menor).

Nota: Para una simplificación del problema, el modelo propuesto asume una capacidad de los vehículos lo suficientemente grande como para que cualquier persona que se encuentre esperando en un andén, pueda subirse al primer tren que llegue sin necesidad de tener que esperar al siguiente por falta de espacio, por lo tanto, el tiempo de espera que esta situación implicaría, no va a ser considerado.

Por otro lado, la demanda total de cada par OD, g_w, se va a dividir entre todas las posibles estrategias π^w, de tal forma que, el flujo que le corresponde a cada una de ellas vendrá dado por la siguiente expresión:

$$f_{\pi_j^w} = g_w \cdot p_{\pi_j^w}$$

Como era de esperar, aquellas estrategias con mayor probabilidad captarán un mayor número de pasajeros.

Se sobreentiende que ese flujo atravesará cada uno de los arcos que compongan dicha estrategia, por lo tanto, a partir de la expresión anterior, se puede obtener cual será la carga que soportará cada uno de los arcos $(i,\ j) \in A$, que componen las diferentes líneas de la red $\ell\ \in\ L$. Para cada arco, el flujo total de pasajeros durante el período de estudio (una hora en el caso de este proyecto), se obtiene a partir de la siguiente expresión:

$$f_{(i,j)}^{\ell} = \sum_{w \in W} \sum_{\pi_e^w \in \pi^w} f_{\pi_e^w} \cdot \theta_{ij}^{we\ell}$$

donde $\theta_{ij}^{we\ell}$ es un parámetro auxiliar que contiene información acerca de la relación existente entre el elemento (i,j,ℓ) y la estrategia e del par OD w, de tal forma que, dicho parámetro tomará valor 1 cuando el arco (i,j) perteneciente a la línea ℓ, esté contenido en la estrategia e del par OD w.

$$\theta_{ij}^{we\ell} = \begin{cases} 1 & si\ (i,j,\ell) \in\ \pi_e^w \\ 0 & en\ caso\ contrario \end{cases}$$

De forma similar, la cantidad de personas que trasbordan de la línea ℓ a la ℓ' en la estación $i\ \in\ \boldsymbol{T}(\pi_j^w)$, se puede obtener de la siguiente expresión:

$$f_i^{\ell\ell'} = \sum_{w \in W} \sum_{\pi_e^w \in \pi^w} f_{\pi_e^w} \cdot \varphi_i^{we\ell\ell'}$$

donde $\varphi_i^{we\ell\ell'}$ es un parámetro auxiliar que nos permite conocer si es posible trasbordar desde la línea ℓ a la ℓ' en la estación i, usando la estrategia π_e^w.

$$\varphi_i^{we\ell\ell'} = \begin{cases} 1 & si\ i\ \in\ T(\pi_e^w)\ y\ \ell\ ,\ell' \in\ T_{\pi_e^w}(i) \\ 0 & en\ caso\ contrario \end{cases}$$

A continuación, se muestra una tabla donde se resumen los parámetros y variables usados en relación con las estrategias de viaje.

$\boldsymbol{P^w}$	*Conjunto de caminos que conectan el par OD* $w \in W$
$\boldsymbol{p^w}$	*Elemento en* P^w
$\boldsymbol{\pi_p^w}$	*Conjunto de estrategias posibles para el camino* p^w
$\boldsymbol{\pi^w}$	*Conjunto de estrategias "razonables" para cualquier camino en* P^w
$\boldsymbol{\pi_j^w}$	*Estrategia concreta en* π^w
$\boldsymbol{L(\pi_j^w)}$	*Conjunto de líneas involucradas en la estrategia* π_j^w
$\boldsymbol{A(\pi_j^w)}$	*Conjunto de arcos involucrados en la estrategia* π_j^w
$\boldsymbol{N(\pi_j^w)}$	*Conjunto de nodos involucrados en la estrategia* π_j^w
$\boldsymbol{T(\pi_j^w)}$	*Conjunto de nodos de transferencia involucrados en la estrategia* π_j^w
$\boldsymbol{T_{\pi_j^w}(i)}$	*Operador que devuelve la pareja de líneas involucradas en la transferencia que se produce en la estrategia* π_j^w*, en la estación i*
$\boldsymbol{l(\pi_j^w)}$	*Longitud total del viaje de la estrategia* $\pi_j^w \in \pi^w$
$\boldsymbol{t(\pi_j^w)}$	*Tiempo total del viaje de la estrategia* $\pi_j^w \in \pi^w$
$\boldsymbol{p_{\pi_j^w}}$	*Probabilidad de seleccionar la estrategia* $\pi_j^w \in \pi^w$
$\boldsymbol{f_{\pi_j^w}}$	*Cantidad de personas que escogen la estrategia* $\pi_j^w \in \pi^w$
$\boldsymbol{f_{(i,j)}^{\ell}}$	*Cantidad de personas que circulan sobre el arco* $(i,j) \in A$ *usando la línea* $\ell \in L$
$\boldsymbol{f_i^{\ell\ell'}}$	*Cantidad de personas que trasbordan de la l línea* ℓ *a la* ℓ' *en la estación* i
$\boldsymbol{\theta_{ij}^{we\ell}}$	*Variable auxiliar que toma valor 1 cuando el arco* (i,j) *perteneciente a la línea* ℓ*, esté contenido en la estrategia* e *del par*
$\boldsymbol{\varphi_i^{we\ell\ell'}}$	*Variable auxiliar que toma valor 1 cuando es posible trasbordar de la línea* ℓ *a la* ℓ' *en la estación* i*, usando la estrategia* π_e^w

Tabla 3. Variables relativas a los caminos y estrategias.

3.3.4 Programación de los trenes

Tasas de subida y bajada

Con el fin de dimensionar correctamente los tiempos de parada de los trenes en las diferentes estaciones, será necesario calcular la cantidad de personas que suben y bajan en cada plataforma de cada línea $p \in P^{\ell}$. De esta forma, podremos dimensionar el número de trenes que debe tener cada línea, y al mismo tiempo, garantizar que los tiempos de parada (dwell time) son adecuados para permitir la subida y bajada de todos los pasajeros en cada plataforma de la red.

Se denota por S_p^ℓ, a la cantidad total de personas que suben en la plataforma $p \in P^\ell$ a lo largo del horizonte temporal considerado (recordemos que en este proyecto trabajamos con una planificación). Tal y como el conjunto P^ℓ ha sido definido, esto implica que:

$$S_p^\ell = 0 \ cuando \ p = |N(\ell)|$$
$$S_p^\ell = 0 \ cuando \ p = 2|N(\ell)|$$

Esto es debido a que no tiene sentido considerar que algún pasajero sube al tren en las dos estaciones finales (sentidos ida y vuelta) de la línea.

Para las demás plataformas de la línea, suponiendo que se ha realizado una asignación de pasajeros a la red, el valor S_p^ℓ estará compuesto por la suma de dos cantidades, $SI_p^\ell \ y \ ST_p^\ell$:

- Por un lado, tenemos a los usuarios que inician su viaje en la plataforma considerada, SI_p^ℓ. Para obtener dicho valor, primero tendremos que recorrer todos los pares OD de la red, y seleccionar aquellos que inician su viaje en la estación a la que corresponde la plataforma considerada, $s^\ell(p)$. Tras esto, para cada par OD, se recorren todas las posibles estrategias que llevan del origen al destino que se está analizando, $\boldsymbol{\pi^w}$. Para cada estrategia $\boldsymbol{\pi_e^w}$, se comprueba si alguno de los arcos $((s^\ell(p), j)$ perteneciente a la línea ℓ que estamos considerando, está contenido en ella (recordemos que esta relación nos la proporciona el parámetro $\boldsymbol{\theta_{ij}^{we\ell}}$). Cuando se cumplen las condiciones anteriores, se contabiliza el flujo de pasajeros correspondiente a la estrategia usada, $f_{\pi_e^w}$. Tras esto, se siguen comprobando las demás estrategias posibles para el par OD considerado, y cuando se finaliza se vuelve a realizar el mismo procedimiento con el siguiente par OD que tenga como origen la estación $s^\ell(p)$. De esta forma, se irán sumando los diferentes flujos de pasajeros que inician su viaje en la plataforma considerada, hasta obtener la cantidad final.

- Por otro lado, tendremos a los usuarios que también se suben al tren en dicha plataforma, debido a que transbordan en la estación $s^\ell(p)$ desde otras líneas de la red a la línea ℓ considerada. Para obtener dicho valor, representado por ST_p^ℓ, primero tenemos que recorrer todos los pares OD de la red, y seleccionar aquellos que inician y finalizan su viaje en una estación diferente a $s^\ell(p)$, ya que en estos dos casos nunca tendrá lugar el trasbordo buscado. Para un par OD concreto, se recorren todas las posibles estrategias que llevan del origen al destino que se está analizando, π^w. Para cada estrategia $\boldsymbol{\pi_e^w}$, se comprueba si siguiéndola es posible trasbordar en la estación $s^\ell(p)$ desde otras líneas de la red a la línea ℓ considerada (recordemos que esta relación nos la proporciona el parámetro $\varphi_i^{we\ell\ell'}$). Si se cumple esta condición, se contabiliza el flujo de pasajeros correspondiente a la estrategia usada, $f_{\pi_e^w}$. Tras esto, se siguen

comprobando las demás estrategias para el par OD considerado, y cuando se finaliza se vuelve a realizar el mismo procedimiento con el siguiente par OD que ni inicie ni finalice su viaje en la estación $s^{\ell}(p)$.

En la misma línea que lo anteriormente expuesto, se denota por B_p^{ℓ}, a la cantidad total de personas que se bajan en la plataforma $p \in P^{\ell}$ a lo largo del horizonte temporal considerado. Tal y como el conjunto P^{ℓ} ha sido definido, esto implica que:

$$B_p^{\ell} = 0 \ \ cuando \ p = 1$$
$$B_p^{\ell} = 0 \ \ cuando \ \ p = |N(\ell)| + 1$$

Esto es debido a que no tiene sentido considerar que algún pasajero baja del tren en las dos estaciones iniciales (sentidos ida y vuelta) de la línea.

Para las demás plataformas de la línea, suponiendo que se ha realizado una asignación de pasajeros a la red, el valor B_p^{ℓ} estará compuesto de la suma de dos cantidades, BF_p^{ℓ} y BT_p^{ℓ}:

- Por un lado, tenemos a los usuarios que finalizan su viaje en la plataforma considerada, BF_p^{ℓ}. Para obtener dicho valor, primero tenemos que recorrer todos los pares OD de la red, y seleccionar aquellos que finalicen su viaje en la estación a la que corresponde la plataforma considerada, $s^{\ell}(p)$. Tras esto, estando en un par OD concreto, se recorren todas las posibles estrategias que llevan del origen al destino que se está analizando, π^w. Para cada estrategia π_e^w, se comprueba si alguno de los arcos $(i, s^{\ell}(p))$ perteneciente a la línea ℓ que estamos considerando, está contenido en ella (recordemos que esta relación nos la proporcionaba el parámetro $\theta_{ij}^{we\ell}$). Cuando todo lo anterior se cumple, se contabiliza el flujo de pasajeros correspondiente a la estrategia usada, $f_{\pi_e^w}$. Tras esto, se siguen comprobando las demás estrategias para el par OD considerado, y cuando se finaliza se vuelve a realizar el mismo procedimiento con el siguiente par OD que tenga como destino la estación $s^{\ell}(p)$. De esta forma, se irán sumando los diferentes flujos de pasajeros que finalizan su viaje en la plataforma considerada, hasta obtener la cantidad final.

- Por otro lado, tendremos a los usuarios que también se bajan del tren en dicha plataforma, debido a que transbordan en la estación $s^{\ell}(p)$ desde la línea ℓ considerada hacia otras líneas de la red. Para obtener dicho valor, denotado mediante BT_p^{ℓ}, primero tenemos que recorrer todos los pares OD de la red, y quedarnos con aquellos inician y finalizan su viaje en una estación diferente a $s^{\ell}(p)$, ya que en estos dos casos nunca tendrá lugar el trasbordo buscado. Para cada par OD concreto, se recorren todas las

posibles estrategias que llevan del origen al destino que se está analizando, π^w. Para cada estrategia $\boldsymbol{\pi_e^w}$, se comprueba si siguiéndola es posible trasbordar en la estación $s^\ell(p)$ desde la línea ℓ considerada hacia otra línea de la red (recordemos que esta relación nos la proporciona el parámetro $\varphi_i^{we\ell\ell'}$). Si se cumple esta condición, se contabiliza el flujo de pasajeros correspondiente a la estrategia usada, $f_{\pi_e^w}$. Tras esto, se siguen comprobando las demás estrategias para el par OD considerado, y cuando se finaliza se vuelve a realizar el mismo procedimiento con el siguiente par OD que ni inicie ni finalice su viaje en la estación $s^\ell(p)$.

Nótese que la formulación matemática de todo lo expuesto en este apartado resulta muy compleja, por lo tanto, no será abordada de forma explícita en este momento, sino que será tratada directamente en el código que explica el procedimiento de cálculo.

Además, debido a que el horizonte temporal considerado en la matriz OD era de una hora (3600 s), se define la media de subidas y bajadas de pasajeros por segundo en cada plataforma como:

$$\widehat{S_p^\ell} = \frac{S_p^\ell}{3600}, \quad \widehat{B_p^\ell} = \frac{B_p^\ell}{3600}, \quad \forall\, p \in P^\ell$$

Tiempo de recorrido

La duración o el tiempo de recorrido de una línea $\ell \in L$ en una determinada vía $v \in V^\ell$, se va a expresar mediante TR_v^ℓ , entendiendo por esta el tiempo tarda un tren en ir desde una plataforma $p \in P^\ell$ hasta una plataforma $p+1 \in P^\ell$.

Debido a los límites de velocidad establecidos, tal duración vendrá dada por:

$$\frac{l_v^\ell}{V_v^{\ell,max}} \leq TR_v^\ell \leq \frac{l_v^\ell}{V_v^{\ell,min}}$$

donde recordemos, l_v^ℓ representa la longitud de la vía v.

Tiempo de parada

Por otro lado, el tiempo de parada o "dwell time" de los trenes en cada plataforma de la red, expresado como Tp_p^ℓ, debe ser lo suficientemente grande como para permitir la bajada de todos los que pasajeros que lo deseen y la subida de todos los que lleguen a esta durante el intervalo de tiempo que existe entre dos trenes consecutivos, el cual, como se vio anteriormente, se denomina "headway", h^ℓ. Esto implica, tener en cuenta para su formulación ciertas restricciones en relación con las tasas de subida y bajada empíricas (medidas en segundos/pasajero), las cuales vienen dadas por a^r y b^r respectivamente. Como se puede observar, dichas tasas dependerán a su vez del tipo de tren r seleccionado, ya que trenes más largos, tendrán mayor número de puertas y como consecuencia, las tasas serán mayores.

Por ello, primero es necesario definir un conjunto de variables binarias que nos permita conocer qué modelo de tren se ha escogido para cada línea.

$$\delta_r^\ell = \begin{cases} 1 & si\ modelo\ r\ se\ selecciona\ para\ \ell \\ 0 & en\ caso\ contrario \end{cases}$$

Además, se impone la condición de que solamente se escoja un modelo para cada línea:

$$\sum_{r \in \Omega^\ell} \delta_r^\ell = 1, \quad \forall\, \ell \in L$$

Tras esto, la restricción del tiempo de parada queda de la siguiente forma:

$$Tp_p^\ell \geq \left(\sum_{r \in \Omega^\ell} \delta_r^\ell \cdot a^r \cdot \widehat{S_p^\ell} + \sum_{r \in \Omega^\ell} \delta_r^\ell \cdot b^r \cdot \widehat{B_p^\ell} \right) \cdot h^\ell, \qquad \forall\, p \in P^\ell, \ell \in L$$

"Headways" y frecuencias

La principal contribución de este trabajo consistía en la determinación de la frecuencia que deben tener las diferentes líneas de la red, la cual viene dada por el inverso del "headway", por lo tanto, el diseño de dicha variable será fundamental.

Normalmente, los valores del "headway" suelen verse limitados tanto por arriba como por abajo, de forma que se establece un mínimo "headway" entre dos trenes consecutivos

por requerimientos del sistema de señalización, y, a su vez, también un valor máximo para garantizar la calidad del servicio ofertado.

$$h_{min}^{\ell} \leq h^{\ell} \leq h_{max}^{\ell}, \qquad \forall \ell \in L$$

Al mismo tiempo, se suelen buscar horarios repetitivos, lo que implica que el "headway" debe ser divisor de 3600. Para conseguirlo, se introduce un conjunto de valores enteros admisibles para el "headway", denotado por $\mathcal{H}$. Habitualmente, se imponen como valores mínimos y máximos 2 y 30 minutos respectivamente, por lo tanto, el conjunto de posibles valores (medidos en segundos) será:

$$\mathcal{H} = \{120, 180, 240, 300, 360, 600, 720, 900, 1200, 1800\}.$$

Como la frecuencia de la línea (número de trenes por hora) se relaciona con el "headway" mediante $f^{\ell} \cdot h^{\ell} = 3600$, $\forall \ell \in L$, tendremos que el posible conjunto de valores admisibles para la frecuencia viene dado por $\mathcal{F}^j = 3600 / \mathcal{H}^j$, $j = 1 \ldots |\mathcal{H}|$.

Además, para que cada línea tome uno y sólo uno de los posibles valores admisibles definidos anteriormente, será necesario definir una serie de variables binarias λ_j^{ℓ} y establecer un conjunto de restricciones:

$$\lambda_j^{\ell} = \begin{cases} 1 & si\ \mathcal{H}^j\ se\ selecciona\ para\ la\ línea\ \ell \\ 0 & en\ caso\ contrario \end{cases}$$

$$h^{\ell} = \sum_{j \in \mathcal{H}} \mathcal{H}^j \cdot \lambda_j^{\ell}, \quad \forall \ell \in L$$

$$f^{\ell} = \sum_{j \in \mathcal{H}} \mathcal{F}^j \cdot \lambda_j^{\ell}, \quad \forall \ell \in L$$

$$\sum_{j \in \mathcal{H}} \lambda_j^{\ell} = 1, \quad \forall \ell \in L$$

Finalmente, se establece una relación obvia entre el "headway" y el "dwell time". El intervalo de tiempo entre dos trenes consecutivos de una misma línea siempre tendrá que ser mayor o igual que los tiempos de espera establecidos para cada una de las plataformas de la línea, más un tiempo de seguridad establecido.

$$Tp_p^{\ell} + Tseg \leq h^{\ell}, \quad \forall p \in P^{\ell}, \ell \in L$$

Tiempo de ciclo y tamaño de la flota

Una vez obtenido el "headway", se puede determinar también, otro de los principales objetivos de este trabajo, el tamaño que debería tener la flota de trenes, F^{ℓ}. Para ello, primero hay que calcular el tiempo de ciclo de cada línea (tiempo necesario para que un tren complete todo el recorrido en los dos sentidos, ida y vuelta). Este se obtiene mediante la siguiente expresión:

$$C^{\ell} = \sum_{v \in V^{\ell}} TR_v^{\ell} + \sum_{p \in P^{\ell}} Tp_p^{\ell} + 2 \cdot TB^{\ell}, \quad \forall\, \ell \in L$$

donde TB^{ℓ} representa el tiempo necesario para realizar el cambio de sentido en cada una de las dos plataformas finales.

A partir de aquí, el tamaño de la flota de trenes sería el siguiente:

$$TF^{\ell} = \frac{C^{\ell}}{h^{\ell}}, \quad \forall\, \ell \in L$$

Por ejemplo, si consideramos una determinada línea con un tiempo de ciclo de 60 min, para que pudiese haber un tiempo entre cada tren consecutivo de 12 min, la línea debería tener una flota de 5 trenes.

Capacidad de los trenes

Teniendo en cuenta que cada modelo de tren $r \in \Omega^{\ell}$ (Ω^{ℓ} representa el conjunto de modelos factibles para la línea ℓ) tiene una capacidad diferente denotada por c_r, se deberá establecer una relación que nos permita asegurar que toda la demanda es atendida. De tal forma que, el número de trenes que se tengan en una hora (frecuencia de la línea) por la capacidad del modelo seleccionado, siempre tendrá que superar el valor del tramo más cargado de la correspondiente línea, y así nos aseguraremos de que todos los pasajeros tienen cabida en el tren.

Recordemos que anteriormente definimos como $f_{(i,j)}^{\boldsymbol{\ell}}$, a la carga de pasajeros que soporta un determinado arco (i,j) perteneciente a la línea ℓ durante el periodo de tiempo considerado (una hora). A través del operador conveniente, obtenemos la vía correspondiente a ese arco, $v = v^{\ell}(i,j)$, y pasamos a denotar por Q_v^{ℓ} a la carga que soporta cada vía de la línea ℓ. De esta forma, Q_{max}^{ℓ} se corresponderá con el máximo valor que tome dicha variable dentro de la línea considerada.

$$Q_{max}^{\ell} \leq \sum_{r \in \Omega^{\ell}} \delta_r^{\ell} \cdot c_r \cdot f^{\ell}, \quad \forall\, \ell \in L$$

A continuación, se muestra una tabla donde se resumen las variables y parámetros usados en relación con la programación de las líneas.

S_p^{ℓ}	*Número de pasajeros que suben en la plataforma p de la línea ℓ*
SI_p^{ℓ}	*Número de pasajeros que suben en la plataforma p de la línea ℓ debido a que inician su viaje en ella*
ST_p^{ℓ}	*Número de pasajeros que suben en la plataforma p de la línea ℓ debido a que trasbordan desde otra línea*
B_p^{ℓ}	*Número de pasajeros que bajan en la plataforma p de la línea ℓ*
BF_p^{ℓ}	*Número de pasajeros que bajan en la plataforma p de la línea ℓ debido a que finalizan su viaje en ella*
BT_p^{ℓ}	*Número de pasajeros que bajan en la plataforma p de la línea ℓ debido a que van a trasbordar hacia otra línea*
TR_v^{ℓ}	*Duración del recorrido de la línea ℓ en la vía v*
$a^{(r)}$	*Tasa de subida (pasajeros/segundo) empírica para el modelo r*
$b^{(r)}$	*Tasa de bajada (pasajeros/segundo) empírica para el modelo r*
Tp_p^{ℓ}	*Tiempo de parada de un tren de la línea ℓ en la plataforma p*
h^{ℓ}	*Intervalo de tiempo entre dos trenes consecutivos de la línea ℓ*
C^{ℓ}	*Tiempo de ciclo de un tren de la línea ℓ*
f^{ℓ}	*Frecuencia de la línea ℓ*
TB^{ℓ}	*Tiempo necesario para el cambio de sentido de la línea ℓ en las estaciones finales*
TF^{ℓ}	*Tamaño de la flota de la línea ℓ*
c_r	*Capacidad del modelo de tren r*
Ω^{ℓ}	*Conjunto de trenes factibles para la línea ℓ*
δ_r^{ℓ}	*Variable binaria que toma valor 1 si el modelo r se escoge para la línea ℓ*

Tabla 4. Variables y parámetros para la programación de las líneas.

3.3.5 Función objetivo

Tal y como se ha expuesto anteriormente, el objetivo del modelo consiste en determinar los valores óptimos de la frecuencia del servicio y del resto de variables necesarias para poder programar las distintas líneas que componen una red de cercanías, todo ello con el propósito de obtener una solución que favorezca tanto a los operadores,

como a los pasajeros. No obstante, sin ningún tipo de información previa sobre las preferencias del decisor o modelador, debido a que estamos considerando dos objetivos diferentes y que, además, estos son contrapuestos, no se podrá afirmar que una solución es mejor que otra, a menos que la domine. Aquí, entra en juego el concepto de Optimalidad de Pareto.

La solución a un problema de optimización multiobjetivo (dos objetivos en este caso), es el conjunto de soluciones no dominadas (a las que se denomina conjunto de Pareto) que cumplen la propiedad de Optimalidad de Pareto, la cual, viene a decirnos los siguiente: "Una solución es óptimo de Pareto cuando no hay forma de mejorar un criterio u objetivo, sin empeorar otro". Es decir, en nuestro caso concreto, una solución sería óptimo de Pareto, si la única forma de mejorar el objetivo de los operadores fuese a costa de empeorar el de los pasajeros, y viceversa.

A su vez, a partir del valor que toman las distintas funciones objetivo para dichas soluciones óptimas, se obtendrá la Frontera de Pareto, la cual, representa la separación en el espacio entre las soluciones factibles y no factibles.

En la siguiente imagen, podemos observar cómo se representaría gráficamente la Frontera de Pareto para el caso en el que se quieran minimizar dos objetivos, tal y como ocurre en este proyecto. Todas las soluciones que se encuentren por encima de dicha frontera se denominan soluciones dominadas, debido a que siempre habrá alguna solución mejor que ellas, es decir, que las domine. Por el contrario, todas las soluciones que quedan por debajo de la frontera no son factibles, por lo tanto, jamás se podrán obtener.

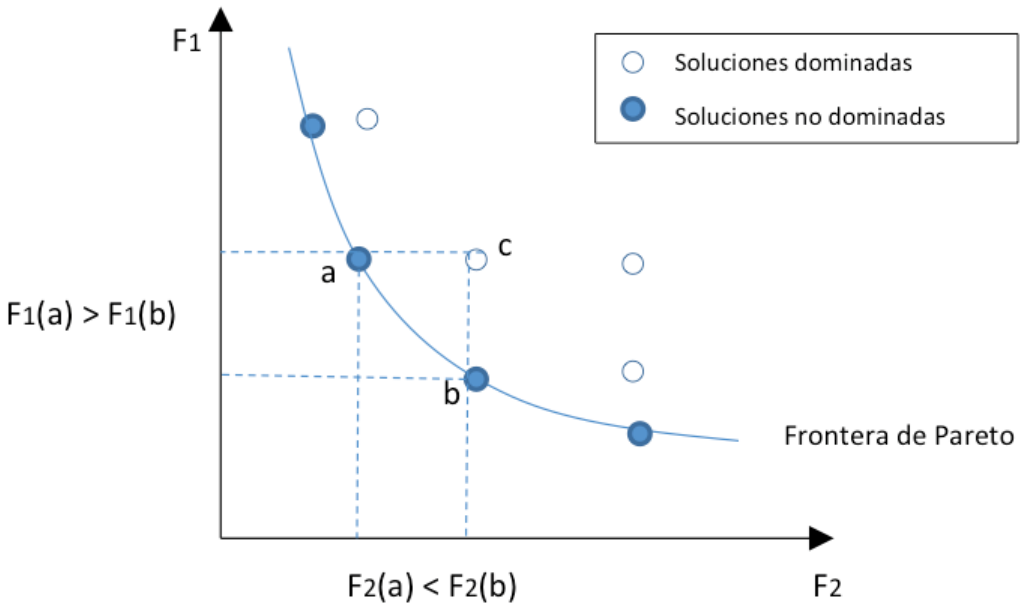


Figura 9. Frontera de Pareto para un problema de minimización (Aranda & Orjuela 2015)

A la hora de posteriormente llevar el modelo a la práctica, se buscará obtener ese conjunto de soluciones que cumplen la optimalidad de Pareto, y a partir de ahí, seleccionaremos de entre las soluciones obtenidas, aquella que en la mayor medida posible satisfaga tanto a operadores como a pasajeros.

Para poder muestrear la frontera, se va a establecer una función objetivo ponderada, de tal forma que, cada uno de los objetivos va a tener un determinado peso en el objetivo global. A su vez, para lograr obtener diferentes puntos de la Frontera de Pareto, se va a llevar a cabo la resolución de varios escenarios, a lo largo de los cuales, se les irán dando distintos pesos a los dos objetivos, y por consecuencia, los resultados obtenidos variarán.

Ahora ya sí, pasamos a explicar la función objetivo que se va a utilizar para tener en cuenta ambos criterios (operadores y pasajeros).

Por un lado, desde el punto de vista del operador, el objetivo será conseguir tanto dinero como sea posible, por lo que para ello, tendrá que disminuir sus costes (Guihaire & Hao 2008). Habitualmente, se consideran dos tipos de costes asociados al operador:

- Coste variable de operación, en función de la distancia recorrida por los trenes de cada una de las líneas que componen la red (medido en €/tren·km). Este coste dependerá de varias partidas, como son: la energía consumida, la amortización y depreciación, el mantenimiento de los trenes, los pagos a ADIF por infraestructura y estaciones, los intereses y gastos financieros, y otros materiales y servicios. A su vez, este coste varía en función del modelo seleccionado.

$$\sum_{\ell \in L} \sum_{r \in \Omega^{\ell}} \theta_1^r \cdot \delta_r^{\ell} \cdot f^{\ell} \cdot Long_{\ell}$$

donde $Long_{\ell} = \sum_{v \in V^{\ell}} \frac{2 \cdot l_v^{\ell}}{1000}$, $\ell \in L$

- Coste asociado al tiempo de operación de los trenes (medido en €/tren·hora), el cual representa los gastos de tripulación. Estos variarán en función del tamaño de la flota de trenes que se establezca para cada línea.

$$\sum_{\ell \in L} \theta_2 \cdot TF^{\ell}$$

Por otro lado, desde la perspectiva de los usuarios, el sistema deberá ofrecer una calidad del servicio adecuada. Esta suele medirse en términos del tiempo total de viaje, el cual, puede ser dividido a su vez en tiempo de espera en la primera estación, tiempo de viaje a bordo del tren, y finalmente, tiempo de transferencia. En muchas ocasiones, también suele añadirse el número de transferencias que han tenido que realizar los pasajeros para llegar a su destino. Para que todos estos factores puedan ser incorporados a la función objetivo junto con los anteriormente descritos, se realiza una conversión a valores monetarios. Para un mejor entendimiento de tal conversión, en (Wardman 2004) se puede encontrar un análisis del valor del tiempo de viaje aplicado a los sistemas de transporte público.

Si nos basamos en el trabajo de (Robenek et al. 2016), el coste que les supone a todos los pasajeros considerados en la matriz de demanda utilizar el servicio ferroviario, viene dado por la expresión que se muestra a continuación:

$$VOT \cdot \left(\beta_1 \sum_{\ell \in L} \sum_{p \in P^\ell} S_p^\ell \cdot \frac{h^\ell}{2 \cdot 3600} + \beta_2 \sum_{\ell \in L} \sum_{p \in P^\ell} ST_p^\ell + \beta_3 \sum_{\ell \in L} \sum_{v \in V^\ell} Q_v^\ell \cdot \frac{TR_v^\ell}{3600} \right)$$

- El primero término se corresponde con el tiempo total de espera que experimentan los pasajeros en las diferentes estaciones de la red, englobando tanto aquellas estaciones en las que inician su viaje, como aquellas en las que esperan para trasbordar hacia otras líneas de la red. El tiempo que emplean esperando la llegada del tren, se asume que es la mitad del "headway" de la línea a la que pertenece la estación en la que se encuentran, por ello, se multiplicará este valor por el total de subidas que se producen en la red.

- El segundo término hace referencia al total de transferencias que tienen lugar en la red. Se puede ver fácilmente como dicho valor equivale al total de subidas (o bajadas) por transferencias que se producen en todas las plataformas. Sin embargo, dicho término será un dato de entrada que el modelo de optimización tome de la fase de asignación de tránsito y, por ello, no habrá posibilidad de minimizarlo. A pesar de ello, será tenido en cuenta para poder conocer la verdadera proporción (escala) que suponen los costes de los pasajeros en el objetivo global.

- El tercer término se corresponde con el tiempo de viaje a bordo del tren de todos los pasajeros, el cual se obtiene a través de sumar cada uno de los tiempos que el tren tarda en recorrer una vía, multiplicado por la carga de pasajeros que esta soporta.

Por otro lado, los distintos parámetros β se corresponden con el peso que se le va a dar dentro del propio criterio de pasajeros, a cada uno de los aspectos considerados, ya que, por ejemplo, un pasajero no otorga la misma importancia al tiempo que gasta dentro del tren, que al que tiene que estar fuera de este esperando su llegada.

Finalmente, una vez se tiene la toda función ponderada y en unidades de tiempo, para poder obtener su transformación a unidades monetarias, va a ser multiplicada por el término "VOT" (Value of Time), el cual, simboliza el precio que un viajero estaría dispuesto a pagar por ahorrarse una hora (o la unidad de tiempo considerada) de viaje.

Los valores numéricos de cada uno de los parámetros que han sido mencionados en este apartado serán explicados en profundidad más adelante cuando se realice la aplicación práctica del modelo.

A continuación, se muestra una tabla donde se resumen los parámetros usados (no definidos con anterioridad) en relación con la función objetivo.

$\boldsymbol{\theta_1^r}$	*Coste variable de operación del modelo de tren r (€/tren·km)*
$\boldsymbol{\theta_2}$	*Costes de la tripulación a cargo del tren (€/tren·hora)*
VOT	*Valor del tiempo (€/min)*
$\boldsymbol{\beta_1}$	*Peso del tiempo de espera en las estaciones (€/min)*
$\boldsymbol{\beta_2}$	*Peso del número de transferencias (€/min)*
$\boldsymbol{\beta_3}$	*Peso del tiempo a bordo del tren (€/min)*

Tabla 5. Parámetros asociados a la función objetivo.

NOTA: Todos los costes que han sido desarrollados están calculados para el horizonte temporal considerado en la matriz OD, que resulta ser de una hora.

3.4 Tratamiento de las no-linealidades

Antes de comenzar, es necesario dejar claros varios conceptos. Con el término de función lineal, se hace referencia a una función polinómica de primer grado, es decir, una función cuya representación en el plano cartesiano es una recta. Por otro lado, una función no-lineal es aquella que resulta del producto de dos o más variables, dando lugar a una función polinómica de grado mayor que la unidad. Sin embargo, algunas expresiones no lineales pueden transformarse a otras que sí lo son, para ello se usa la linealización. Linealizar consiste en encontrar una función lineal que pueda aproximar una función dada alrededor de un punto.

Se pretende conseguir que el modelo propuesto con anterioridad sea de programación lineal, ya que estos resultan mucho más sencillos y rápidos de resolver que los no lineales, no obstante, esto implica que tanto la función objetivo como todas las restricciones involucradas, deben ser lineales. Por lo tanto, para poder resolver el modelo anterior, será necesario linealizar aquellas expresiones no-lineales. Concretamente, un término de la función objetivo y tres restricciones que contienen un producto de variables desconocidas:

(1) $TF^{\ell} = \dfrac{C^{\ell}}{h^{\ell}}$

(2) $Q^{\ell}_{max} \leq \sum_{r \in \Omega^{\ell}} \delta^{\ell}_{r} \cdot c_r \cdot f^{\ell}$

(3) $Tp^{\ell}_{p} \geq \left(\sum_{r \in \Omega^{\ell}} \delta^{\ell}_{r} \cdot a^{r} \cdot \widehat{S^{\ell}_{p}} + \sum_{r \in \Omega^{\ell}} \delta^{\ell}_{r} \cdot b^{r} \cdot \widehat{B^{\ell}_{p}} \right) \cdot h^{\ell}$

(4) $\sum_{\ell \in L} \sum_{r \in \Omega^{\ell}} \theta^{r}_{1} \cdot \delta^{\ell}_{r} \cdot f^{\ell} \cdot Long_{\ell}$

A continuación, se explica la metodología seguida para la transformación de la primera expresión:

1. Se sustituye h^{ℓ} por la relación de selección de “headway” establecida anteriormente, y se pasa al otro lado de la ecuación multiplicando, quedando la ecuación de la siguiente forma:

$$TF^{\ell} \cdot \sum_{j \in \mathcal{H}} \mathcal{H}^{j} \cdot \lambda^{\ell}_{j} = C^{\ell}$$

2. En la expresión anterior, el producto $TF^{\ell} \cdot \lambda_j^{\ell}$ sigue siendo no-lineal, por lo tanto, va a ser sustituido por una nueva variable τ_j^{ℓ}.

$$TF^{\ell} \cdot \lambda_j^{\ell} = \tau_j^{\ell}$$

3. Para seguir cumpliendo la ecuación original, la nueva variable deberá tomar el valor TF^{ℓ} cuando λ_j^{ℓ} sea 1, y el valor 0 cuando λ_j^{ℓ} sea 0. Esto se consigue incorporando las restricciones que se muestran a continuación, donde M es un límite superior que equivaldría a la mayor flota de trenes posible, un valor razonable superior a ese sería imposible de alcanzar.

$$\tau_j^{\ell} \leq M \cdot \lambda_j^{\ell}$$

$$TF^{\ell} - \mathrm{M} \cdot (1 - \lambda_j^{\ell}) \leq \tau_j^{\ell} \leq TF^{\ell}$$

De hecho, se puede apreciar como cuando $\lambda_j^{\ell} = 0$, la primera ecuación obliga a τ_j^{ℓ} a tomar el valor 0, sin embargo, cuando $\lambda_j^{\ell} = 1$, la primera ecuación le da libertad para tomar cualquier valor menor o igual a M, mientras que la segunda, le fuerza a valer TF^{ℓ}.

4. Se puede obtener un valor de M realizando el siguiente razonamiento. Si el tamaño de flota viene dado por $TF^{\ell} = C^{\ell}/h^{\ell}$, el mayor valor posible se obtendría cuando el tiempo de ciclo de la línea fuese máximo y el "headway" mínimo.

De esta manera, la ecuación original (1) quedaría bien aproximada a una expresión lineal si se sustituye por las restricciones y variables formuladas en (números).

En la inecuación (2), la no-linealidad viene dada por el producto $\delta_r^{\ell} \cdot f^{\ell}$, por lo que se puede seguir el mismo procedimiento anterior, de tal forma que quedaría sustituida por las siguientes restricciones:

$$\begin{cases} Q_{max}^{\ell} \leq \sum_{r \in \Omega^{\ell}} c_r \cdot \alpha_r^{\ell} \\ \alpha_r^{\ell} \leq A \cdot \delta_r^{\ell} \\ f^{\ell} - \mathrm{A} \cdot (1 - \delta_r^{\ell}) \leq \alpha_r^{\ell} \\ \alpha_r^{\ell} \leq f^{\ell} \end{cases}$$

donde A, es un límite superior que representa el máximo valor posible para f^{ℓ}. Este se ha establecido anteriormente en 30 trenes/hora, por lo tanto, A=30.

Para la inecuación (3), la no-linealidad viene dada por el producto $\delta_r^\ell \cdot h^\ell$. Se procede de igual forma que en los dos casos anteriores:

$$\begin{cases} Tp_p^\ell \geq \sum_{r \in \Omega^\ell} \omega_r^\ell \cdot a^r \cdot \widehat{S_p^\ell} \; + \sum_{r \in \Omega^\ell} \omega_r^\ell \cdot \; b^r \cdot \widehat{B_p^\ell} \\ \omega_r^\ell \leq T \cdot \delta_r^\ell \\ h^\ell - \mathrm{T} \cdot (1 - \delta_r^\ell) \leq \omega_r^\ell \\ \omega_r^\ell \leq h^\ell \end{cases}$$

donde T, es un límite superior que representa el máximo valor posible para h^ℓ. Este se ha establecido anteriormente en 1800 segundos, por lo tanto, T=1800

Finalmente, procedemos a linealizar el término (4) perteneciente a la función objetivo. Se puede observar como la no-linealidad viene dada por el producto de las mismas dos variables ya tratadas en la segunda expresión $\delta_r^\ell \cdot f^\ell$, por lo tanto, basta con sustituir la expresión (4) por la siguiente, ya que las restricciones necesarias ya han sido incluidas en el modelo:

$$\sum_{\ell \in L} \sum_{r \in \Omega^\ell} \theta_1^r \; \cdot \; \alpha_r^\ell \cdot Long_\ell$$

3.5 Formulación matemática del modelo

Finalmente, tras la linealización de las expresiones requeridas, se obtiene el siguiente modelo de programación lineal, cuyos resultados dependerán a su vez, de una previa asignación de tránsito.

$$Min \quad \sum_{\ell \in L} \sum_{r \in \Omega^\ell} \theta_1^r \, \alpha_r^\ell \, Long_\ell \; + \sum_{\ell \in L} \theta_2 \, TF^\ell$$

$$+ VOT \; (\beta_1 \sum_{\ell \in L} \sum_{p \in P^\ell} S_p^\ell \cdot \frac{h^\ell}{2 \cdot 3600} \; + \; \beta_2 \sum_{\ell \in L} \sum_{p \in P^\ell} ST_p^\ell \; + \; \beta_3 \sum_{\ell \in L} \sum_{v \in V^\ell} Q_v^\ell \cdot \frac{TR_v^\ell}{3600} \;)$$

sa:

$$l_v^\ell / V_v^{\ell,max} \; \leq \; TR_v^\ell \; \leq l_v^\ell / \, V_v^{\ell,min} \, , \qquad \forall \, \ell \in \, L \, , \; v \, \in \, V^\ell$$

$$Tp_p^\ell \geq \sum_{r \in \Omega^\ell} \omega_r^\ell \, a^r \widehat{S_p^\ell} \; + \sum_{r \in \Omega^\ell} \omega_r^\ell \, b^r \widehat{B_p^\ell} \, , \qquad \forall \, p \, \in \, P^\ell , \ell \, \in \, L$$

$$h_{min}^\ell \leq h^\ell \leq h_{max}^\ell \, , \qquad \forall \, \ell \in \, L$$

$$Tp_p^\ell + Tseg \leq h^\ell , \qquad \forall \, p \, \in \, P^\ell , \ell \, \in \, L$$

$$C^\ell = \sum_{v \in V^\ell} TR_v^\ell \; + \sum_{p \in P^\ell} Tp_p^\ell + 2 \, TB^\ell , \qquad \forall \, \ell \in \, L$$

$$\sum_{r \in \Omega^\ell} \delta_r^\ell = 1 \, , \qquad \forall \, \ell \in \, L$$

$$\omega_r^\ell \leq T \delta_r^\ell \, , \qquad \forall \, \ell \in \, L \, , r \, \in \Omega^\ell$$

$$h^\ell - \mathrm{T}(1 - \delta_r^\ell) \leq \omega_r^\ell \, , \qquad \forall \, \ell \in \, L \, , r \, \in \Omega^\ell$$

$$\omega_r^\ell \leq h^\ell , \qquad \forall \, \ell \in \, L \, , r \, \in \Omega^\ell$$

$$h^\ell = \sum_{j \in \mathcal{H}} \mathcal{H}^j \, \lambda_j^\ell \, , \qquad \forall \, \ell \in \, L$$

$$f^\ell = \sum_{j \in \mathcal{H}} \mathcal{F}^j \lambda_j^\ell \, , \qquad \forall \, \ell \in \, L$$

$$\sum_{j \in \mathcal{H}} \lambda_j^\ell = 1\,, \qquad \forall\, \ell \in L$$

$$\sum_{j \in \mathcal{H}} \mathcal{H}^j \tau_j^\ell = C^\ell, \qquad \forall\, \ell \in L$$

$$\tau_j^\ell \leq M\lambda_j^\ell\,, \qquad \forall\, \ell \in L,\ j = 1\ldots|\mathcal{H}|$$

$$TF^\ell - \mathrm{M}(1 - \lambda_j^\ell) \leq \tau_j^\ell\,, \qquad \forall\, \ell \in L,\ j = 1\ldots|\mathcal{H}|$$

$$\tau_j^\ell \leq TF^\ell, \qquad \forall\, \ell \in L,\ j = 1\ldots|\mathcal{H}|$$

$$Q_{max}^\ell \leq \sum_{r \in \Omega^\ell} c_r\, \alpha_r^\ell\,, \qquad \forall\, \ell \in L$$

$$\alpha_r^\ell \leq A\delta_r^\ell\,, \qquad \forall\, \ell \in L,\ r \in \Omega^\ell$$

$$f^\ell - \mathrm{A}(1 - \delta_r^\ell) \leq \alpha_r^\ell\,, \qquad \forall\, \ell \in L,\ r \in \Omega^\ell$$

$$\alpha_r^\ell \leq f^\ell, \qquad \forall\, \ell \in L,\ r \in \Omega^\ell$$

$$\alpha_r^\ell,\ \omega_r^\ell \geq 0 \qquad \forall\, \ell \in L,\ r \in \Omega^\ell$$

$$\tau_j^\ell \geq 0 \qquad \forall\, \ell \in L,\ j = 1\ldots|\mathcal{H}|$$

$$\delta_r^\ell = \{0,1\} \qquad \forall\, \ell \in L,\ r \in \Omega^\ell$$

$$\lambda_j^\ell = \{0,1\} \qquad \forall\, \ell \in L,\ j = 1\ldots|\mathcal{H}|$$

$$TF^\ell, C^\ell \geq 0 \qquad \forall\, \ell \in L$$

$$h^\ell \in \mathcal{H} \qquad \forall\, \ell \in L$$

$$f^\ell \in \mathcal{H} \qquad \forall\, \ell \in L$$

$$Tp_p^\ell \geq 0 \qquad \forall\, p \in P^\ell, \ell \in L$$

$$TR_v^\ell \geq 0 \qquad \forall\, \ell \in L\,,\ v \in V^\ell$$

3.6 Procedimiento de resolución

Para la resolución del problema propuesto se va a aplicar un procedimiento iterativo, el cual consta de dos fases: primero se realizará la asignación de tránsito, y, posteriormente, se procederá a la optimización del modelo matemático propuesto. Como se mencionó con anterioridad, ambos problemas están completamente relacionados, cada uno influenciando al otro, de tal forma que los resultados obtenidos en uno pasarán a ser datos de entrada del otro, por ello, se realizarán sucesivas iteraciones hasta que se consiga alcanzar la convergencia, es decir, hasta que en dos iteraciones sucesivas los resultados sean prácticamente idénticos.

El procedimiento en su totalidad constará de los siguientes pasos:

1. Se calculan los K (≤ 3) caminos más cortos (en forma de lista de nodos) para cada par OD de la red usando las longitudes de los arcos que los componen.

2. Se procede a “expandir” los caminos, de tal forma, que ahora los caminos vienen dados en forma de (arco, línea). Así se obtienen las diferentes estrategias que los pasajeros pueden seguir para realizar un determinado camino, las cuales, irán en función de la combinación de líneas que los pasajeros escojan.

3. Se eliminan las estrategias “dominadas” mediante dos procedimientos. El primero de ellos, para cada par OD, contabiliza el número de trasbordos que hay que realizar para cada una de las posibles estrategias, guarda el menor valor, y mantiene solamente aquellas estrategias con dicho valor mínimo. Posteriormente se realiza un segundo procedimiento, en el cual, para cada par OD, se contabiliza la longitud de viaje de cada una de las estrategias restantes y guarda el menor valor. Tras esto, procede a eliminar todas aquellas estrategias del par, cuya longitud de viaje supere a la mínima en un 10%.

4. Se calcula la probabilidad de escoger cada una de las estrategias mediante las fórmulas expuestas anteriormente. Recordemos que, en la primera iteración, las probabilidades van en función de las longitudes totales de los caminos, ya que se desconocen las frecuencias de paso de cada línea y las velocidades de cada tren, sin embargo, en las siguientes iteraciones las probabilidades dependerán del tiempo total de viaje.

5. Una vez obtenida la probabilidad y conociendo la demanda de cada par OD, se calcula la cantidad de personas que circula por cada camino, y como consecuencia, por cada vía de la red. Al mismo tiempo, para cada línea de la red, se almacena el valor de aquella vía que soporta mayor número de pasajeros, ya que el modelo que se seleccione posteriormente tendrá que tener capacidad suficiente para albergarlos a todos.

6. Se calcula la cantidad de personas que suben y bajan en cada plataforma de la red, separando los dos sentidos de ida y vuelta.

7. A continuación, se procede a realizar la fase de optimización del modelo propuesto. Mediante un bucle principal, se calculan los siguientes valores para cada línea de la red: la frecuencia, el tamaño de la flota, los tiempos de parada en cada estación, los tiempos de recorrido de cada vía y se selecciona un modelo de tren para cada línea.

8. Se vuelve al punto 3, pero esta vez teniendo en cuenta los nuevos valores obtenidos para la frecuencia y los tiempos de recorrido en cada vía, ya que estos provocarán cambios en los tiempos de viaje de cada estrategia, y como consecuencia, todo lo calculado a partir de ellos también, lo que dará lugar a una asignación diferente.

9. Se evalúa el criterio global de parada. Si este se cumple, nos quedamos con la solución encontrada. En caso contrario, se vuelve a resolver el modelo de optimización con la nueva asignación obtenida.

El criterio de parada actuará en el siguiente caso:

Al final de cada asignación, se comprueba la diferencia entre la cantidad de personas que soporta cada vía de la red en la iteración actual y en la anterior, de tal forma, que cuando en dos iteraciones sucesivas todas las diferencias entre las cargas de cada vía sean nulas, se habrá alcanzado la convergencia, y podremos dar por finalizado el proceso.

4. RESOLUCIÓN MEDIANTE PYTHON

4.1 Introducción al software

El desarrollo y resolución de los problemas planteados ha sido realizado mediante el lenguaje de programación Python, a través de su distribución gratuita Anaconda. Para instalarla, solo ha sido necesario descargar la última versión disponible para el sistema operativo deseado desde su página web https://www.anaconda.com/download/**.** Concretamente, se ha trabajado con Python 3.6 para Windows (64-Bits).

A continuación, se procede a realizar una breve explicación sobre este lenguaje, exponiendo algunas de las características que han llevado a su utilización.

Python es un lenguaje de programación moderno, de alto nivel, interpretado y multipropósito, creado por un investigador holandés, llamado Guido van Rossum, a inicios de los años noventa.

Este presenta una serie de características que lo hacen muy atractivo, tanto para su uso profesional como para el aprendizaje de la programación. Entre las más interesantes destacan (Montoro 2012):

- Es interpretado, lo que significa que no es necesario compilar el código para su ejecución, ya que existe un intérprete que se encarga de leer el fichero fuente y ejecutarlo.

- Es multiplataforma, es decir, puede ser utilizado en diversas plataformas sin necesidad de cambiar el código fuente, entre las que destacan Windows, Mac OS X y Linux.

- Cabe destacar también, la simplicidad de su sintaxis y semántica, lo que facilita el aprendizaje del lenguaje y resulta en un código sencillo de leer.

- Es un lenguaje orientado a objetos y por ello dispone de las características más importantes de este paradigma, tales como clases y objetos, encapsulamiento de objetos y polimorfismo; aunque también soporta otros estilos de programación, como la programación funcional y la imperativa.

- Incluye una poderosa y extensa librería que incorpora multitud de funciones para diversos ámbitos.

- Contiene un recolector de basura (a diferencia de lenguajes compilados), por lo que no es necesario pedir y liberar memoria, de forma explícita, para crear y destruir objetos.

- Cabe destacar también su gratuidad. Puedes descargar el intérprete de la página web http://www.python.org.

Sin embargo, señalar como desventaja que, al ser un lenguaje interpretado, representa mayor lentitud mientras se ejecuta el programa que los lenguajes compilados.

Módulos y paquetes

Python, a su vez, está compuesto por módulos y paquetes. Los módulos son ficheros codificados que permiten una organización y división lógica de nuestro código, facilitando el mantenimiento y la lectura de programas demasiado largos (Montoro 2012). Además, Python también nos permite unificar distintos módulos bajo un mismo nombre configurando jerarquías de módulos y submódulos, y constituyendo de esta forma, lo que se denomina, un paquete.

Dentro de los numerosos paquetes que Python nos ofrece, para la realización de este trabajo se han usado NetworkX y Matplotlib.

NetworkX es un paquete o librería para la creación, manipulación y estudio de las estructuras, dinámicas y funciones de redes complejas (contenido extraído de https://networkx.github.io).

Tras haber definido los distintos elementos necesarios para constituir la estructura de la red (nodos, arcos, longitudes, etc.), dicho paquete ha permitido su creación y su posterior manipulación. Para su utilización ha sido necesario importarlo previamente. Esto resulta tan sencillo como utilizar al comienzo del código la instrucción: *from networkx import **

Concretamente, dentro de todos los que nos ofrece, se han utilizado los siguientes algoritmos y funciones:

- ***dijkstra_path* (*G, source, target, weight='weight'*)**

Nos devuelve el camino más corto (lista de nodos) del grafo G, para ir desde un nodo origen (source) hasta uno destino (target)

- ***dijkstra_path_length (G, source, target, weight='weight')***

Nos devuelve la longitud del camino más corto obtenido.

- ***def k_shortest_paths (G, source, target, k, weight=None)***

Función que devuelve los k (o < k) caminos más cortos para ir desde un nodo origen (source) hasta uno destino (target).

Por otro lado, Matplotlib es una biblioteca de trazado en 2D de Python, que produce figuras en una gran variedad de formatos impresos y entornos interactivos para todas las plataformas. Puede generar gráficos, histogramas, espectros de potencia, gráficos de barras, diagramas de errores, diagramas de dispersión, etc., con solo pocas líneas de código (contenido extraído de https://matplotlib.org).

Dicho paquete, concretamente su módulo pyplot, ha permitido la representación gráfica en los ejes cartesianos de los recorridos realizados por cada una de las líneas de la red (tiempos de recorrido en cada arco, tiempos de parada en cada estación, intervalo de tiempo entre trenes consecutivos, etc.). Para su utilización ha sido necesario escribir al comienzo del código la instrucción: *from matplotlib import pyplot.*

Solver

Para resolver mediante Python el modelo de optimización propuesto, se va a recurrir al empleo de un “solver”. Este es un término genérico referido a una pieza de software matemático que ’resuelve’ un problema matemático. Para ello, toma como entrada la descripción del problema y luego calcula su solución (Mazo 2011).

Concretamente se ha escogido el software GUROBI, un “solver” comercial para la programación lineal (LP), programación cuadrática (QP), programación cuadrática restringida (QCP), programación lineal entera mixta (MILP), programación cuadrática entera mixta (MIQP) y programación entera mixta restringido cuadráticamente (MIQCP). (contenido extraído de http://www.gurobi.com/).

4.2 Estructuras de datos del modelo en Python

En este apartado, se hace una descripción detallada de todas las estructuras de datos (concretamente, listas y diccionarios) que ha sido necesario crear, para poder llevar a cabo mediante lenguaje de programación todo el proceso de resolución planteado en el apartado 2.3. del presente trabajo.

Inicialmente, es necesario crear una serie de listas que van a definir la estructura de la red con la que se va a trabajar. Por lo tanto, todas las listas que se muestran a continuación son datos de entrada proporcionados por el usuario:

- **Nodos:** Lista con todos los nodos de la red.

- **Aristas**: Lista con todas las aristas de la red.

- **LongAristas**: Lista con las longitudes de todas las aristas de la red (metros).

- **NodosL1, …, NodosLN:** Se definen tantas listas como líneas haya. Contienen los nodos que componen cada línea.

- **AristasL1, …, AristasLN:** Se definen tantas listas como líneas haya. Contienen las aristas que componen cada línea.

- **L_Nodos_lineas**: Lista construida por el usuario a partir de la concatenación de las listas de nodos de cada una de las líneas que conforman la red, de esta forma se tienen todos los nodos en una sola lista.

- **L_Aristas_lineas**: Lista construida por el usuario a partir de la concatenación de las listas de aristas de cada una de las líneas que conforman la red, de esta forma se tienen todas las aristas en una sola lista.

- **L_OrigenDestino**: Lista que contiene los valores de la demanda de los pares OD. Dicha lista, multiplicada por el **factor de escala** (también es dato de entrada), se corresponde con las personas que desean viajar desde un nodo i a uno j, para un período de una hora.

A partir de las listas anteriores, se construyen otra serie de estructuras de datos que harán falta para cálculos posteriores. Estas listas ya no las proporciona el usuario.

- **D_OrigenDestino**: Diccionario de ParesOD con el correspondiente factor de escala aplicado. Almacena el valor de la demanda de pasajeros que va desde i a j, donde i, j son nodos del grafo y, además, serán la clave de entrada al diccionario.

- **Arcos**: Lista que contiene todos los arcos de la red, orientados.

- **L_Arcos_Long:** Lista de tuplas que contiene 3 valores para cada elemento. Cada tupla contiene el origen, el destino y la longitud del arco orientado (i, j, Long). Se necesita para montar el grafo con pesos (longitudes) en arcos a partir de las librerías networkx y poder trabajar con la clase Digrafo.

- **D_Arcos_Long**: Diccionario cuyas claves son los diferentes arcos orientados de la red, y cuyos valores son sus correspondientes longitudes en metros. D_Arcos_long [(i, j)]: Long.

- **L_Arcos_lineas:** Lista que contiene el conjunto de arcos que pertenecen a cada línea. Cada elemento es a su vez una lista con tuplas de los arcos. Por ejemplo, si la línea 0 está formada por los nodos 1->2->3 y la línea 1 está formada por 4->5->6, se tendría: [[(1,2), (2,1), (2,3), (3,2)], [(4,5), (5,4), (5,6), (6,5)]]. Es decir, se contemplan todos los arcos, tanto en sentido 'up' (sentido en el que se definió la lista de nodos que formaban cada una de las líneas Linea0: [1,2,3] Linea1: [4,5,6] como en el contrario 'dw', y aparecen ordenados de forma consecutiva, primer arco y su contrario, segundo arco y su contrario, y así sucesivamente. L_Arcos_lineas[i] devuelve una lista con todos los arcos de la línea i.

- **D_ArcosLinea_up:** Diccionario que contiene para cada línea una lista de tuplas con los arcos que la componen en sentido up (ida). Por ejemplo, D_ArcosLinea_up [1] devuelve [(4, 1), (1, 2), (2, 5), (5, 6)]

- **D_ArcosLinea_dw:** Diccionario que contiene para cada línea una lista de tuplas con los arcos que la componen en sentido down (vuelta). Por ejemplo, D_ArcosLinea_dw [1] devuelve [(6, 5), (5, 2), (2, 1), (1, 4)]

- **D_LineasporNodo:** Diccionario que contiene las líneas que pasan por cada nodo de la red. El nodo es la clave de acceso, y los valores de este, equivalen a las líneas que pasan por el mismo. De esta forma obtenemos cuales son las líneas que pasan por cada estación. Por ejemplo, si por el nodo 1 pasan las líneas 0, 1 y 2 tendríamos: D_LineasporNodo [1] devuelve [0, 1, 2].

- **D_NodosTransbordo:** Diccionario que contiene los nodos de transbordo de la red. Para cada elemento (nodo) indexado por su código, se proporciona una lista que contiene las líneas que pasan por ese nodo (siempre que haya más de 1). Por ejemplo, si en la red hay dos nodos de transbordo, el 5 y el 7, y por el nodo 5 pasan la línea 0 y la 3, y por el nodo 7 pasan las líneas 1 y 4 tendríamos: D_NodosTransbordo devuelve {5: [0,3], 7: [1,4]}

- **D_LineasPorArco:** Diccionario que contiene el conjunto de líneas que atraviesan un arco en forma de lista. La clave es el arco, y sus valores las diferentes líneas que pasan por este. Por ejemplo, si el arco (1,2) está recorrido por las líneas 2 y 3, tendríamos {(1,2): [2,3]}

Posteriormente, una vez se ha generado la estructura del grafo mediante el ya mencionado paquete Networkx, seremos capaces de obtener las estructuras de datos que se muestran a continuación, las cuales permiten llevar a cabo el proceso de asignación de tránsito.

- **D_Shortest_path:** Diccionario que contiene el camino más corto en el grafo para cada par OD (i, j). Indexado por el par, contiene una lista de nodos que definen el camino. Por ejemplo, si para ir del nodo 1 al 3 el camino más corto es el que pasa por los nodos 1, 2, 3 tendríamos: D_Shortest_path [1, 3] devuelve [1, 2, 3].

- **D_Shortest_path_Long:** Diccionario que contiene la longitud (metros) del camino más corto en el grafo para cada par OD. La clave de entrada al diccionario será el par OD.

- **D_K_Shortest_paths:** Diccionario que contiene los K (o < K) mejores caminos en el grafo para cada par origen destino. Se recalculan en función de las frecuencias de las líneas de forma iterativa. Por ejemplo, si para ir del nodo 1 al 3 hay 3 caminos diferentes tendríamos: D_K_Shortest_paths [1, 3]: [[1, 2, 3], [1, 7, 8, 3], [1, 6, 5, 2, 3]]

- **D_K_Long_shortest_paths:** Diccionario que contiene las longitudes de los K mejores caminos (más cortos) en el grafo para cada par OD. La clave de entrada al diccionario será el par OD.

- **D_Num_shortest_paths:** Diccionario que contiene el número de mejores caminos encontrados (<=K) (más cortos) en el grafo para cada par OD. La clave de entrada al diccionario será el par OD.

- **D_Ext_Shortest_paths:** Diccionario que contiene los mejores caminos extendidos para cada par OD. Al extenderse los caminos, ahora ya no están dados en una lista de nodos, sino que tenemos una serie de tuplas de 3 elementos, formadas por los arcos que componen el camino junto con la línea de la red a la que estos pertenecen. De tal forma que si antes un camino era [1, 2, 3] ahora queda de la forma [(1, 2, línea), (2, 3, línea)]. Al haberse extendido los K caminos iniciales, se obtienen más posibilidades para cada uno de ellos, debido a que un mismo arco puede pertenecer a más de una línea. Tras esta expansión, se realizan dos procesos para eliminar las estrategias “dominadas” y así obtener el diccionario final D_Ext_Shortest_paths: el primer procedimiento, para cada par OD, contabiliza el número de trasbordos que hay que realizar para cada una de las posibles estrategias, guarda el menor valor, y mantiene solamente aquellas estrategias con dicho valor mínimo. Posteriormente se realiza un segundo procedimiento, en el cual, para cada par OD, se contabiliza la longitud de viaje de cada una de las estrategias restantes y guarda el menor valor. Tras esto, procede a eliminar todas aquellas estrategias del par, cuya longitud de viaje supere a la mínima en un 10%.

- **D_Ext_Lon_shortest_paths:** Diccionario que contiene las longitudes de los mejores

caminos (más cortos) extendidos para cada par origen destino, tras la realización de la expansión y las dos purgas. La clave de entrada al diccionario será el par OD.

- **D_Ext_Num_shortest_paths:** Diccionario que contiene el número de mejores caminos extendidos para cada par origen destino que han quedado tras la realización de la expansión y las dos purgas. La clave de entrada al diccionario será el par OD.

- **D_Ext_Prop_shortest_paths:** Diccionario que contiene las proporciones en las que se reparte la demanda por cada uno de los caminos extendidos que sirven el par OD (i, j). Cada clave (i, j) tendrá tantas proporciones como elementos en D_Ext_Num_shortest_paths. Cuando solo hay un camino la proporción es 1. En la primera asiganción, la proporción va en función de la longitud de cada camino, por lo que este diccionario se apoya en otro denominado D_Ext_Sum_long_paths que contiene la suma de las longitudes de todos los caminos que sirven a cada par OD. En las iteraciones posteriores, las proporciones irán en función del tiempo total de cada camino. Constituye la base para la asignación que se realiza en cada iteración.

- **D_Ext_Sum_long_paths:** Diccionario que almacena la suma de las longitudes de todos los caminos extendidos (tras las 2 purgas) que sirven a cada par OD (i, j). Este diccionario se recalcula, de tal forma que, una vez se haya resuelto el modelo de optimización, las longitudes de los caminos serán sustituidas por los tiempos de viaje. Se usa para el cálculo de las proporciones de demanda sobre cada camino. La clave de entrada al diccionario será el par OD.

- **D_Ext_Arcos_paths:** Diccionario que contiene la lista de arcos que forman cada uno de los caminos extendidos que sirven cada par OD de la red. La clave de entrada al diccionario será el par OD.

- **D_Ext_Carga_Arco:** Diccionario que contiene la carga de cada uno de los arcos de la red (suma de pasajeros de todos los caminos que lo atraviesan), una vez que se ha realizado la asignación. Habrá que recalcularlo en cada iteración. Se usa para determinar la convergencia, la cual se alcanza cuando en dos iteraciones consecutivas las cargas sobre los arcos se mantienen prácticamente iguales. La clave de entrada al diccionario será cada arco de la red.

- **L_Ext_CargaMax_linea:** Lista que contiene el valor de la carga del segmento más cargado (mayor número de personas circulan por él) para cada una de las líneas de la red. Este valor servirá para seleccionar el modelo de tren (en función de su capacidad) que requiere cada línea.

- **D_Ext_Destino_nodo_up, D_Ext_Destino_nodo_dw:** Diccionarios que contienen la cantidad de personas que se bajan en cada nodo “t” perteneciente al sentido up (down) de la

línea en estudio, debido a que han llegado a su destino. La clave de entrada al diccionario será el arco que lleva al nodo junto con la línea a la que pertenece, es decir, D_Ext_Destino_nodo_up [línea, (s, t)].

- **D_Ext_Transbordo_nodo_up, D_Ext_Transbordo_nodo_dw:** Diccionarios que contienen la cantidad de personas que se bajan en cada nodo "t" perteneciente al sentido up (down) de la línea en estudio para transbordar hacia otras líneas de la red. La clave de entrada al diccionario será el arco que lleva al nodo junto con la línea a la que pertenece, es decir, D_Ext_Transbordo_nodo_up [línea, (s, t)].

- **D_Ext_Bajadas_nodo_up, D_Ext_Bajadas_nodo_dw:** Diccionarios que contienen la cantidad total de personas que se bajan (los que llegan a destino y los que se bajan desde la línea en estudio para transbordar hacia otras líneas de la red) en cada nodo "t" perteneciente al sentido up (down) de la línea en estudio. La clave de entrada al diccionario será el arco que lleva al nodo junto con la línea a la que pertenece, es decir, D_Ext_Bajadas_nodo_up [línea, (s, t)].

- **D_Ext_Origen_nodo_up, D_Ext_Origen_nodo_dw:** Diccionarios que contienen la cantidad de personas que se suben en cada nodo "s" perteneciente al sentido up (down) de la línea en estudio, debido a que comienzan su trayecto en dicho nodo. La clave de entrada al diccionario será el arco que parte desde el nodo junto con la línea a la que pertenece, es decir, D_Ext_Origen_nodo_up [línea, (s, t)].

- **D_Ext_TransbordoS_nodo_up, D_Ext_TransbordoS_nodo_dw:** Diccionarios que contienen la cantidad de personas que se suben en cada nodo "s" perteneciente al sentido up (down) de la línea en estudio, debido a que están trasbordando desde otras líneas de la red. La clave de entrada al diccionario será el arco que parte desde el nodo junto con la línea a la que pertenece, es decir, D_Ext_TransbordoS_nodo_up [línea, (s, t)].

- **D_Ext_Subidas_nodo_up, D_Ext_Subidas_nodo_dw:** Diccionarios que contienen la cantidad total de personas que se suben (los que inician su viaje y los que se suben hacia otras líneas para transbordar desde la línea en estudio) en cada nodo "s" perteneciente al sentido up (down) de la línea en estudio. La clave de entrada al diccionario será el arco que parte desde el nodo junto con la línea a la que pertenece, es decir, D_Ext_Subidas_nodo_up [línea, (s, t)].

- **SubidasenPlatUp, SubidasenPlatDw:** Lista que contiene a su vez tantas listas como líneas haya. Dentro de estas, se encuentran los mismos valores que en D_Subidas_nodo_up y D_Subidas_nodo_dw, solo que ahora son subidas en cada plataforma de la línea y además se impone que, en la última de ellas, las subidas tomen valor 0. Se usarán para poder dimensionar los dwell times (tiempo de parada de cada tren para cargar y descargar personas).

- **BajadasenPlatUp, BajadasenPlatDw:** Lista que contiene a su vez tantas listas como líneas haya. Dentro de estas, se encuentran los mismos valores que en D_Ext_Bajadas_nodo_up, D_Ext_Bajadas_nodo_dw, solo que ahora son bajadas en cada plataforma de la línea y además se impone que, en la primera de ellas, las bajadas tomen valor 0. Se usarán para poder dimensionar los dwell times (tiempo de parada de cada tren para cargar y descargar personas).

Finalmente, ha sido necesario crear una serie de listas donde almacenar los datos obtenidos tras la resolución del modelo de optimización:

- **TRup, TRdw:** Lista que contiene a su vez tantas listas como líneas hay. Dentro de estas, se encuentran los tiempos de recorrido (segundos) obtenidos para cada vía (i, j) de la línea correspondiente en sentido up (down).

- **TPup, TPdw:** Lista que contiene a su vez tantas listas como líneas hay. Dentro de estas, se encuentran los tiempos de parada (segundos) obtenidos para cada estación de la línea correspondiente en sentido up (down).

- **frec:** Lista donde se almacena la frecuencia (trenes/hora) obtenida para cada línea, por lo que tiene tanto elementos como líneas haya.

- **head:** Lista donde se almacena el “headway” (segundos) obtenido para cada línea, por lo que tiene tanto elementos como líneas haya.

- **TF:** Lista donde se almacena el tamaño de flota obtenido para cada línea, por lo que tiene tanto elementos como líneas haya.

- **TC:** Lista donde se almacena el tiempo de ciclo (segundos) obtenido para cada línea, por lo que tiene tanto elementos como líneas haya.

5. APLICACIÓN PRÁCTICA

Para llevar a la práctica el modelo propuesto con el propósito de evaluar su rendimiento, se ha escogido representar una red de cercanías real, resultando de esta forma más atractiva y veraz la ilustración práctica del modelo. Concretamente, se va a trabajar con una parte de la red de cercanías de Valencia, la cual, está compuesta por 6 líneas, 252 km de vías férreas con un ancho de vía ibérico (1688 mm), y 66 estaciones.

A continuación, se puede observar una representación gráfica de la red mencionada.

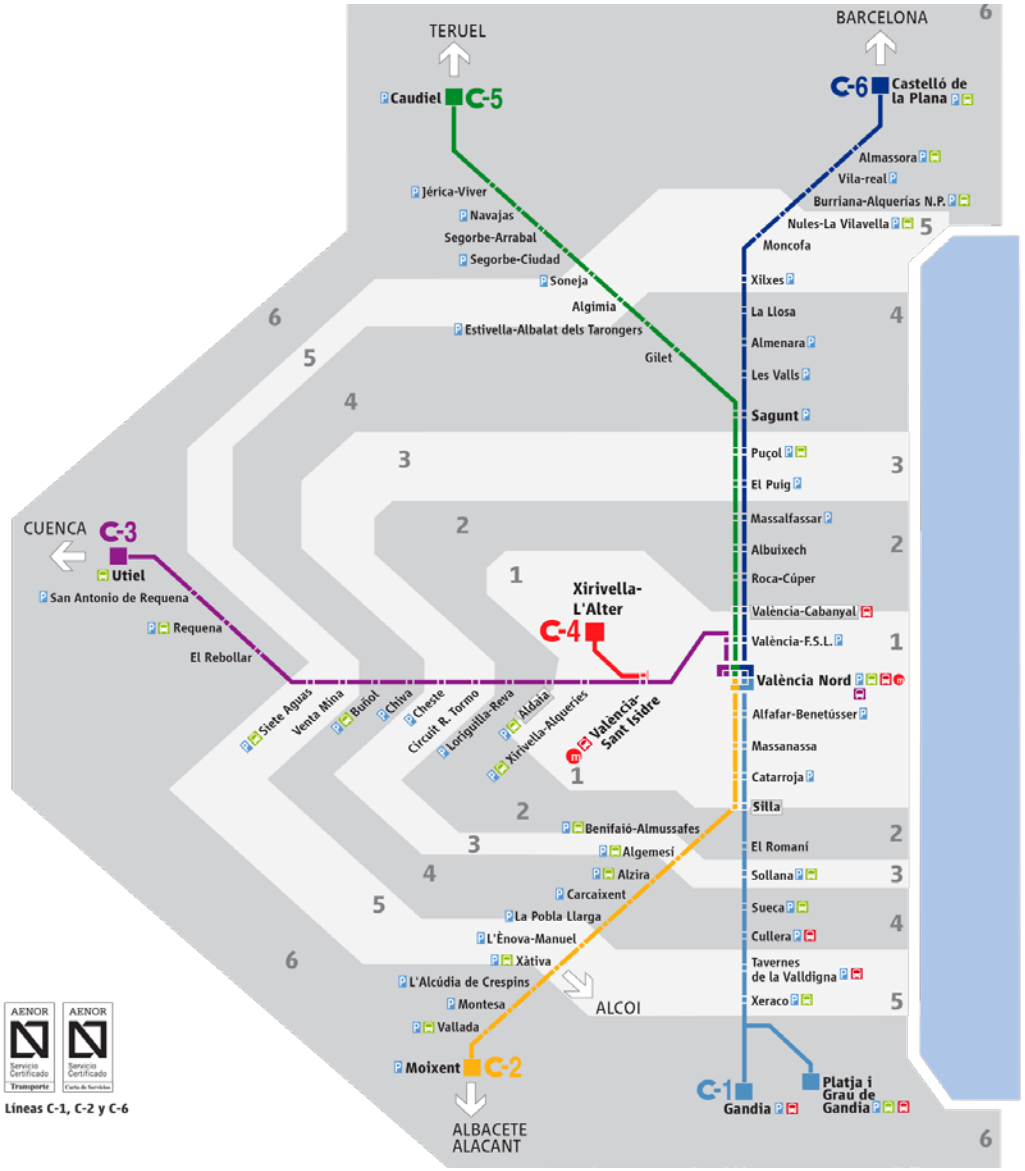


Figura 10. Plano del Cercanías de Valencia (Recuperado de www.renfe.com)

A pesar de que esta red cuenta con 6 líneas, la aplicación práctica se va a realizar solamente para las líneas C-1, C-2 y C-6, debido a que sus características son las que mejor se adaptan a la tipología del modelo planteado. Las tres líneas restantes, cuentan en la realidad con una frecuencia de paso muy baja, y por ello, aplicar el modelo planteado en dichas líneas no resultaría adecuado, debido entre otros motivos, a que como se mencionó con anterioridad al describir el problema, la asignación basada en frecuencias resulta de utilidad y proporciona buenos resultados cuando se trabaja con frecuencias de paso elevadas.

En el presente apartado, en un inicio se expondrán de forma detallada todos los datos de entrada que el modelo necesita. Posteriormente, se continúa exponiendo los resultados que han ido siendo obtenidos hasta alcanzar la convergencia deseada. Una vez lograda, se pasa a realizar una explicación de los resultados finales, junto con una representación gráfica que facilite su comprensión.

5.1 Datos de entrada del modelo

5.1.1 Características de la red

Lo primero a la hora de llevar a cabo el modelo, será la construcción de la red. Para ello será necesario definir el conjunto de líneas que la componen (en este caso solamente las tres primeras), junto con sus correspondientes estaciones y longitudes existentes entre cada una de ellas.

Para facilitar la ilustración de los datos, a continuación, se muestran una serie de tablas, una por cada línea, en las cuales aparece recogida la información que será de utilidad para la implementación del modelo. Conviene resaltar que, como se observa en las siguientes tablas, se ha establecido una numeración para las 41 estaciones que componen las líneas con las que se trabajará. A su vez, la numeración de las vías irá en función de las estaciones que estas separen. Como se puede apreciar en la Figura 10, existen tramos de recorrido compartidos por varias líneas, no obstante, a pesar de dichos tramos tener la misma numeración, cada línea dispondrá de su propia infraestructura, lo que supone que cada línea tiene sus propias vías y plataformas.

Número de estación	Nombre de estación	Vía	Longitud vía (m)
1	Valencia Nord	(1, 2)	5110
2	Alfafar-Benetússer	(2, 3)	1530
3	Massanassa	(3, 4)	1020
4	Catarroja	(4, 5)	4460
5	Silla	(5, 6)	6800
6	El Romaní	(6, 7)	3040
7	Sollana	(7, 8)	10160
8	Sueca	(8, 9)	5160
9	Cullera	(9, 10)	11820
10	Tavernes de la Valldigna	(10, 11)	6510
11	Xeraco	(11, 12)	7310
12	Gandía		

Figura 11. Datos de la línea C-1

Número de estación	Nombre de estación	Vía	Longitud vía (m)
1	Valencia Nord	(1, 2)	5110
2	Alfafar-Benetússer	(2, 3)	1530
3	Massanassa	(3, 4)	1020
4	Catarroja	(4, 5)	4460
5	Silla	(5, 13)	8900
13	Benifaió-Almussafes	(13, 14)	10200
14	Algemesí	(14, 15)	5500
15	Alzira	(15, 16)	3600
16	Carcaixent	(16,17)	4500
17	La Pobla Llarga	(17, 18)	4700
18	Manuel-L'Ènova	(18, 19)	7100
19	Xàtiva	(19, 20)	10500
20	L'Alcúdia de Crespins	(20, 21)	5800
21	Montesa	(21, 22)	6100
22	Vallada	(22, 23)	6900
23	Moixent		

Figura 12. Datos de la línea C-2

Número de estación	Nombre de estación	Vía	Longitud vía (m)
1	Valencia Nord	(1, 24)	3820
24	València-F.S.L	(24, 25)	1610
25	València-Cabanyal	(25, 26)	6270
26	Roca-Cuper	(26, 27)	1800
27	Albuixech	(27, 28)	1630
28	Massalfassar	(28, 29)	3690
29	El Puig	(29, 30)	3290
30	Puçol	(30, 31)	7130
31	Sagunt	(31, 32)	5540
32	Les Valls	(32, 33)	4120
33	Almenara	(33, 34)	2300
34	La Llosa	(34, 35)	1810
35	Xilxes	(35, 36)	4400
36	Moncofa	(36, 37)	4180
37	Nules-La Vilavella	(37, 38)	6370
38	Burriana	(38, 39)	4230
39	Vila-real	(39, 40)	2960
40	Almassora	(40, 41)	9670
41	Castelló de la Plana		

Figura 13. Datos de la línea C-6

5.1.2 Demanda

Con el objetivo de estimar una demanda aproximada que no se aleje mucho de la realidad, se van a tomar como datos de entrada los resultados de un estudio publicado por el Ministerio de Fomento de España en su página web (Figura 14), en el cual, se puede observar la demanda diaria media de cada una de las líneas que componen la red.

Basándonos en tal estudio, tomaremos como 20.994, 26.038 y 18.517, el número de pasajeros diarios de las líneas C-1, C-2 y C-6, respectivamente. Los demás datos de la gráfica no serán utilizados, debido a que no se va a trabajar con las correspondientes líneas, sin embargo, estos nos sirven para comprobar como la demanda de las líneas C-3, C-4 y C-5, es muy baja y, por lo tanto, sus respectivas frecuencias también lo serán, lo que permite constatar que el modelo planteado no proporcione buenos resultados para dichas líneas. Cabe mencionar que, siguiendo este razonamiento, estamos distorsionando una cierta cantidad de demanda, debido a

que en los datos que refleja la Figura 14, las líneas escogidas interfieren también con las que han sido descartadas, habiendo probablemente pasajeros que quieran viajar desde un punto de una línea "desechada", hacia un punto de una que sí ha sido seleccionada y/o viceversa, y, sin embargo, este aspecto no está siendo considerado.

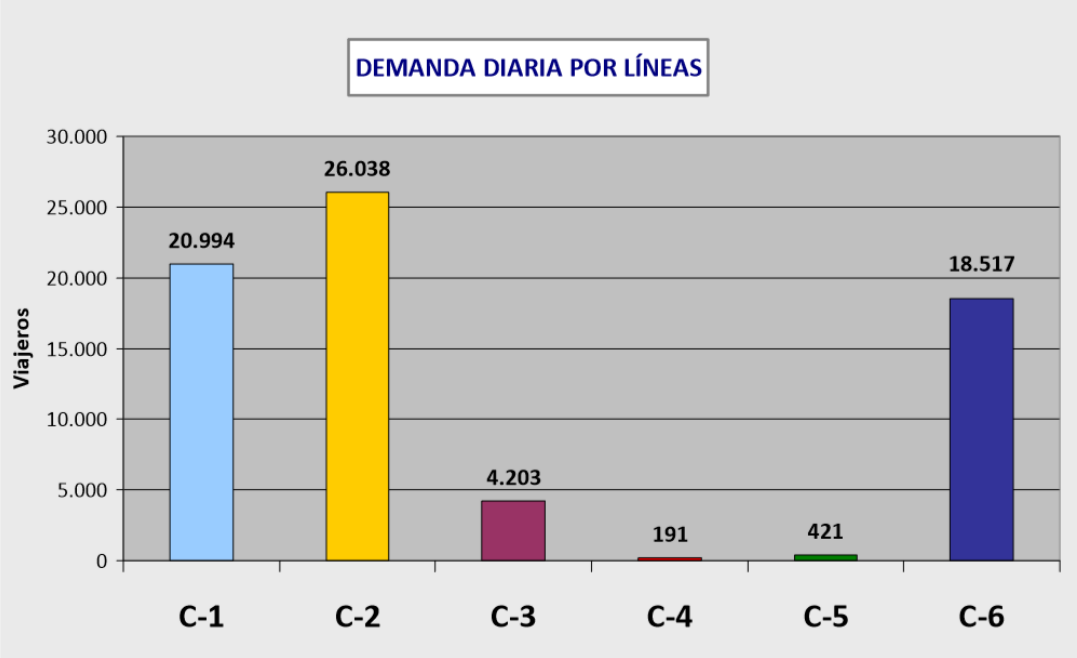


Figura 14. Estadísticas de la demanda diaria por líneas del cercanías de Valencia.

Por otro lado, como se mencionó con anterioridad, la matriz de demanda OD que se utilizará en el modelo se corresponderá con los valores de una de las horas puntas del día. Un estudio realizado por la plataforma de Promoción del Transporte Público (PTP), llega a la conclusión de que la demanda en la hora punta de un día laborable (14:00 a 15:00 horas), suele ser del 10% del total diario, por lo que tomaremos dicho valor para estimar una Matriz que, en la medida de lo posible, se ajuste a la realidad. En base a este porcentaje, obtenemos que la demanda en hora punta para cada línea sería aproximadamente la siguiente:

- C-1: 2.099 pasajeros
- C-2: 2.604 pasajeros
- C-6: 1.852 pasajeros

Debido a que no se dispone de la demanda entre cada estación, se va a definir una matriz aleatoria, de tal forma que, la suma de todos sus elementos coincida con la suma total de la

demanda en hora punta que existe en las tres líneas escogidas, la cual tiene un valor de 6.555 pasajeros (comprobaremos que dicho valor se cumple mediante un sumatorio denominado "DemandaTotal"). Además, se va a incluir en el modelo un factor de escala que permita modificar todos los datos de la matriz proporcionalmente, para de esta forma poder tener en cuenta otras horas del día o eventos especiales que puedan ocasionar un cambio en la demanda. Junto con ello, también merecen ser tenidos en cuenta a la hora de construir la matriz, ciertos aspectos sobre algunas estaciones:

- La estación de Valencia Nord, la cual se corresponde con el número 1 de los definidos anteriormente, es con diferencia la más transitada de todas. Por ello, de ella partirán y a ella llegarán el mayor número de pasajeros.

- Seguida de esta se encuentran las estaciones de Castelló de la Plana, Xàtiva, Gandía, Silla y Alfafar-Benetússer, las cuales vienen representadas con los números 41, 19, 12, 5 y 2 respectivamente. Este conjunto de estaciones también contará con un alto número de llegadas y salidas, aunque no tantas como la estación de Valencia Nord.

- Por el contrario, las estaciones que cuentan con la menor demanda del servicio son El Romaní, L'Alcúdia de Crespins, Montesa, Vallada, Moixent, Roca Cúper, Massalfassar, Les Valls, La Llosa y Moncofa, las cuales se corresponden con la numeración 6, 20, 21, 22, 23, 26, 28, 32, 34 y 36 respectivamente. De hecho, como se puede comprobar en los horarios disponibles en www.redtransporte.com/valencia/cercanias-renfe/, dichas estaciones cuentan en la realidad con una frecuencia de servicio mucho menor al resto, debido a que no disponen de la suficiente demanda. En la matriz, se podrá observar como el número de pasajeros que tienen como origen o destino alguna de las estaciones mencionadas será bastante bajo.

Resulta obvio que la matriz obtenida no refleja la realidad tal cual es, ya que no se dispone de datos suficientes y se han tenido que realizar muchas suposiciones para obtenerla, sin embargo, esto no supondrá ningún problema, ya que el objetivo es comprobar el funcionamiento del modelo realizado, por lo que será suficiente con tener una matriz consistente.

Por otro lado, debido a las dimensiones que ocupa la matriz, será definida directamente en el modelo desarrollado en Python, el cual, se puede encontrar en el anexo final del documento.

5.1.3 Modelos de tren

Para la selección del tipo de tren que va a requerir cada línea, se ha optado por elegir entre las posibilidades que nos ofrecen las series de trenes CIVIA, debido a que estos modelos de tren están siendo implementados en la actualidad en la red de Cercanías de Valencia. Toda la información que se muestra a continuación sobre la gama de trenes con la que se va a trabajar, ha sido extraída de la página web de RENFE, concretamente del enlace http://www.renfe.com/viajeros/nuestros_trenes/civia.html.

La serie 46X de Civia es un tipo de material rodante autopropulsado, que entró en servicio en el año 2004, destinado al servicio de viajeros en trenes de Cercanías y desarrollado por CAF (Construcciones y Auxiliar de Ferrocarriles) para Renfe. La numeración de estos trenes sería 462, 463, 464 y 465, pero al poder mezclarse todas las series, se decidió agrupar a todos ellos con el nombre de Civia.

Estos constituyen el origen de una nueva gama de trenes de Cercanías destinada a satisfacer las necesidades de transporte en grandes zonas metropolitanas, debido a que, entre sus múltiples características, ofrecen un gran confort, una alta eficiencia y permiten disminuir los costes energéticos y de explotación.

La característica más visible de esta gama de trenes es su modularidad, la cual, permite la formación de composiciones con diferente número coches en función de la hora del día, la línea por la que estén circulando en ese momento, etc.

Existen tres tipos de coches: un coche extremo con cabina de conducción y piso alto; un intermedio de piso alto; y un intermedio con W.C. y piso bajo a la altura del andén de cercanías, con objeto de facilitar el acceso a personas con movilidad reducida. Con las diferentes combinaciones de estos coches se da lugar a las series 462 (dos coches con cabina), 463 (dos coches con cabina y un remolque de piso bajo), 464 (dos coches con cabina, un remolque de piso bajo y otro de piso alto) y 465 (dos coches con cabina, un remolque de piso bajo y dos de piso alto).

Figura 15. Dibujo esquemático de un Civia compuesto por 5 coches.

Todos los trenes Civia, sea cual sea su número de coches, tienen idénticas características (mismas prestaciones funcionales y de confort; mismos sistemas de potencia, control, y de información al viajero) diferenciándose únicamente en su capacidad de transporte.

En la siguiente tabla, se muestran aquellos datos pertenecientes a las fichas técnicas disponibles en la web, que serán de utilidad para el modelo de optimización propuesto:

	Serie 462	**Serie 463**	**Serie 464**	**Serie 465**
Velocidad mínima (km/h)	40	40	40	40
Velocidad máxima (km/h)	120	120	120	120
Consumo (kWh/km)	2,73	3,62	4,5	5,4
Número de coches	2	3	4	5
Número de puertas	4	6	8	10
Plazas sentadas	126	169	223	277
Plazas totales	414	607	832	997

Tabla 6. Datos técnicos de las series CIVIA.

A partir de los datos de la tabla anterior, se pueden obtener las tasas de subida y bajada empíricas (medidas en segundos/pasajero) para cada modelo, definidas en el modelo como a^r y b^r respectivamente. Partiendo de la base de que para una sola puerta se estima una tasa de 0,5 segundos/pasajero, y teniendo a su vez en cuenta el número total de puertas de cada modelo, se obtienen los siguientes valores:

- Serie 462: 0,125 segundos/pasajero
- Serie 463: 0,08333 segundos/pasajero
- Serie 464: 0,0625 segundos/pasajero
- Serie 465: 0,05 segundos/pasajero

5.1.4 Costes asociados a la función objetivo

Recordemos del apartado anterior, que en la función objetivo podíamos distinguir 5 parámetros diferentes para definir los costes (θ_1^r, θ_2, β_1, β_2, β_3). A continuación, se procede a detallar como se ha obtenido el valor de cada uno de ellos.

- **Costes para el operador:**

Costes de operación:

Como se puede recordar del apartado en que se describió la función objetivo, los costes de operación (medidos en €/tren·km) dependían de varias partidas. En concreto, según se ha podido obtener a partir de los datos disponibles en la enciclopedia colaborativa del ferrocarril en España (www.ferropedia.es), los porcentajes aproximados que le corresponderían a cada una de ellas serían: la energía consumida (17%), amortización y depreciación (20%), mantenimiento de los trenes (14%), los pagos a ADIF por infraestructura y estaciones (13%), los intereses y gastos financieros (6%), y otros materiales y servicios (24%). Teniendo en cuenta todo ello, obtenemos aproximadamente los siguientes costes para cada modelo de tren:

- Serie 462: 6,60 € / tren·km.
- Serie 463: 7,53 € / tren·km.
- Serie 464: 8,46 € / tren·km.
- Serie 465: 9,79 € / tren·km.

Costes de la tripulación:

Desde mayo del año 2000, la dirección de Renfe implantó en todos sus trenes el agente único de conducción (www.elpais.com), por lo que, desde entonces, todos los trenes en España circulan con un único maquinista sin ayudante alguno. Como consecuencia de ello, el único coste de personal que se va a contemplar en el modelo es el salario anual de un maquinista.

En la siguiente imagen extraída del Boletín Oficial del Estado (BOE) con fecha 29 de noviembre de 2016, podemos observar los diferentes salarios establecidos para el grupo de maquinistas en el I Convenio colectivo del Grupo de empresas RENFE.

	FIJO CLAVE 002 €/AÑO	COMPLEMENTO DE CONDUCCIÓN €/AÑO CLAVE 280	(1) SEGURIDAD, FORMACIÓN Y/O GESTIÓN - €/MES CLAVE 281	(2) GASTOS DE VIAJE - €/DÍA CLAVE 551
MANDO INTERMEDIO DE CONDUCCIÓN/JEFE DE MAQUINISTAS NIVEL A	40.242,72	7.272,96	259,77	12,988500
MANDO INTERMEDIO DE CONDUCCIÓN/JEFE DE MAQUINISTAS NIVEL B		5.194,92		
MAQUINISTA JEFE DEL TREN	33.381,00	7.272,96		
MAQUINISTA PRINCIPAL	27.045,72	5.194,92		
MAQUINISTA (Anterior I Convenio Colectivo Grupo Renfe)	26.795,76	2.077,92		
MAQUINISTA (I Convenio Colectivo Grupo Renfe)	23.260,68	2.077,92		
MAQUINISTA DE ENTRADA	17.743,32	1.038,96	155,85	

Figura 16. Tablas salariales de los maquinistas.

En base a la imagen anterior, considerando que los trenes de cercanías serán conducidos por un Maquinista Principal, y teniendo a su vez en cuenta que, según el Convenio Colectivo del Grupo RENFE, se establecen 1728 horas de trabajo anuales distribuidas en 216 días, se define el siguiente coste anual de personal:

27.045,72 €/año + 5.194,92 €/año + 259,77 €/mes · 12 meses + 12,9885 €/día · 216 días = 38163.396 €/año

Si dividimos esta cantidad entre las 1728 horas, obtenemos un coste de personal de 22,085 €/hora.

- **Costes para los pasajeros:**

Se representaban mediante los parámetro**s** β_1, β_2 y β_3**,** los pesos que tienen en el objetivo cada uno de los tres aspectos (tiempo a bordo del tren, número de transferencias y tiempos de espera) que representaban coste alguno para los usuarios de la red.

Para obtener dichos valores, hemos vuelto a recurrir al estudio de (Robenek et al. 2016) el cual, a su vez, se apoya en razonamientos realizados en estudios previos al suyo por otros autores. De esta forma, los distintos pesos son obtenidos a partir de comparar la falta de utilidad o pérdida del tiempo que supone al pasajero cada uno de los aspectos con relación al tiempo que está a bordo del tren. Por lo tanto, se tienen los siguientes valores:

- β_1=1, ya que el tiempo a bordo del tren está siendo comparado con sí mismo.

- β_2=10. Cada transferencia que realiza un pasajero se penaliza con 10 minutos en relación con el tiempo dentro del tren.
- β_3=2,5. El tiempo que un pasajero pasa esperando la llegada de un tren, le supone perder 2,5 veces más tiempo que si estuviese a bordo del tren.

Por otro lado, resulta necesario definir también el valor del término "VOT". Obviamente, resulta muy complicado otorgarle un valor económico al tiempo, ya que este depende de multitud de factores como la renta familiar, el tipo de viaje, la edad del viajero, etc. Por ello, (Robenek et al. 2016) utilizan un valor promedio de 27,81 CHF/hora, el cual equivale a 24,17 €/hora según el cambio actual, lo que básicamente se traduce como el precio que un viajero estaría dispuesto a pagar por ahorrarse una hora de viaje.

5.1.5 Otros datos de entrada

Finalmente, existen otra serie de parámetros que también necesitan ser definidos antes de resolver el modelo:

- Se impone que la frecuencia solo puede tomar los valores 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 trenes/hora, y, por lo tanto, el "headway" su inverso correspondiente.
- Tiempo de parada mínimo de 10 segundos en cada estación.
- Tiempo mínimo de seguridad de 60 segundos entre dos trenes consecutivos.
- Tiempo necesario para dar la vuelta en las estaciones finales: 100 segundos.

5.2 Implementación y análisis de los resultados obtenidos

Una vez han sido definidos todos los datos de entrada necesarios para llevar a la práctica el modelo, sólo nos queda dar paso a su implementación. En el anexo final, se puede encontrar todo el código de programación utilizado para obtener los resultados que a continuación se mostrarán. A su vez, dentro del mismo, también se halla una breve explicación de cada una de las funciones y algoritmos que han sido utilizados.

Como ya se mencionó con anterioridad, para llevar a cabo la resolución y el análisis del problema, se van a realizar diferentes pruebas computacionales, las cuales, tendrán como única diferencia la ponderación que se le otorga a los dos criterios considerados en la función objetivo (ponderaciones disponibles en la Tabla 7).

Tras llevar a la práctica el modelo, cabe destacar que, en cada uno de los escenarios, ha sido necesario realizar tres iteraciones hasta alcanzar la convergencia, y con ello, la solución final. Recordemos que esta se alcanzaba cuando no existía diferencia alguna en la carga de pasajeros que soporta cada vía de la red entre dos iteraciones consecutivas. En la resolución computacional, esta diferencia se está comprobando automáticamente, de tal forma que todo el código se encuentra contenido en un bucle y, siempre que exista diferencia alguna de cargas entre la iteración actual y la anterior, el código se vuelve a ejecutar.

Concretamente, de forma resumida, el proceso que se ha seguido en cada escenario ha sido el siguiente:

Inicialmente, se ha realizado una asignación del tránsito teniendo solamente en cuenta las longitudes totales de las diferentes estrategias, lo que implica que los resultados tras la asignación fuesen comunes a los diferentes escenarios. Como ya se explicó anteriormente, esto es debido a que, en esta primera asignación, aún no había sido resuelto el modelo de optimización, por ello, los tiempos de viaje no podían ser calculados. Posteriormente, se han incorporado en el modelo de optimización los datos obtenidos tras la primera asignación (carga de pasajeros en las distintas vías, valor de la carga máxima para cada línea, subidas y bajadas en las plataformas), y en esta ocasión, sí que se han obtenido distintos resultados en cada escenario para las diferentes variables que el modelo proporciona (frecuencia, “headway”, tiempo de ciclo, tamaño de flota, modelo de tren, tiempos de parada), debido a que cada uno de ellos se pondera la función objetivo de una determinada manera diferente al resto. A la hora de realizar la siguiente asignación del tránsito, se tomaron como datos de entrada los resultados obtenidos tras la optimización del modelo, por lo que, en esta nueva ocasión, el tiempo total de viaje fue lo que se utilizó para calcular las nuevas probabilidades de que un pasajero seleccionase una estrategia u otra, pasando a sustituir a las longitudes usadas con anterioridad. De esta forma, se obtuvieron diferentes resultados en esta segunda asignación, cambiando el número de pasajeros que toma cada camino, y como consecuencia

de ello, cambiando las cargas en las distintas vías. A partir de aquí, se siguió realizando el mismo proceso iterativo, hasta que finalmente, tras la tercera iteración, se alcanzó la convergencia.

Con relación al alcance de la convergencia, cabe resaltar que, a lo largo de las diferentes iteraciones, todas las cargas se mantuvieron igual excepto aquellas correspondientes a las vías (1, 2), (2, 3), (3, 4) y (4, 5) y su sentido contrario. De hecho, cabía esperar tales resultados, debido a que, si se observa la Figura 10 correspondiente al plano de la red de cercanías implementada, puede deducirse con facilidad como las diferencias entre cargas, solamente van a poder tener lugar en los arcos (1, 2), (2, 3), (3, 4) y (4, 5), tanto en el sentido principal de la red como en el contrario. Esto es debido a que las únicas estaciones pertenecientes a más de una línea están todas comprendidas en dichos segmentos, por lo tanto, cuando los pasajeros se encuentren en alguna de las estaciones correspondientes, ya sea para iniciar su viaje o trasbordar, van a tener más de una opción disponible (estrategia) para realizar su trayecto, y la elección de una u otra (en el apartado 3.2.3. se explicaba de que dependía tal elección) será lo que provoque que la cantidad de pasajeros que soporten los correspondientes arcos varíe de una iteración a la siguiente. Es decir, imaginemos, por ejemplo, que nos encontramos en una estación cualquiera de las tres líneas escogidas, tal y como está diseñada la topología de la red, si deseásemos viajar hasta una estación perteneciente a otra línea, siempre habría que pasar por alguno de los tramos (1, 2), (2, 3), (3, 4) y (4, 5), y en sus estaciones podríamos tomar la línea 1 o la 2. El tomar una u otra, será lo único que produzca que cambie la carga de pasajeros de sus correspondientes vías, ya que, si suponemos el otro caso posible, un pasajero que desea viajar entre dos estaciones de una misma línea, sea la iteración que sea, realizará siempre el mismo trayecto, mantenerse en la línea, y esto no producirá cambios en la carga de las vías.

Una vez se ha explicado brevemente el proceso seguido hasta alcanzar la convergencia, pasamos a mostrar los costes finales obtenidos en las diferentes pruebas computacionales llevadas a cabo, a partir de los cuales, se ha construido la “Frontera de Pareto” representada en la Figura 17.

Posteriormente, para aquellos puntos de la frontera más representativos, se realizará un breve análisis de los resultados obtenidos, para finalmente, quedarnos con aquel de ellos que, en la mayor medida posible, favorezca tanto a pasajeros como a operadores. No obstante, para no sobrecargar con muchos resultados, se mostrarán solamente los resultados finales del escenario seleccionado.

ESCENARIO	OPERADORES		PASAJEROS		COSTE TOTAL (€)
	PESO	COSTE (€)	PESO	COSTE (€)	
1	0	117.737,591	1	84.091,106	201.828,697
2	1	7.235,820	0	221.905,565	229.141,385
3	1	30.472,775	1	102.774,516	133.247,291
4	1	42.185,263	2	94.972,873	137.158,136
5	1	60.923,465	5	88.761,958	149.685,423
6	1	91.374,155	10	84.091,106	175.465,261
7	2	18.310,167	1	121.457,926	139.768,093
8	5	13.238,572	1	139.370,917	152.609,489
9	10	10.175,835	1	159.097,665	169.273,500
10	1.5	25.803,689	1	108.884,659	134.688,348
11	1	34.219,536	1.5	99.644,247	133.863,783

Tabla 7. Costes obtenidos en cada escenario.

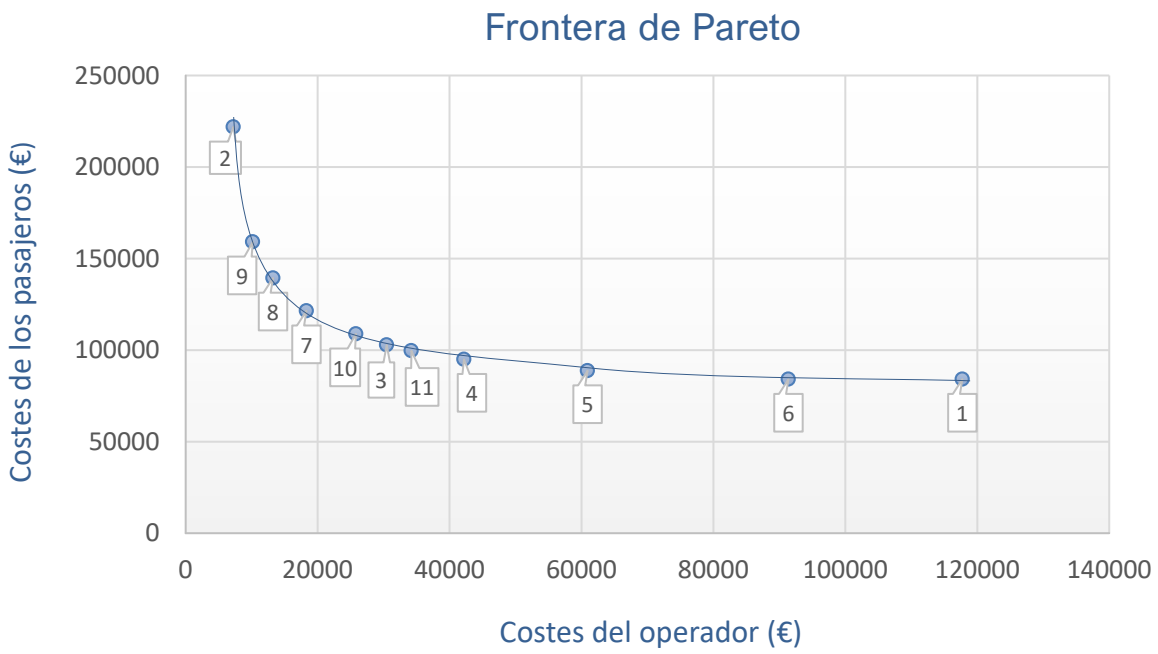


Figura 17. Representación de la Frontera de Pareto.

Como era de esperar, en el Escenario 1 donde solamente se han tenido en cuenta los costes de los pasajeros, se han obtenido las mayores frecuencias posibles para todas las líneas (30 trenes/hora), debido a que, el modelo únicamente ha buscado satisfacer a los pasajeros, proporcionándoles como consecuencia un servicio máximo. Al contrario ocurre en el Escenario 2, donde solamente se han tenido en cuenta los costes de los operadores. En esta ocasión, se han obtenido las mínimas frecuencias que permiten cubrir toda la demanda de la matriz OD, puesto que, en este escenario, no importa que los pasajeros tengan que dedicar largos tiempos a esperar la llegada del tren, sólo se busca minimizar al máximo las frecuencias y tamaños de flota.

Con relación al punto de la frontera que permite satisfacer lo máximo posible el objetivo planteado en este proyecto, en realidad, debido a que no se tiene un criterio previo establecido, no se puede asegurar con total certeza que uno sea superior al resto, por lo que vamos a seguir el siguiente razonamiento para escoger uno de ellos: observando todos los puntos obtenidos, nos vamos a quedar con aquellos cuya suma de ambos costes (operadores y pasajeros) sea inferior al resto, en concreto, existen 3 de ellos que tienen una suma inferior a 135.000 €, el resto de puntos supera esta cantidad. Cabe resaltar que no solo estos 3 puntos (correspondientes a los escenarios 3, 10 y 11) son los que tienen los costes totales más bajos, sino que, además, como se puede ver en la Figura 17, son puntos que cumplen adecuadamente el objetivo de satisfacer tanto a pasajeros como a operadores, es por ello por lo que podría tomarse como resultado final cualquiera de ellos, y sería una decisión correcta. No obstante, para finalmente seleccionar uno de ellos, no escogeremos uno aleatoriamente, sino que, como ya sabemos que cualquiera de los puntos proporcionaría una calidad del servicio a los pasajeros adecuada, vamos a buscar aquel con menores costes para los operadores, debido a que es el único criterio que realmente supone un coste monetario como tal (recordemos que los costes de los pasajeros eran una conversión del tiempo a dinero).

Por lo tanto, finalmente vamos a seleccionar como solución al problema planteado los valores obtenidos para el escenario 10, los cuales, se muestran en la siguiente tabla y se comentan de forma breve posteriormente:

	LÍNEA 1	**LÍNEA 2**	**LÍNEA 3**
FRECUENCIA (trenes/h)	10	6	10
HEADWAY (s)	360	600	360
TIEMPO DE CICLO (s)	4320	6000	5400
TAMAÑO DE FLOTA	12	10	15
MODELO DE TREN	Serie 462	Serie 462	Serie 462

Tabla 8. Resultados finales del modelo de optimización para el escenario 10.

Como se puede ver en la Tabla 8, se han obtenido frecuencias relativamente altas que permiten el paso de un tren cada 6 minutos para las líneas 1 y 3, y cada 10 minutos para la línea 2. Tales resultados, permiten reafirmar el logro de uno de nuestros objetivos, la alta calidad en el servicio ofertado a los usuarios.

Respecto al modelo de tren, puede observarse como para todas las líneas ha sido seleccionado el mismo, el civia Serie 462. Este se corresponde con el modelo de menor capacidad, y si se analiza detenidamente, es totalmente razonable su elección. Recordemos del modelo matemático propuesto que, la capacidad del tipo de tren seleccionado para cada línea por su correspondiente frecuencia debía ser suficiente para atender la carga máxima que tuviese la línea en cuestión. Tal y como se puede observar en la Tabla 9 correspondiente al número de pasajeros que soporta cada arco durante todo el horizonte temporal considerado en la matriz OD, la carga máxima (tramo con mayor número de pasajeros) para cada línea se corresponde con los valores 1011, 1124, 1261 para las líneas 1, 2 y 3, respectivamente. La capacidad del modelo civia 462, que recordemos era de 414 pasajeros, por la frecuencia obtenida para cada una de las líneas (Tabla 8), es superior a la carga máxima, por lo tanto, debido a que se buscan minimizar costes, no hay necesidad alguna de seleccionar un modelo con capacidad superior.

LÍNEA 1		**LÍNEA 2**		**LÍNEA 3**	
SENTIDO PRINCIPAL					
Arco	Carga	Arco	Carga	Arco	Carga
(1, 2)	668	**(1, 2)**	748	**(1, 24)**	1190
(2, 3)	710	**(2, 3)**	784	**(24, 25)**	1184
(3, 4)	734	**(3, 4)**	827	**(25, 26)**	1150
(4, 5)	724	**(4, 5)**	824	**(26, 27)**	1085
(5, 6)	1011	**(5, 13)**	1121	**(27, 28)**	1061
(6, 7)	924	**(13, 14)**	1065	**(28, 29)**	1013
(7, 8)	797	**(14, 15)**	1016	**(29, 30)**	1014
(8, 9)	637	**(15, 16)**	931	**(30, 31)**	1030
(9, 10)	477	**(16,17)**	861	**(31, 32)**	1025
(10, 11)	397	**(17, 18)**	771	**(32, 33)**	997
(11, 12)	281	**(18, 19)**	666	**(33, 34)**	939
		(19, 20)	410	**(34, 35)**	885
		(20, 21)	308	**(35, 36)**	821
		(21, 22)	190	**(36, 37)**	757
		(22, 23)	91	**(37, 38)**	674
				(38, 39)	595
				(39, 40)	482
				(40, 41)	378

LÍNEA 1		LÍNEA 2		LÍNEA 3	
SENTIDO CONTRARIO					
Arco	Carga	Arco	Carga	Arco	Carga
(12, 11)	254	**(23, 22)**	49	**(41, 40)**	323
(11, 10)	374	**(22, 21)**	103	**(40, 39)**	469
(10, 9)	529	**(21, 20)**	152	**(39, 38)**	596
(9, 8)	637	**(20, 19)**	225	**(38, 37)**	689
(8, 7)	780	**(19, 18)**	423	**(37, 36)**	805
(7, 6)	893	**(18, 17)**	553	**(36, 35)**	831
(6, 5)	927	**(17, 16)**	670	**(35, 34)**	885
(5, 4)	633	**(16, 15)**	800	**(34, 33)**	906
(4, 3)	636	**(15, 14)**	904	**(33, 32)**	960
(3, 2)	652	**(14, 13)**	1024	**(32, 31)**	997
(2, 1)	619	**(13, 5)**	1124	**(31, 30)**	1041
		(5, 4)	790	**(30, 29)**	1104
		(4, 3)	769	**(29, 28)**	1129
		(3, 2)	757	**(28, 27)**	1128
		(2, 1)	722	**(27, 26)**	1152
				(26, 25)	1158
				(25, 24)	1183
				(24, 1)	1261

Tabla 9. Carga obtenida tras la asignación final para cada tramo de la red.

Con relación al tiempo de parada en las diferentes estaciones, sin más dilación, cabe resaltar que la estación número 1 (Valencia Nord) ha resultado ser para todas las líneas, aquella en la que los trenes se detienen un mayor tiempo. Tal resultado era de esperar, debido a que en la matriz de demanda OD se estableció que, tal y como ocurría en la realidad, esta fuese la estación en la que mayor número de pasajeros inician y finalizan trayecto, por lo tanto, los trenes en dicha estación necesitan detenerse un tiempo mayor para que todos los pasajeros dispongan del tiempo suficiente para subir y bajar. Además, a esto se le suma el hecho de que la estación número 1 es la única que conecta las tres líneas, por lo tanto, también ha resultado ser la más utilizada para realizar trasbordos.

Finalmente, con ayuda del paquete “Matplotlib”, pasamos a representar gráficamente los resultados obtenidos para cada una de las líneas con las que se ha trabajado, de tal forma que esto nos facilite su comprensión.

En las gráficas que se muestran a continuación, aparecen representadas en el “eje y” mediante líneas horizontales, las diferentes plataformas que componen cada línea. A su vez, la separación entre dichas líneas, muestra la distancia existente entre cada una de las plataformas. Esto nos permite, que la pendiente de cada una de las líneas del recorrido equivalga a la velocidad que lleva el tren en dicho tramo (ver Figura 7 para una mejor comprensión). Por otro lado, tenemos el “eje x”, donde se muestra el tiempo que la correspondiente línea tarda en recorrer las distintas vías que unen cada par de plataformas en los dos sentidos, ida y vuelta. Se ha escogido representar el recorrido solamente para un intervalo de una hora, ya que es con el que hemos trabajado a lo largo de todo el problema, aunque si se siguiese representando el recorrido un tiempo mayor, se podría ver como finalizan su recorrido de vuelta los distintos trenes.

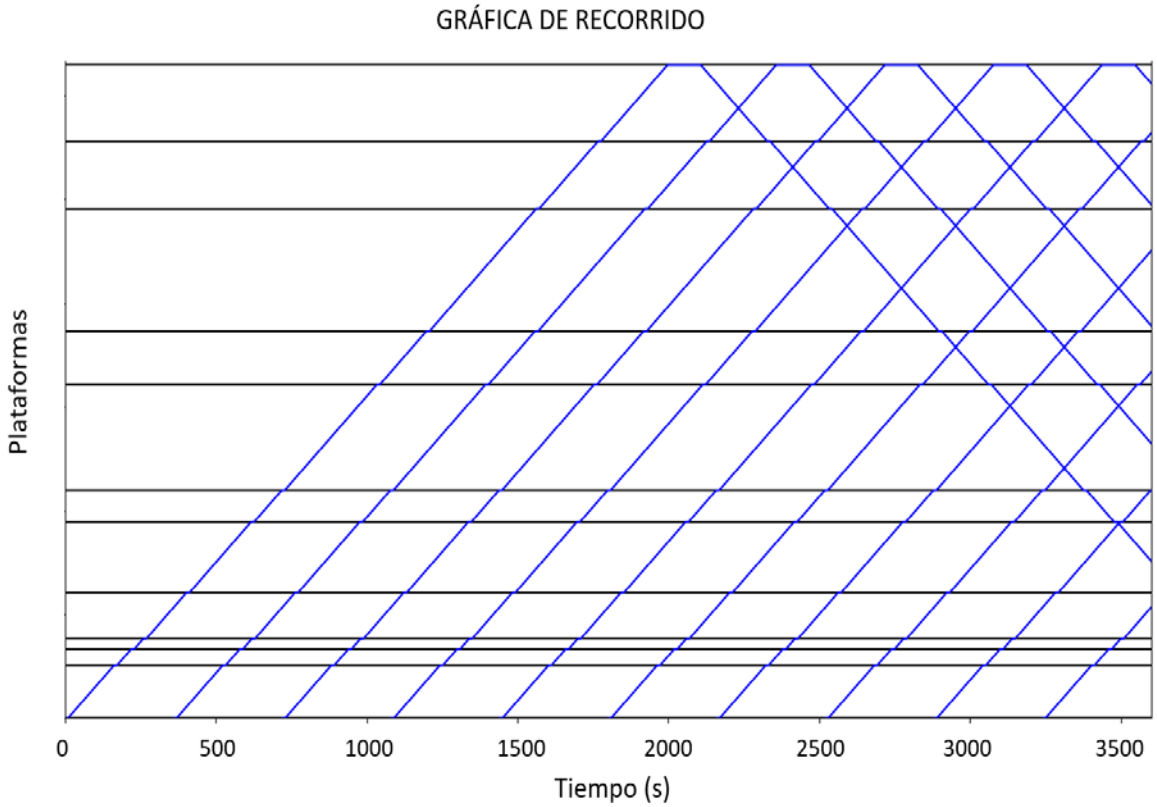


Figura 18. Gráfica del recorrido realizado por la línea 1.

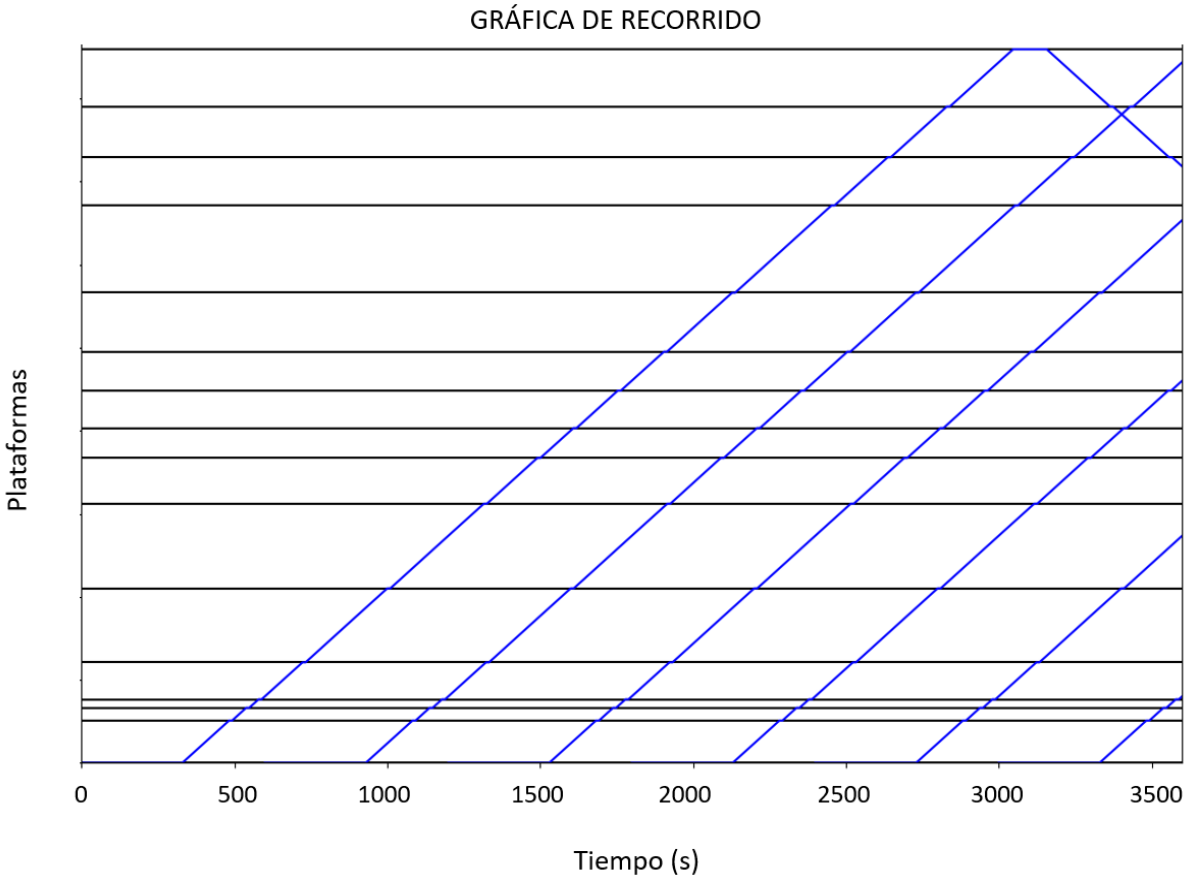


Figura 19. Gráfica del recorrido realizado por la línea 2.

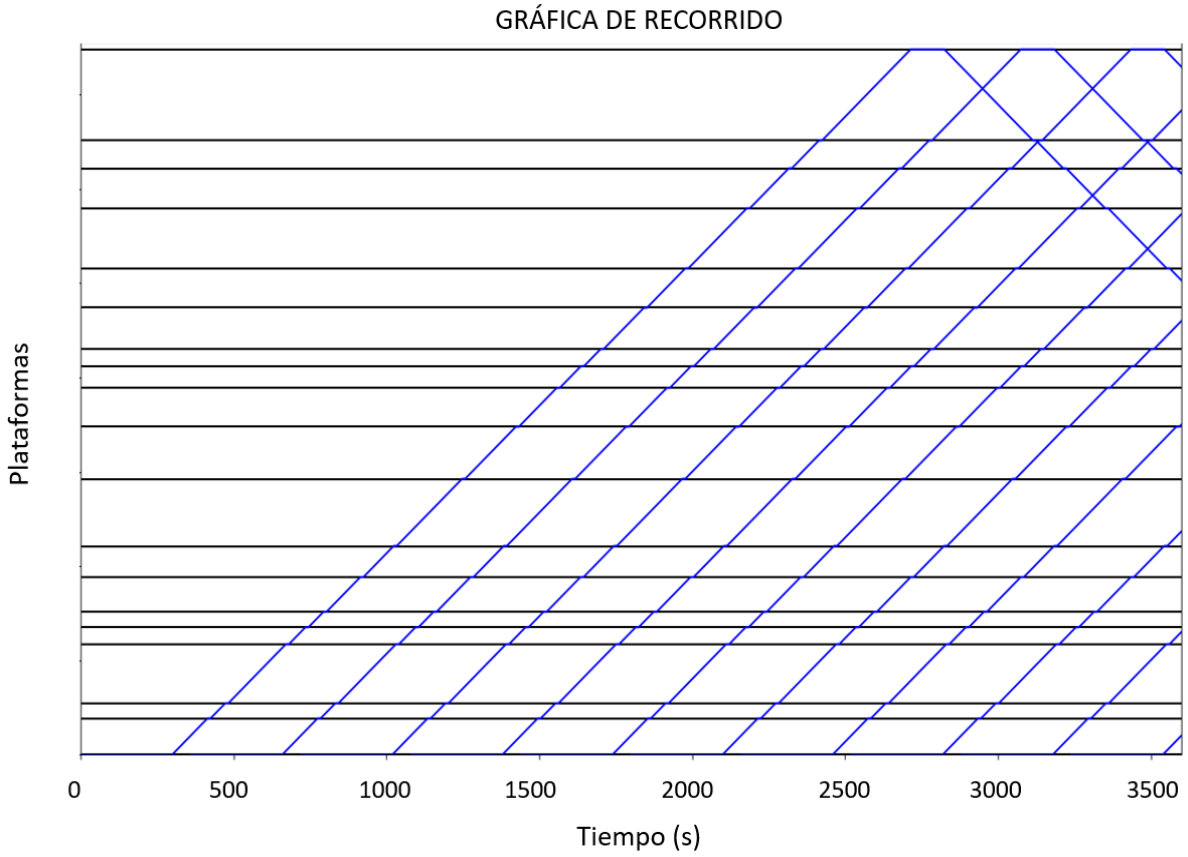


Figura 20. Gráfica del recorrido realizado por la línea 3.

Tal y como se puede observar en las diferentes figuras, se representa también el tiempo de parada obtenido para cada estación. En la mayoría de ellas, este toma el valor de los 10 segundos que se impusieron como tiempo de parada mínimo, sin embargo, se puede apreciar claramente como en la línea 2 y 3, ocurre lo ya mencionado con anterioridad, el tiempo de parada en la primera plataforma (estación Valencia Nord), es muy superior al resto. A su vez, en la estación final aparece también representado el tiempo necesario para realizar el cambio de sentido.

Además, la gráfica muestra el recorrido realizado no solo por un tren, sino por todos los trenes que parten de la primera estación durante horizonte temporal considerado (1 hora), es decir, se observan tantos recorridos como trenes indique la frecuencia obtenida, siendo obviamente el intervalo de tiempo entre cada uno de ellos el correspondiente “headway”.

6. CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO

En vista a los resultados obtenidos, puede afirmarse la consecución de los objetivos planteados inicialmente. Tras la necesidad de realizar ciertas linealizaciones en el modelo matemático propuesto, finalmente, el modelo de programación lineal planteado y su resolución mediante lenguaje de programación Python, han sido más que adecuados para apoyar la toma de decisiones en una de las fases de la planificación ferroviaria. Concretamente, este ha permitido determinar las diferentes variables que son necesarias para poder programar una red de cercanías en el horizonte temporal considerado.

Al mismo tiempo, la resolución de distintos escenarios, en los cuales, se ha ido variando la ponderación otorgada a los dos objetivos (pasajeros y operadores), ha permitido obtener una representación de la Frontera de Pareto y con ello, poder analizar los diferentes puntos de ésta hasta obtener una solución que permite satisfacer en la mayor medida posible tanto a operadores como a pasajeros. Además, debido a que se ha logrado alcanzar la convergencia en cada uno de los escenarios planteados, se puede concluir que, en base a las características y exigencias de un sistema como el que se ha planteado, los resultados obtenidos han sido los óptimos dado el objetivo establecido. Adicionalmente, se ha podido comprobar la gran sensibilidad que el modelo presenta ante cambios en el objetivo, y es que, tan solo una pequeña variación en los pesos dados a los dos tipos de costes, provoca grandes modificaciones en los resultados.

Por el contrario, cabe resaltar que el objetivo del trabajo nunca fue reflejar la situación que tiene lugar actualmente en la red de cercanías de Valencia, es por ello por lo que no se ha realizado ninguna comparación entre los datos obtenidos y los existentes en la realidad, ya que, además, se han ido realizando múltiples suposiciones a lo largo de la implementación que no dan pie a ella.

Tal y como ha sido explicado, se han obtenido valores bastante razonables para las frecuencias, modelos de tren, tiempos de parada y demás variables requeridas. Junto con ello, también se ha obtenido en todo momento una coherente asignación del tránsito, obteniendo para las cargas de pasajeros en las distintas vías aquellos resultados esperados.

A pesar de que el diseño del código ha resultado la parte más compleja del proyecto, finalmente, se ha logrado disponer de una herramienta fácil de utilizar a la hora de la obtención de los resultados, sin necesidad de poseer amplios conocimientos de programación. Es suficiente con introducir los datos de entrada requeridos para que el programa proporcione la solución.

Por otro lado, no debemos olvidar que este proyecto cuenta con unos límites de tiempo que implican tener que realizar un trabajo acorde a ellos. Sin embargo, el campo de la planificación ferroviaria ofrece una gran cantidad de posibilidades no abarcadas. Por lo tanto, el presente trabajo podría servir como base para en un futuro realizar múltiples proyectos de mayor envergadura y dificultad.

A continuación, se muestran algunas de las posibles líneas de trabajo que podría resultar de interés llevar a cabo en un futuro proyecto:

- Aplicación de curvas de consumo energético dependientes de carga y velocidad, en lugar de suponer un consumo constante de los trenes a lo largo de todo el trayecto.
- Implementación de tramos de vías compartidos por más de una línea, de tal forma que esto supondría tener que coordinar las frecuencias entre las diferentes líneas implicadas para un correcto funcionamiento del sistema.
- Establecer una demanda variable a lo largo del tiempo.

7. BIBLIOGRAFÍA

Abbink, E.; Fischetti, M.; Kroon, L.; Timmer, G. & Vromans, M., 2005. Reinventing crew scheduling at Netherlands Railways. *Interfaces*, 35, pp.393–401.

Akyol, E., 2017. *Train Platforming Problem Solving*. PhD Thesis, University of Pardubice.

Aranda, J. A. & Orjuela, J. A., 2015. Optimización multiobjetivo en la gestión de cadenas de suministro de biocombustibles. Una revisión de la literatura. Ingeniería, 20 (1), pp. 37–63

Arenas, D.; Chevrier, R.; Hanafi, S. & Rodriguez, J., 2015. Solving the Train Timetabling Problem, a mathematical model and a genetic algorithm solution approach. In *6th International Conference on Railway Operations Modelling and Analysis*, Tokyo, Japon.

Balas, E. & Carrera, M.C., 1996. A dynamic subgradient-based branch-and-bound pocedure for set covering. *Operations Research*, 44, pp.875–890.

Berbey, A.; Caballero, R.; Bobi, J. S.; Brunel, J.; Guerra, K.; Flores, J.; Samaniego, A. & Orozco, W., 2013. Trenes: material rodante del transporte ferroviario. *Prisma Tecnológico*, 4(1).

Bianco, L.; Bielli, M.; Mingozzi, A.; Ricciardelli, S. & Spadoni, M., 1992. A Heuristic Procedure for the Crew Rostering Problem. *European Journal of Operational Research*, 58(2), pp.272–283.

Bruno, G.; Gendreau, M. & Laporte, G., 2002. A Heuristic for the Location of a Rapid Transit Line. *Computers & Operations Research*, 29, pp.1–12.

Bruno, G.; Ghiani, G. & Improta, G., 1998. A Multi-modal Approach to the Location of a Rapid Transit Line. *European Journal of Operational Research*, 104, pp.321–332.

Bussieck, M., 1998. *Optimal lines in public rail transport*. PhD thesis, Technische Universität Braunschweig.

Bussieck, M.R.; Winter, T. & Zimmermann, U., 1997. Discrete optimization in public rail transport. *Mathematical Programming*, 79(1-3), pp.415–444.

Bussieck, M.R.; Lindner, T. & Lübbecke, M.E., 2004. A fast algorithm for near cost optimal line plans. *Mathematical Methods of Operations Research*, 59, pp.205–220.

Cacchiani, V., 2008. Models and Algorithms for Combinatorial Optimization Problems arising in Railway Applications. *A Quarterly Journal of Operations Research*, 7(1), pp.109–112.

Cai, X. & Goh, C.J., 1994. A Fast Heuristic for the Train Scheduling Problem. *Computers and Operations Research*, 50, pp.499–510.

Canca, D. & Zarzo, A., 2017. Design of energy-Efficient timetables in two-way railway rapid transit lines. *Transportation Research Part B: Methodological*, 102, pp.142–161.

Canca, D.; Andrade-Pineda, J. L.; De los Santos, A. & Calle, M., Forthcoming. The Railway Rapid Transit frequency setting problem with speed-dependent operation costs. *Transportation Research Part B: Methodological.*

Caprara, A.; Fischetti, M.; Toth, P.; Vigo, D. & Guida, P., 1997. Algorithms for railway crew management. *Mathematical Programming*, 79, pp.125–141.

Caprara, A.; Focacci, F.; Lamma, E.; Mello, P.; Milano, M.; Toth, P. & Vigo, D., 1998. Integrating Constraint Logic Programming and Operations Research Techniques for the Crew Rostering Problem. *Software Practice and Experience*, 28, pp.49–76.

Caprara, A.; Kroon, L.; Monaci, M.; Peeters, M. & Toth, P., 2007. Passenger Railway Optimization. *Handbooks in Operations Research and Management Science*, 14, pp.129–187.

Caprara, A.; Fischetti, M. & Toth, P., 2002. Modeling and solving the train timetabling problem. *Operations Research*, 50(5), pp.851–861.

Cardaso, L. & Marín, A., 2012. Integration of timetable planning and rolling stock in rapid transit networks. *Annals of Operation Research*, 199, pp.113–135.

Carey, M. & Lockwood, D., 1995. A model, algorithms and strategy for train pathing. *Operations Research Society*, 46(8), pp.988–1005.

Carey, M. & Carville, S., 2003. Scheduling and platforming trains at busy complex stations. *Transportation Research*, 37, pp.195–224.

Carraresi, P. & Gallo, G., 1984. A multilevel bottleneck assignment approach to the bus driver's rostering problem. *European journal of operational research*, 16(2), pp.163–173.

Ceder, A., 1984. Bus frequency determination using passenger count data. *Transportation Research –Part A*, 18(5-6), pp.439–453.

Dienst, H., 1978. Linienplanung im spurgeführten Personenfernverkehr mit Hilfe eines heuristischen Verfahrens. PhD thesis, Braunschweig University of Technology, Germany.

Dufourd, H.; Gendreau, M. & Laporte, G., 1996. Locating a transit line using tabu search. *Location Science*, 4(1), pp.1–19.

Ernst, A.; Jiang, H.; Krishnamoorthy, M.; Nott, H. & Sier, D., 2001. Rail Crew Scheduling and Rostering Optimization Algorithms. *Lecture Notes in Economics and Mathematical Systems*, 595, pp.53–71.

Ernst, A. & Dowling, D., 1998. Train crew rostering using simulated annealing. *Proceedings of International Conference on Optimization Techniques and Application*, pp.859–866.

Freling, R.; Lentink, R.; Kroon, L. & Huisman, D., 2005. Shunting of Passenger Train Units in a Railway Station. *Transportation Science*, 39(2), pp.261–272.

Freling, R.; Lentink, R.M. & Wagelmans, A.P., 2004. A decision support system for crewplanning in passenger transportation using a flexible branch-and-price algorithm. *Annals of Operations Research*, 127, pp.203–222.

Friedrich, M., 1994. Computer assisted design of public transport systems in rural areas. *In Schriftenreihe des Lehrsstuhls für Verkehrs- und Stadtplanung*, 5.

Friedrich, M. & Wekeck, S., 2004. A schedule-based Transit Assignment Model addressing the Passengers' Choice among competing Connections. *Schedule-Based Dynamic Transit Modeling: theory and applications*, 28, pp.159–173.

Glover, F., 1989. "Tabu Search – Part 1". *ORSA Journal on Computing*, 1 (2), pp. 190–206.

Glover, F., 1990. "Tabu Search – Part 2". ORSA Journal on Computing, 2 (1), pp. 4–32.

Goossens, J.W. & Hoesel, S.V., 2004. A Branch-and-Cut Approach for Solving Railway Line-Planning Problems. *Transportation science*, 38(3), pp.379–393.

Guihaire, V. & Hao, J.K., 2008. Transit Network Design and Scheduling: A Global Review. *Transportation Research Part A: Policy and Practice*, 42(10), pp.1251–1273.

Haijema, R.; Duin, C. & van Dijk, N.M., 2006. Train Shunting: A Practical Heuristic Inspired by Dynamic Programming. *Planning in Intelligent Systems: Aspects, Motivations, and Methods*, pp.437–477.

Hamdouch, Y. & Lawphongpanich, S., 2008. Schedule-based transit assignment model with travel strategies and capacity constraints. *Transportation Research Part B*, (42),

pp.663–684.

Han, A.F. & Wilson, N.H.M., 1982. The allocation of buses in heavily utilized networks with overlapping routes. *Transportation Research – Part B*, 16(3), pp.221–232.

Herbon, A. & Hadas, Y., 2015. Determining optimal frequency and vehicle capacity for public transit routes: A generalized newsvendor model. *Transportation Research Part B*, 71, pp.85–99.

Heydar, M.; Petering, M.E.H. & Bergmann, D.R., 2013. Mixed integer programming for minimizing the period of a cyclic railway timetable for a single track with two train types. *Journal Computers and Industrial Engineering*, 66(1), pp.171–185.

Higgins, A. & Kozan, E., 1998. Modeling train delays in urban networks. *Transportation science*, 32(4), pp.346–357.

Hirai, C.; Tomii, N.; Tashiro, Y.; Kondou, S. & Fujimori, A., 2006. An algorithm for train rescheduling using rescheduling pattern description language R. *Proceedings of the COMPRAIL*, pp.551–561.

Hu, S.R. & Liu, C.T., 2014. Optimizing Headways for Mass Rapid Transit Services. *Journal of Transportation Engineering*, 11(140).

Jovanovic, D. & Harker, P.T., 1991. Tactical Scheduling of Rail Operations: the SCAN I System. *Transportation Science*, 25, pp.46–64.

Kinder, M., 2008. *Models for periodic timetabling*. PhD Thesis, Technische Universität Berlin.

Kirkpatrick, S.; Gelatt Jr, C.D.; Vecchi, M.P. (1983). "Optimization by Simulated Annealing". *Science*. 220 (4598): 671–680.

Kroon, L.G. & Peeters, L.W.P., 2003. A variable trip time model for cyclic railway timetabling. *Transportation Science*, 37, pp.198–212.

Kroon, L.G.; Romeijn, H.E. & Zwaneveld, P.J., 1997. Routing trains through railway stations complexity issues. *European journal of operational research*, 98, pp.485–498.

Larson, R.C. & Odoni, A.R., 1981. *Urban Operations Research*, Prentice-Hall.

Lindner, T. & Zimmermann, U., 2000. *Train schedule optimization in public rail transport*, Technical Report, Ph, Technische Universitat.

Lingaya, N.; Cordeau, J.; Desaulniers, G.; Desrosiers, J. & Soumis, F., 2002. Operational

car assignment at VIA Rail Canada. *Transportation Research*, 36, pp.755–778.

López-Ramos, F.; Codina, E.; Marín, A. & Guarnaschelli, A., 2017. Integrated approach to network design and frequency setting problem in railway rapid transit systems. *Computers & Operations Research*, 80, pp.128–146.

De Luca Cardillo, D. & Mione, N., 1998. k L-list tau coloring of graphs. *European journal of operational research*, 106, pp.160–164.

Maróti, G., 2006. *Operations research models for railway rolling stock planning*. PhD Thesis, Technische Universiteit Eindhoven.

Mazo, R., 2011. *A Generic Approach for Automated Verification of Product Line Model*. PhD Thesis, Université Panthéon-Sorbonne, Paris.

Mistry, P. & Kwan, R.S.K., 2003. Generation and Optimization of Train Timetables using coevolution. *Lecture Notes in Computer Science*, pp.693–694.

Montoro, A.F., 2012. *Python 3 al descubierto*, RC libros.

Nachtigall, K., 1994. *A branch and cut approach for periodic network programming*. Technical Report 29, Hildesheimer Informatik-Berichte.

Nachtigall, K. & Voget, S., 1996. A genetic algorithm approach to periodic railway synchronization. *Computers and Operations Research*, 23(5), pp.453–463.

Nuzzolo, A., 2003. Transit path choice and assignment model approaches. *in William H. K. Lam, Michael G. H. Bell (ed.). Advanced Modelling for Transit Operations and Service Planning*, pp.93–124.

Odijk, M.A., 1996. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research*, 30(6), pp.455–464.

Orro, A., 2006. Planificación de sistemas ferroviarios metropolitanos. *Ingeniería y territorio*, 76, pp.18–23.

Pashchenko, F.; Kuznetsov, N.; Ryabykh, N.; Minashina, I.; Zakharova, E. & Tsvetkova, O., 2015. Implementation of Train Scheduling System in Rail Transport using Assignment Problem Solution. *Procedia Computer Science*, 63, pp.154-158

Patz, A., 1925. Die richtige Auswahl von Verkehrslinien bei großen Straßenbahnnetzen. *Verkehrstechnik*, pp.50-51.

Peeters, L.W.P., 2003. *Cyclic railway timetable optimizatrion*. PhD thesis, Erasmus University Rotterdam.

Poon, M.H.; Wong, S.C. & Tong, C.O., 2004. A dynamic schedule-based model for congested transit networks. *Transportation Research Part B*, 38(4), pp.343–368.

Robenek, T.; Azadeh, S.; Maknoon, Y. & Bierlaire, M., 2017. Hybrid Cyclicity: Combining The Benefits Of Cyclic And Non-Cyclic Timetables. *Transportation Research Part C: Emerging Technologies*, 75, pp.228–253.

Robenek, T.; Maknoon, Y.; Azadeh, S.; Chen, J. & Bierlaire, M., 2016. Passenger centric train taimetabling problem. *Transportation Research Part B: Methodological*, 89, pp.107–126.

Scheele, S., 1980. A supply model for public transit services. *Transportation Research – Part B*, 14, pp.133–146.

Schmöcker, J.D. & Bell, M.G., 2009. The build-up of capacity problems during the peak hour. *In Schedule-Based Modeling of Transportation Networks: Theory and Applications*, pp.217–239.

Scholl, S., 2005. *Customer-oriented line planning*. PhD thesis, University of Kaiserslautern.

Schrijver, A., 1993. Minimum circulation of railway stock. *CWI Quarterly*, 6, pp.205–217.

Serafini, P. & Ukovich, W., 1989. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics 2*, pp.550–581.

Shakibayifar, M.; Sheikholeslami, A.; Corman, F. & Hassannayebi, E., 2017. An integrated rescheduling model for minimizing train delays in the case of line blockage. *Operational Research*, pp.1–29.

Spiess, H. & Florian, M., 1989. Optimal strategies: a new assignment model for transit networks. *Transportation Research Part B.*, 23, pp.83–102.

Di Stefano, G. & Love Koci, M., 2004. A Graph Theoretical Approach To The Shunting Problem. *Electronic Notes in Theoretical Computer Science*, 92, pp.16–43.

Szeto, W.Y. & Jiang, Y., 2014. Transit route and frequency design: Bi-level modeling and hybrid artificial bee colony algorithm approach. *Transportation Research Part B Methodological*, 67, pp.235–263.

Szpigel, B., 1973. Optimal Train Scheduling on a Single Line Railway. *Operations Research*, 72, pp.343–351.

Tomii, N.; Zhou, L.J. & Fukumura, N., 1999. An Algorithm for Station Shunting Scheduling Problems Combining Probabilistic Local Search acand PERT. *Lecture Notes in Computer Science*, 119(3), pp.29–34.

Wardman, M.; Shires, J.; Lythgoe, W. & Tyler, J., 2004. Consumer benefits and demand impacts of regular train timetables. *International Journal of Transport Management 2*, pp.39–49.

Wardman, M., 2004. Public Transport Values of Time. *Transport Policy*, 11(4), pp.363–377.

Xiong, Y. & Schneider, J., 1992. Transportation network design using a cumulative genetic algorithm and neural network. *Transportation Research Record*, 1364, pp.37–44.

Zwaneveld, P.; Kroon, L. & Ambergen, H., 1996. A decision support system for routing trains through railway stations. Proceedings of Comp Rail, 96, pp.217–226.

REFERENCIAS DE PÁGINAS WEB

Costes de servicios del ferrocarril. [7 de junio 2018] Obtenido de http://ferropedia.es/wiki/Costes_del_ferrocarril:_servicios

Definición Sistemas de Tránsito Rápido. [16 de mayo 2018] Obtenido de https://www.britannica.com/technology/rapid-transit

Demanda del servicio ferroviario en hora punta. [4 de junio 2018] Obtenido de https://transportpublic.org/es/divulgacio-ptp-com-calculem-locupacio-al-transport-public/

Demanda diaria por líneas cercanías Valencia. [6 de mayo 2018] Obtenido de http://www.fomento.es/NR/rdonlyres/0110af44-8f9a-4c13-892e-7b66c7c91ac7/71621/100420_pres_1.pdf

Descarga Gurobi. [15 de enero 2018] Obtenido de www.gurobi.com

Descarga Anaconda Versión 5.2. [12 de diciembre 2018] Obtenido de https://www.anaconda.com/download/

Descarga Pycharm Community Edition. [12 de diciembre 2018] Obtenido de https://www.jetbrains.com/pycharm/download/#section=windows

Distancia entre estaciones cercanías Valencia. [29 de mayo 2018] Obtenido de https://es.wikipedia.org/wiki/Cercanías_Valencia

Frecuencias línea 4 cercanías Madrid. [12 de marzo 2018] Obtenido de https://en.redtransporte.com/madrid/cercanias-renfe/linea-c-4.html

Frecuencias del servicio de cercanías Valencia. [30 de mayo 2018] Obtenido de https://www.redtransporte.com/valencia/cercanias-renfe/

Implantación del maquinista único. [1 de junio 2018] Obtenido de https://elpais.com/diario/2000/03/16/economia/953161239_850215.html

Información técnica trenes civia. [30 de mayo 2018] Obtenido de http://www.renfe.com/viajeros/nuestros_trenes/civia.html

Plano cercanías de Valencia. [28 de mayo 2018] Obtenido de http://www.renfe.com/viajeros/cercanias/planos/valencia.html

Recorrido líneas cercanías Valencia. [29 de mayo 2018] Obtenido de http://www.renfe.com/viajeros/cercanias/valencia/lineas/index.html

Tutorial Networkx. [15 de enero 2018] Obtenido de https://networkx.github.io

Tutorial Matplotlib. [3 de junio 2018] Obtenido de https://matplotlib.org/users/index.html

8. ANEXOS

```
# -*- coding: utf-8 -*-
from networkx import *
from itertools import islice
from gurobipy import *
from matplotlib import pyplot


"DATOS NECESARIOS PARA DEFINIR UNA RED (EN FORMA DE LISTA) PROPORCIONADOS POR EL USUARIO"

"Se definen los nodos correspondientes a las estaciones que conforman la red"
Nodos = [i for i in range(41)]

"Se definen las aristas correspondientes a los tramos de vía que unen una estación con otra"
Aristas =[(0,1),(1,2),(2,3),(3,4),(4,5),(5,6),(6,7),(7,8),(8,9),(9,10),(10,11),(4,12),(12,13),(13,14),(14,15),(15,16),
(16,17),(17,18),(18,19),(19,20),(20,21),(21,22),(0,23),(23,24),(24,25),(25,26),(26,27),(27,28),(28,29),(29,30),(30,31,
(31,32),(32,33),(33,34),(34,35),(35,36),(36,37),(37,38),(38,39),(39,40)]

"Se definen las longitudes en metros de las aristas anteriores"
LongAristas = [5110,1530,1020,4460,6800,3040,10160,5160,11820,6510,7310,8900,10200,5500,3600,4500,4700,7100,10500,
              5800,6100,6900,3820,1610,6270,1800,1630,3690,3290,7130,5540,4120,2300,1810,4400,4180,6370,4230,2960,9670]

"Se definen los nodos que corresponden a cada una de las 3 líneas"
NodosL1 = [0,1,2,3,4,5,6,7,8,9,10,11]
NodosL2 = [0,1,2,3,4,12,13,14,15,16,17,18,19,20,21,22]
NodosL3 = [0,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40]

"Se definen las aristas que corresponden a cada una de las 3 lineas"
AristasL1 = [(0,1),(1,2),(2,3),(3,4),(4,5),(5,6),(6,7),(7,8),(8,9),(9,10),(10,11)]
AristasL2 = [(0,1),(1,2),(2,3),(3,4),(4,12),(12,13),(13,14),(14,15),(15,16),(16,17),(17,18),(18,19),(19,20),(20,21),
              (21,22)]
AristasL3 = [(0,23),(23,24),(24,25),(25,26),(26,27),(27,28),(28,29),(29,30),(30,31),(31,32),(32,33),(33,34),(34,35),
              (35,36),(36,37),(37,38),(38,39),(39,40)]
```

```
"""Se define la Matriz OD en forma de lista. En principio tiene tantas filas y columnas como nodos, ordenadas según
código de nodo. Dicha matriz, multiplicada por el factor de escala, se corresponde con las personas que desean viajar
desde un nodo i a uno j, para un período de una hora"""

L_OrigenDestino=[[0,43,11,20,34,5,21,31,10,15,18,30,20,11,22,5,9,4,18,7,8,7,4,6,6,5,10,5,3,9,2,6,4,3,4,5,3,5,6,5,41],
                 [32,0,10, 9,27,13,12,11,15,6, 5,25, 4,4, 5,10,3,8,26,4,5,6,4,5,3,5,4,5,2,3,4,2,3,5,4,2,4,3,4,5,30],
                 [20,14,0, 6,10,5, 9,15, 9, 1, 4,12,11,4,3,5,6,9,7,4,10,9,2,4,8,12,10,1,4,1,3,1,4,5,1,0,3,1,2,3,13],
                 [21,14, 5, 0,10,3,5, 8, 7, 1, 5,10,3,2,1,1,2,1,14, 4,6,5, 2, 3,5, 8,6,1,1,3,1,4,3,1,2,5,1,2,4,0,5],
                 [30,12,10,5, 0,4,15,15,10, 4, 5,13,9,2,5,7,9,9,15, 5,2,5, 5,6,5,15,10,1,4,5,2,3,2,1,4,1,0,4,2,1,9],
                 [ 9, 2, 1,5, 2,0, 3, 1, 4, 0, 2, 1,1,2,3,1,0, 1,3, 1,0,2, 1, 0,2, 0,0,2,0,0,0,1,3,2,0,1,0,2,0,0,4],
                 [18,15, 4, 9,10,3,0, 5, 7, 8, 2,11,3,5,2,3,1, 3,9, 3,1,2, 4, 2,1,3,1,10,1,1,0,2,1,6,2,1,3,1,6,1,5],
                 [15,11, 4, 5, 8,3,11,0, 9, 4, 5, 8,3,5,3,1,9, 9,4, 2,5,8, 4, 2,9, 8,5,2,3,0,1,2,1,2,1,1,2,3,0,7,9],
                 [ 7, 9, 5, 6,10,1, 6,5, 0, 4, 3, 5,9,2,3,4,3, 2,6, 5,3,1, 2, 1,2, 5,3,2,1,6,3,1,0,2,4,2,1,4,3,0,4],
                 [10,13, 5, 9, 7,4, 5,7, 8, 0, 3, 3,7,4,5,2,7, 3,7, 1,1,2, 1, 4,1, 3,2,3,1,0,5,2,1,2,1,4,8,0,5,4,7],
                 [16,11, 9, 5,10,3, 2, 5, 7,1, 0, 4,3,1,3,2,1, 1,9, 1,3,1, 2, 1,1, 0,2,1,3,1,1,0,1,2,1,1,2,1,2,5,4],
                 [19,15, 7, 7,21,3, 5, 8,10,5, 5, 0,6,9,7,5,9, 5,19, 9,8,2,7, 5,8, 2,2,4,5,1,3,1,3,2,4,1,4,6,3,0,9],
                 [10, 7, 6,15,15,5, 3, 5, 7,1, 3, 4,0,9,6,5,7, 5,10,5,4, 2,7, 5,4, 1,3,1,2,1,2,2,5,2,1,3,1,2,0,1,5],
                 [ 8,10, 2, 5, 9,4, 7,13, 4,1, 4, 7,1,0,4,8,7, 5, 4,3,8, 1,3, 1,7, 3,4,5,1,1,3,8,1,2,2,3,7,2,5,6,9],
                 [ 6, 4, 9, 9,10,3, 5, 5, 9,1, 2, 6,3,1,7,6,5, 9, 9,1,1, 3,1, 4,0, 1,2,1,2,1,2,1,3,3,2,1,4,3,6,4,6],
                 [ 5,18, 7, 7, 9,4, 2,15,10,3, 4, 6,5,3,4,0,4, 3, 6,5,4, 1,2, 2,1, ,2,1,3,5,2,0,6,2,1,5,3,0,1,3,13],
                 [ 9,10, 5, 8,10,1, 9, 5, 8,1, 4, 7,1,4,2,5,0, 9, 4,3,9, 4,1, 5,3, 1,2,2,1,1,2,2,4,2,3,1,2,1,3,1,3],
                 [ 7,12, 8, 5, 9, 9,11,5,10,1, 0, 4,2,4,3,5,1, 0, 8,7,5, 1,3, 1,2, 1,3,1,2,3,2,3,2,4,3,2,0,1,7,4,5],
                 [16, 7, 9, 5,12, 5, 2,6, 7,5, 4,23,3,8,6,9,5, 5, 0,2,8, 3,2, 2,3,5,1,3,0,1,5,0,2,5,7,2,3,5,2,6,15],
                 [ 6, 2, 3, 1, 5, 2, 2,3, 0,2, 1, 3,2,1,0,3,2, 4, 3,0,3, 2,1, 4,1, 2,3,1,0,0,1,2,2,3,1,0,2,2,3,0,2],
                 [ 5, 2, 3, 1, 1, 3, 0,1, 0,3, 2, 3,2,1,0,1,0, 2, 0,1,0, 1,0, 0,1, 2,0,3,1,0,0,1,3,2,1,0,2,1,0,1,3],
                 [ 6, 3, 3, 4, 5, 3, 0,2, 0,0, 1, 2,1,2,0,1,2, 0, 1,0,1, 0,0, 1,2, 1,3,0,1,0,2,0,0,1,1,0,1,1,0,0,3],
                 [ 4, 2, 1, 2, 2, 0, 1,0, 1,2, 0, 1,2,1,0,1,1, 1, 2,0,2, 0,0, 0,2, 3,1,0,2,0,1,1,0,0,3,2,1,0,2,1,4],
                 [17, 2, 9, 5, 9, 2, 5,2, 5,1, 2, 4,7,4,1,2,5, 4, 7,2,3, 3,3, 0,4,2,2,5,2,0,1,3,2,5,4,4,2,7,3,2,10],
                 [ 9, 3, 2, 5, 1, 2, 5,1, 2,1, 0, 5,0,1,3,1,2, 1, 6,0,1, 2,1, 1,0, 1,0,4,3,0,3,1,3,4,7,1,2,5,1,3,9],
                 [10, 2, 5, 1, 1, 0, 1,5, 1,0, 1, 2,1,0,1,2,1, 0, 3,1,0, 1,0, 2,5, 0,1,2,3,1,0,2,1,0,3,4,1,2,0,2,5],
                 [ 8,12, 1, 5,10, 2, 5,2, 1,2, 0, 8,3,4,0,1,0, 6, 3,5,1, 2,1, 2,0,0,0,5,6,4,2,8,2,3,7,4,2,1,3,1,10],
                 [ 5, 2, 1, 0, 3, 0, 1,0, 1,2, 1, 2,1,0,10,0,2,0, 1,0,1, 0,2, 0,0, 0,1,2,0,1,0,2,1,1,0,2,3,4,2,2,5],
                 [11, 4, 2, 1, 1, 2, 2,0, 3,1, 2, 4,1,3,2,1,2, 1, 5,1,1, 3,1, 1,2,1,3,1,0,2,1,3,8,7,2,2,3,7,8,5,10],
                 [ 5, 7, 2, 1,10, 2, 5,5, 7,1, 3, 9,3,1,2,1,5, 1, 5,1,2,1,2,3,3,5,12,2,1,0,3,1,3,4,3,7,12,7,13,5,9],
                 [10, 4, 2, 1, 5, 0, 2,1, 0,1, 2, 4,0,1,5,0,1, 3, 4,7,1, 2,0, 4,1,2,2,1,3,5,0,1,2,1,2,2,9,5,8,7,15],
                 [ 9, 4, 2, 0, 1, 0, 2,0, 0,1, 2, 3,1,2,2,4,2, 1, 5,4,2, 1,2, 1,2, 0,1,1,4,2,3,0,0,3,2,1,0,2,3,1,4],
                 [ 8, 9, 2, 1, 6, 0, 2,1, 2,1, 4, 5,3,2,3,2,1, 6, 3,1,2, 1,3, 1,2,1,2,1,3,2,1,4,0,3,5,2,6,4,3,5,12],
                 [ 7, 2, 2, 2, 1, 2, 1,2, 1,2, 3, 1,2,1,2,2,1, 2, 1,0,1, 0,1, 0,0, 0,0,1,2,3,1,2,5,0,2,3,3,2,1,1,7],
```

```
                [11, 4, 2, 5, 1, 0, 2,5, 0,1, 5, 9,2,1,2,1,2, 1,10,1,1, 1,1, 3,1,2,1,5,2,4,3,1,2,1,0,5,9,7,4,7,11],
                [ 6, 2, 1, 1, 4, 0, 0,1, 0,1, 0, 1,1,0,0,0,2, 0, 3,0,0, 1,2, 0,2, 3,4,2,3,1,1,2,1,3,4,0,3,2,5,2,4],
                [15, 9, 2, 5, 1, 2, 2,3,10,1, 5, 2,2,3,5,1,2, 3, 4,1,2, 1,0,1,2,0,10,9,3,2,4,1,5,7,9,1,0,3,7,4,15],
                [15, 2, 2, 5, 3, 2, 4,1, 3,1, 2, 5,3,1,2,1,2, 3, 2,4,3, 1,3, 1,4,5,4,2,4,3,6,2,3,5,4,5,3,0,5,9,15],
                [10, 3, 4,15, 2, 6, 2,5, 3,1, 2, 3,2,2,1,3,3, 5, 4,1,1, 1,3, 1,2,6,4,2,6,5,4,3,4,6,4,5,3,8,0,7,12],
                [12, 4, 3, 5, 5, 2, 5,1, 0,1, 4, 9,3,1,4,1,1, 3, 8,3,2, 4,3, 1,1,4,1,7,3,9,2,4,7,3,8,9,3,5,4,0,17],
                [30,16, 9,10,15, 5, 7,5,6,4,5,17,2,5,3,7,4,3,19,4,1,6,5,4,3,11,15,3,3,8,5,8,4,7,10,6,10,15,14,9,0]]

"""En función del factor de escala establecido, podemos representar la demanda existente durante las diferentes
horas del día"""
FactorEscala = 1

"-------------------------------------------------------------------------------------------------"

"""A partir de las aristas se construyen los arcos de la red, ya que estamos tratando con un grafo dirigido
(se conoce el sentido que llevan las diferentes líneas de la red)"""
Arcos = []

print("1.Construyendo arcos a partir de aristas...")
for (i, j) in Aristas:
    Arcos.append((i, j))
    Arcos.append((j, i))
# print (Arcos)


"Se añaden tantas listas como líneas existen, para tener todos los nodos de las líneas en una sola lista"
L_Nodos_lineas = []

print("2.Incorporando en una sola lista los nodos que componen cada línea...")
L_Nodos_lineas.append(NodosL1)
L_Nodos_lineas.append(NodosL2)
L_Nodos_lineas.append(NodosL3)
# print (L_Nodos_lineas)


"""Se añaden tantas listas como líneas existen, para tener todas las aristas de las líneas en una sola lista"""
L_Aristas_lineas = []
```

```
print("3.Incorporando en una sola lista las aristas que componen cada línea...")
L_Aristas_lineas.append(AristasL1)
L_Aristas_lineas.append(AristasL2)
L_Aristas_lineas.append(AristasL3)
# print (L_Aristas_lineas)


"Se convierte la lista OD a diccionario OD, teniendo en cuenta el factor de escala"
D_OrigenDestino = {}

print("4.Generando diccionario con la demanda entre cada par OD y calculando demanda total...")
DemandaTotal=0
for i in range(len(Nodos)):
    for j in range(len(Nodos)):
        D_OrigenDestino[(i, j)] = L_OrigenDestino[i][j] * FactorEscala
        DemandaTotal=DemandaTotal+D_OrigenDestino[(i, j)]
    #print (D_OrigenDestino)
#print('Demanda Total',DemandaTotal)


"""A partir de los datos anteriores, definimos una lista y un diccionario que contienen
los diferentes arcos con sus correspondientes longitudes"""
L_Arcos_long = []
D_Arcos_long = {}

print("5.Generando lista y diccionario de arcos con sus correspondientes longitudes...")
for b, (i, j) in enumerate(Aristas):
    ListaAux = [0, 0, 0]
    ListaAux2 = [0, 0, 0]
    ListaAux[0] = i
    ListaAux[1] = j
    ListaAux[2] = LongAristas[b]
    ListaAux2[0] = j
    ListaAux2[1] = i
    ListaAux2[2] = LongAristas[b]
    L_Arcos_long.append(tuple(ListaAux))
    L_Arcos_long.append(tuple(ListaAux2))
    D_Arcos_long[(i, j)] = LongAristas[b]
```

```
    D_Arcos_long[(j, i)] = LongAristas[b]
# print (D_Arcos_long)
# print (L_Arcos_long)


"""A partir de la lista que contiene las aristas que componen cada línea, se crea una lista con todos los arcos
que componen cada líneas"""
L_Arcos_lineas = []
numlineas = len(L_Nodos_lineas)

print("6.Generando lista de arcos que componen cada línea...")
for i in range(numlineas):
    ListaAux = []
    for (j, k) in L_Aristas_lineas[i]:
        ListaAux.append((j, k))
        ListaAux.append((k, j))
    L_Arcos_lineas.append(ListaAux)
# print (L_Arcos_lineas)


"""Se define una lista que contiene la longitud total de cada línea, es decir, suma de las longitudes de todos
los arcos que la componen en ambos sentidos"""
L_LongTotal=[]

print("7.Generando lista con la longitud total de cada línea(ida y vuelta)...")
for i in range(numlineas):
    Sum=0.0
    for (j,k) in L_Arcos_lineas[i]:
      Sum=Sum+D_Arcos_long[(j,k)]
    L_LongTotal.append(Sum)
#print(L_LongTotal)


"""Se crean listas y diccionarios para los arcos que componen cada línea, pero esta vez se separan los arcos según
 sean sentido UP o DOWN (ida o vuelta) de la línea"""
L_ArcosLinea_up = []
L_ArcosLinea_dw = []
D_ArcosLinea_up = {}
```

```
D_ArcosLinea_dw = {}
print("8.Generando lista y diccionario de arcos que componen cada línea separando sentido UP y DOWN...")
for i in range(numlineas):
    ListaAux = []
    ListaAux2 = []
    for (j, k) in L_Aristas_lineas[i]:
        ListaAux.append((j, k))
        ListaAux2.append((k, j))
    L_ArcosLinea_up.append(ListaAux)
    L_ArcosLinea_dw.append(list(reversed(ListaAux2)))
    D_ArcosLinea_up[i] = ListaAux
    D_ArcosLinea_dw[i] = list(reversed(ListaAux2))


"""Se crea un diccionario donde cada clave se corresponde con un nodo de la red, y los valores de cada nodo
 equivalen a las líneas que pasan por el mismo. De esta forma obtenemos cuantas líneas pasan por cada estación"""
D_LineasporNodo = {}

print("9.Generando diciconario con las líneas que pasan por cada nodo de la red...")
for i in (Nodos):
    ListaAux = []
    for l in range(numlineas):
        if i in L_Nodos_lineas[l]:
            ListaAux.append(l)
    D_LineasporNodo[i] = ListaAux
#print (D_LineasporNodo[1])


"""Se crea un diccionario con los nodos que sirven para transbordar. Si por un nodo pasa más de una línea, dicho
nodo será de transbordo y se añadirá al diccionario. La clave del diccionario es el nodo y su valor las líneas
que pasan por él"""
D_NodosTransbordo = {}

print("10.Generando diccionario con los nodos que sirven para transbordar...")
for i in (Nodos):
    ListaAux = []
    for l in range(numlineas):
        if i in L_Nodos_lineas[l]:
```

```
            ListaAux.append(l)
    if len(ListaAux) > 1:
        D_NodosTransbordo[i] = ListaAux
# print (D_NodosTransbordo)


"""Se crea un diccionario con las líneas que pasan por cada uno de los arcos. La clave del diccionario es el arco,
 y sus valores son las diferentes líneas que pasan por este"""
D_LineasporArco = {}

print("11.Generando diccionario con las líneas que pasan por cada arco de la red...")
for (i, j) in Arcos:
    ListaAux = []
    for l in range(numlineas):
        if (i, j) in L_Arcos_lineas[l]:
            ListaAux.append(l)
    if len(ListaAux) == 1:
        D_LineasporArco[(i, j)] = ListaAux[0]
    else:
        D_LineasporArco[(i, j)] = tuple(ListaAux)
# print(D_LineasporArco)



"""Pasamos a generar la estructura del grafo mediante el paquete Networkx. Para ello, le pasamos la lista de nodos
y la lista de arcos con sus correspondientes longitudes"""
G = DiGraph()
print("13.Generando estructura del grafo de la red...")
G.add_nodes_from(Nodos)
G.add_weighted_edges_from(L_Arcos_long)


"""Mediante una función de Networkx, se obtiene el camino más corto (en forma de lista de nodos) de la red,
para ir desde un nodo origen i hacia uno destino j, y se guarda en un diccionario.
Además, también se obtiene otro diccionario con la longitud de dicho camino"""
D_Shortest_path = {}
D_Shortest_path_Long = {}
```

```python
for (i, j) in D_OrigenDestino:
    D_Shortest_path[(i, j)] = dijkstra_path(G, i, j, weight='weight')
    D_Shortest_path_Long[(i, j)] = dijkstra_path_length(G, i, j, weight='weight')
#print (D_Shortest_path)
# print (D_Shortest_path_Long)
print("14.Generando diccionario con el camino más corto para ir de i a j")
print("15.Generando diccionario con la longitud del camino más corto")



def k_shortest_paths(G, source, target, k, weight=None):

    """Calcula los k caminos más cortos para ir desde un nodo origen hasta otro destino.

    Como datos de entrada se tienen que proporcionar la red, el nodo origen y el destino,
    y el número k de caminos que se desean obtener.

    La función nos devuelve la lista de nodos correspondiente a cada uno de los k mejores caminos.

    Para su cálculo se sirve de una función perteneciente al paquete Networkx.
    """

    return list(islice(nx.shortest_simple_paths(G, source, target, weight=weight), k))
# for path in k_shortest_paths(G, 0, 5, 3):
# print (path)



def devuelve_pos(busca, Lista):

    """Calcula la posición de un elemento en una lista.

    Para ello, toma como entrada un elemento (en este caso un arco concreto de la red)
    y la lista completa de arcos.

    Devuelve la posición de dicho elemento en la lista mediante el método index.
    """
    return Lista.index(busca)
```

```
def compute_path_length(path, Lista_arcos, Lista_long):

    """Calcula la longitud total de un camino de la red.

    Función que toma como entrada un camino (lista de nodos que lo forman), la lista con todos los arcos de la red
    y las longitudes de dichos arcos.

    Nos devuelve la longitud total del camino a base de ir sumando las longitudes de los diferentes arcos
    que lo componen.
    """
    Suma = 0.0
    for c, i in enumerate(path):
        if c < len(path) - 1:
            origen = path[c]
            fin = path[c + 1]
            Suma = Suma + (float)(Lista_long[devuelve_pos((origen, fin), Lista_arcos)][2])
    return Suma



def ext_compute_path_length(ext_path, Lista_arcos, Lista_long):

    """Calcula la longitud total de un camino extendido de la red.

    Función que toma como entrada un camino extendido (lista de arcos que lo forman junto con la línea a la que
    pertenecen), la lista con todos los arcos de la red y las longitudes de dichos arcos.

    Nos devuelve la longitud total del camino a base de ir sumando las longitudes de los diferentes arcos
    que lo componen.
    """
    Suma=0.0
    for c,(i,j,k) in enumerate(ext_path):
            Suma=Suma+(float)(Lista_long[devuelve_pos((i,j),Lista_arcos)][2])
    return Suma
```

```
def generate_list_of_arcos(path):

    """Genera la lista de arcos que componen un camino de la red.
    Toma como entrada un camino de la red en forma de lista de nodos.
    """
    ListadeArcos = []

    for c, i in enumerate(path):
        if c < len(path) - 1:
            origen = path[c]
            fin = path[c + 1]
            ListadeArcos.append((origen, fin))

    #print (ListadeArcos)
    return ListadeArcos

def generate_list_of_arcos_extpath(extpath):

    """Genera la lista de arcos que componen un camino extendido de la red.

    Toma como entrada un camino extendido, es decir, lista de arcos junto con sus correspondientes líneas.
    """

    ListadeArcos=[]
    for c,(i,j,k) in enumerate(extpath):
            origen=i
            fin=j
            ListadeArcos.append((origen,fin))
    #print (ListadeArcos)
    return ListadeArcos



def generate_list_of_lineas_path(path_arcos):

    """Genera la lista de líneas a las que pertenecen los distintos arcos que componen un camino de la red.

    Toma como entrada la lista de arcos que componen un camino de la red.
```

```
    Devuelve las líneas a las que pertenecen los arcos, puede que un arco se corresponda con más de una línea.
    Si por ejemplo el segundo arco pertenece a las líneas 1 y 2 devuelve: [0,(0,1),2,0]
    """

    Lista_lineas_path = [() for (i, j) in path_arcos]
    for c, (i, j) in enumerate(path_arcos):
        Lista_lineas_path[c] = D_LineasporArco[(i, j)]

    return Lista_lineas_path



def devuelve_expandidos(ArcosPath, LineasPath):

    """Devuelve los caminos expandidos, es decir, en lugar de la lista de nodos que llevan de un origen i a
    un destino j, devuelve el conjunto de los arcos y las líneas de la red a las que estos corresponden, que
    llevan de i a j. """
    NumArcos=len(ArcosPath)
    NumlinArcos=[0 for i in range(NumArcos)]

    #se calculan cuántas lineas pasan por cada arco del path
    for i in range(NumArcos):
        aux=LineasPath[i] #aux es el valor que hay en el elemento i de la lista de lineas
        if type(aux)==int:
            NumlinArcos[i] = 1
        if type(aux)==tuple or type(aux)==list:
            NumlinArcos[i] = len(aux)

    #se calcula el número posible (multiply) de opciones para un mismo path, debido a que puede haber más de una
    #línea pasando por un mismo arco
    Multiply=1
    for i in range(NumArcos):
        Multiply*=NumlinArcos[i]

    #Abrimos mueva lista que deberá tener "Multiply" sublistas
    ListaExtend=[[] for i in range(Multiply)]
    for p in range(Multiply):
        ListaExtend[p]=list([(0,0,0) for (i,j) in ArcosPath])
```

```
    #Rellenamos la lista extendida del path con (arco,línea)
    for p in range(Multiply):
        for c,(i,j) in enumerate(ArcosPath):
            if NumlinArcos[c]==1:
                ListaExtend[p][c]=(i,j,LineasPath[c])
            #cuando un arco corresponde a más de una línea, se trata abajo de otra forma
            if NumlinArcos[c]>1:
                ListaExtend[p][c] = (i, j, 'x')

    #Completamos en la lista extendida, aquellos arcos por los que pasa más de una línea
    NumBuclesExt = 1
    for c, (i, j) in enumerate(ArcosPath):
        if NumlinArcos[c] > 1:
            Contador = 0 #marcará el número de estrategia en el que nos encontramos
            NumBuclesExt *= NumlinArcos[c]
            NumBuclesInt = int(Multiply / NumBuclesExt)
            #Se rellena el arco i,j tantas veces (multiply) como opciones para un mismo caminos haya
            while Contador < Multiply:
                z = 0
                for n in range(NumBuclesExt):
                    if z > (NumlinArcos[c] - 1):
                        z = 0
                    #pone el mismo arco y linea tantas veces como bucles internos, si hay 6 opciones y dos lineas,
                    # lo pone 3 veces, cuando termina pasa a la siguiente linea z+1 que pasa por el arco i,j
                    for m in range(NumBuclesInt):
                        ListaExtend[Contador][c] = (i, j, LineasPath[c][z])
                        Contador = Contador + 1 #pasa a rellenar la siguiente estrategia
                    z = z + 1
    #print (ListaExtend)
    return ListaExtend



def purga_expandidosTrasb(ListaExt):

    """Función que contabiliza el número de trasbordos que tiene cada uno de los posibles caminos extendidos
    que existen para cada par OD y guarda el valor del menor de ellos, para posteriormente eliminar aquellos
    caminos extendidos con mayor número de trasbordos que el mínimo."""
```

```
    NumPosibilidades=len(ListaExt)  #num de estrategias debido a la posibilidad de que por un arco pase más de una
línea

    # se crean dos listas aux, una cuenta trasbordos y otra anota que caminos se van a eliminar
    ListaContador = [0 for i in range(NumPosibilidades)]
    ListaDelete = [0 for i in range(NumPosibilidades)]

    if NumPosibilidades > 1:
        for i in range(NumPosibilidades):
            NumArcos = len(ListaExt[i])  # numero de arcos
            if NumArcos > 1:
                Trasb = 0
                for j in range(1, NumArcos):
                    #si el elemento en la  posicion 2(línea) es diferente de un arco al siguiente, cuento un trasbordo
                    if ListaExt[i][j][2] != ListaExt[i][j - 1][2]:
                        Trasb = Trasb + 1
                ListaContador[i] = Trasb
            if NumArcos == 1:
                Trasb = 0
                ListaContador[i] = Trasb
        #print (ListaContador)
        NumMinimo = min(ListaContador) #se guarda el número mínimo de trasbordos,
        #para a continuación eliminar los caminos que tengan más
        for i in range(NumPosibilidades):
            if ListaContador[i] > NumMinimo:
                ListaDelete[i] = 1

        # nueva lista de caminos
        y = [s for o, s in enumerate(ListaExt) if ListaDelete[o] == 0]

    # si solo hay un camino se queda la lista igual
    if NumPosibilidades == 1:
        y = [s for o, s in enumerate(ListaExt)]

    #print (y)
    return y
```

```
def purga_expandidosLong(ListaExt):

    """Función que calcula la longitud de cada camino extendido, guarda el valor del menor de ellos,
    para posteriormente, eliminar todos aquellos caminos que superen el valor de la longitud mínima en un 10%"""

    NumPosibilidades = len(ListaExt)  #numero de posibles paths debido a la posibilidad de que por un arco pase mas de
una línea

    ListaDelete = [0 for i in range(NumPosibilidades)]

    if NumPosibilidades > 1:
        ListaLon = []
        for k in range(NumPosibilidades):
            ListaAux = ListaExt[k]
            Aux = ext_compute_path_length(ListaAux, Arcos, L_Arcos_long)
            ListaLon.append(Aux)
        Longitudes = ListaLon   #lista que contiene la longitud de cada camino extendido
        Longmin = min(Longitudes)

        for i in range(NumPosibilidades):
            #si un camino tiene una longitud de más de un 10% que el camino con menor longitud, se elimina
            if Longitudes[i] > 1.1*Longmin:
                ListaDelete[i] = 1

        #nueva lista de caminos
        y = [s for o, s in enumerate(ListaExt) if ListaDelete[o] == 0]

    #si solo hay un camino posible, se queda la lista igual
    if NumPosibilidades == 1:
        y = [s for o, s in enumerate(ListaExt)]
    #print (y)
    return y
```

```
def F_append_elem(Lista1,Lista2):

    "Permite añadir elementos a un diccionario uno detrás de otro"

    Num1=len(Lista1)
    Num2=len(Lista2)
    Num3=Num1+Num2
    dev=[0 for i in range(Num3)]
    for c,i in enumerate(Lista1):
        dev[c]=Lista1[c]
    for c, i in enumerate(Lista2):
        dev[Num1+c] = Lista2[c]
    return dev



D_K_Shortest_paths = {}
D_K_Long_shortest_paths = {}
D_Num_shortest_paths = {}
NUMPATHS = 3
D_Ext_Shortest_paths={}
D_Ext_Lon_shortest_paths={}
D_Ext_Num_shortest_paths={}

def compute_k_shortest_paths_in_network():

    """Calcula diferentes características de los k caminos más cortos de la red, como son:

         *Conjunto de nodos que forman los k mejores caminos para cada par OD
         *Número de caminos que existen para cada par OD (siempre k o <k)
         *Longitudes de cada uno de los posibles caminos para cada par OD

    Todas las características anteriores las almacena en diccionarios.

    Posteriormente, dichos caminos que vienen dados en forma de nodos, los pasa a arcos, y tras esto los
    expande formando una tripleta que contiene (arco,linea)

    Se realiza una purga para eliminar aquellos caminos con más trasbordos que aquel que tiene menos, para
```

```
posteriormente realizar una segunda purga, que dejará solamente aquellos caminos con menos arcos, lo que
equivale a eliminar aquellos caminos de mayor longitud.
"""

for (i, j) in D_OrigenDestino:
    D_K_Shortest_paths[(i, j)] = k_shortest_paths(G, i, j, NUMPATHS, weight='weight')
    D_Num_shortest_paths[(i, j)] = len(D_K_Shortest_paths[(i, j)])
    ListaLong = []
    ListaAgregada=[]

    for k in range(0, D_Num_shortest_paths[(i, j)]):
        ListaArcosPath = generate_list_of_arcos(D_K_Shortest_paths[(i, j)][k])
       # print(ListaPathAux1)
        ListaLineasPath = generate_list_of_lineas_path(ListaArcosPath)
       # print(ListaPathAux2)
        ListaExpandidos = devuelve_expandidos(ListaArcosPath, ListaLineasPath)
        #print (k, ListaExpandidos)
        Listapurg = purga_expandidosTrasb(ListaExpandidos)
        #print ('purga1',Listapurg)
        ListaAgregada = F_append_elem(ListaAgregada, Listapurg)
        ListaAgregada=purga_expandidosLong(ListaAgregada)
        #print ('purga2',ListaAgregada)
    D_Ext_Shortest_paths[(i, j)] = ListaAgregada
    D_Ext_Num_shortest_paths[(i, j)] = len(ListaAgregada)

    for k in range(0, D_Num_shortest_paths[(i, j)]):
        #cálculo de la longitud de los caminos
        ListaAux = D_K_Shortest_paths[(i, j)][k]
        Aux = compute_path_length(ListaAux, Arcos, L_Arcos_long)
        ListaLong.append(Aux)
    D_K_Long_shortest_paths[(i, j)] = ListaLong

    ListaLon = []
    for k in range(0, D_Ext_Num_shortest_paths[(i, j)]):
        #cálculo de la longitud de los caminos extendidos resultantes tras las dos purgas
        ListaAux = D_Ext_Shortest_paths[(i, j)][k]
        Aux = ext_compute_path_length(ListaAux, Arcos, L_Arcos_long)
        ListaLon.append(Aux)
```

```
        D_Ext_Lon_shortest_paths[(i, j)] = ListaLon

    print("Paths Ext",D_Ext_Shortest_paths)
    # print("Numero Path Ext",D_Ext_Num_shortest_paths)
    # print("Longitud Path Ext",D_Ext_Lon_shortest_paths)

print ("17. Calculando los K=4 menores caminos (si los hay) para cada par OD como candidatos a mover a los
usuarios...")
print ("18. Almacenando el número de caminos para cada par OD ......")
print ("19. Calculando las longitudes de los diferentes caminos para cada par OD .......")
print ("20. Transformando los paths definidos en lista de nodos a lista de arcos.......")
print ("21. Obteniendo la lista de líneas que componen cada path......")
print ("22. Expandiendo caminos de tal forma que queden como arco,linea.......")
print ("23. Realizando purga para dejar solamente aquellos caminos con menos trasbordos......")
print ("24. Realizando segunda purga para dejar aquellos caminos con menos arcos......")



"""A PARTIR DE AHORA, LOS K CAMINOS YA NO SON LOS K (=4) INICIALES, SON LOS QUE HAN QUEDADO TRAS EXPANDIR LOS CAMINOS
Y REALIZAR LAS DOS PURGAS."""



def Ext_update_paths_with_freqs():

    """ Función que actualiza el diccionario de longitudes totales de cada camino, de tal forma que lo transforma
    en tiempos de  viaje (suma de los tiempos de cada vía que componen el camino), y le suma el tiempo que transcurre
    esperando la llegada del tren donde se inicia el trayecto, y además, si se realiza algún trasbordo,
    le suma también el tiempo esperando la llegada de la línea a la que se trasborda. """
    for (i, j) in D_OrigenDestino:
        ListaLon = []
        for k in range(0, D_Ext_Num_shortest_paths[(i, j)]):
            Aux = 0.0
            # Para cada camino extendido del par (i,j), se recorren sus arcos
            for m, (r, s, t) in enumerate(D_Ext_Shortest_paths[(i, j)][k]):
                 if (r, s) in L_ArcosLinea_up[t]:
                       Aux = Aux + TRup[t, r, s].x
```

```
                else:
                      Aux = Aux + TRdw[t, r, s].x
            ListaLon.append(Aux)
        D_Ext_Lon_shortest_paths[(i, j)]=ListaLon

        for k in range(0, D_Ext_Num_shortest_paths[(i, j)]):
            T_espera = 0.0
            for m, (r, s, t) in enumerate(D_Ext_Shortest_paths[(i, j)][k]):
               if m==0:
                   #si es el primer arco del camino (inicio viaje), se le suma el tiempo de espera de llegada del
tren,
                   #el cual se considera la mitad del headway de la linea
                    T_espera +=3600.0 / (2.0 * frec[t].x)

               if m > 0:
                   # Cuando se realiza un trasbordo hacia otra linea de la red, se considera que el tiempo de
                   #transferencia es la mitad del headway de la línea a la que se transborda
                    if D_Ext_Shortest_paths[(i, j)][k][m][2] != D_Ext_Shortest_paths[(i, j)][k][m-1][2]:
                         T_espera += 3600.0 / (2.0 * frec[t].x)
            #print('tiempo',T_espera)
            D_Ext_Lon_shortest_paths[(i, j)][k] = D_Ext_Lon_shortest_paths[(i, j)][k] + T_espera

        for k in range(0, D_Ext_Num_shortest_paths[(i, j)]):
            T_parada = 0.0
           # print(D_Ext_Shortest_paths[(i,j)][k])
            for (r,s,t) in (D_Ext_Shortest_paths[(i, j)][k]):
                a=devuelve_pos(r,L_Nodos_lineas[t])
                if (r,s) in L_ArcosLinea_up[t]:
                    T_parada=T_parada+TPup[t,a].x
                if (r,s) in L_ArcosLinea_dw[t]:
                    T_parada=T_parada+TPdw[t,a].x
              #  print(T_parada)
        #print(D_Ext_Lon_shortest_paths[(i, j)])
D_Ext_Prop_shortest_paths={}
D_Ext_Sum_long_paths={}
```

```
def Ext_compute_proportions_k_shortest_paths_in_network():
    """
        Función que calcula la proporción de personas que sigue cada uno de los K posibles caminos extendidos
        para cada par OD.

        Para ello, primero almacena en un diccionario D_Ext_Sum_long_paths[(i, j)] la suma de las longitudes (tiempos)
        de todos los posibles caminos.

        Si solo hay un camino para ir de i a j, todos los pasajeros irán por él.
        Sin embargo, si hay más de uno, el porcentaje de personas que toman cada camino se obtiene de forma
        inversamente proporcional a la longitud (tiempo para las siguientes asignaciones) de cada uno de los
        posibles caminos, es decir, a más longitud del camino, menos pasajeros irán por él.

        Estas proporciones son almacenadas en un diccionario D_Ext_Prop_shortest_paths[(i, j)]
     """

    for (i,j) in D_OrigenDestino:

        #Primero vamos a sumar las longitudes de todos los paths extendidos para ir de i a j
        D_Ext_Sum_long_paths[(i, j)] = 0.0
        for k in range(0, D_Ext_Num_shortest_paths[(i, j)]):
            D_Ext_Sum_long_paths[(i, j)] += D_Ext_Lon_shortest_paths[(i, j)][k]
        #print(i,j,D_Ext_Sum_long_paths[(i, j)])

        ListaProp = []
        #Se calcula la proporción que va por cada path
        if D_Ext_Num_shortest_paths[(i, j)] > 1:
            for k in range(0, D_Ext_Num_shortest_paths[(i, j)]):
                Aux=(D_Ext_Sum_long_paths[(i,j)]-D_Ext_Lon_shortest_paths[(i,j)][k])\
                    /((D_Ext_Num_shortest_paths[(i, j)]-1)*D_Ext_Sum_long_paths[(i,j)])
                ListaProp.append(Aux)

        #Si sólo hubiese un camino, todos los pasajeros se irán por el (proporción=1)
        else:
            ListaProp.append(1.0)
        D_Ext_Prop_shortest_paths[(i, j)] = ListaProp

        #print(i,j,D_Ext_Prop_shortest_paths[(i,j)])
```

```
print ("24. Calculando las fracciones de demanda de cada par extendido que sigue cada uno de sus caminos
admisibles...")


D_Ext_Arcos_paths = {}

def Ext_compute_arcos_en_paths():

    """Obtiene un diccionario D_Ext_Arcos_paths[(i, j)] que contiene las listas de arcos que forman cada uno de los
    K posibles caminos extendidos (tras haber realizado las 2 purgas) que sirven cada par OD de la red.

    Se usará para posteriormente computar la carga de pasajeros en los arcos.
    """

    for (i, j) in D_OrigenDestino:
        ListaAux = []
        for k in range(0, D_Ext_Num_shortest_paths[(i, j)]):
            ListaAux.append(generate_list_of_arcos_extpath(D_Ext_Shortest_paths[(i, j)][k]))
        D_Ext_Arcos_paths[(i, j)] = ListaAux
        #print(i,j,D_Ext_Arcos_paths[(i, j)])

print ("25. Almacenando la lista de arcos que componen cada path extendido (para cada par OD)......")


def devuelve_lin(Tupla1, Lista2):

    """Nos que permite obtener la línea a la que pertenece un determinado arco del path.

     Para ello, recibe como entrada un arco determinado y el camino extendido al que pertenece.
     """

    for (a,b,c) in Lista2:
        #Cuando el arco que se está analizando coincide con los dos primeros elementos de una tupla del camino
        # extendido, se copia el tercer elemento de dicha tupla como línea correspondiente.
        if Tupla1[0] == a and Tupla1[1]== b:
            linea=c

    return linea
```

```
D_Ext_Carga_Arco_Linea={}

def ext_compute_carga_en_arcos():

    """Calcula el flujo de personas que pasan por cada arco de la red, teniendo en cuenta a su vez, la línea a la
    que este pertenece.

    Para ello, se recorre toda la matriz OD buscando en todos los posibles caminos extendidos (k o <k)
    de cada par (i,j), si en el camino que va de i a j se pasa por el arco (s,t), se obtiene la línea
    a la que el arco pertenece y se agrega la cantidad de personas que circulan por este(Demanda OD x proporcion)

    """
    for (s,t) in Arcos:
        #Se realiza dos veces el mismo bucle, primero para inicializar a 0 las cargas de todos los arcos, y
        #después para ir agregando cargas al arco

        for (i,j) in D_OrigenDestino:
            for k in range(0, D_Ext_Num_shortest_paths[(i, j)]):
                if (s, t) in D_Ext_Arcos_paths[(i, j)][k]:
                    linea = devuelve_lin((s,t),D_Ext_Shortest_paths[(i, j)][k])
                    D_Ext_Carga_Arco_Linea[(s, t),linea]=0.0

        for (i,j) in D_OrigenDestino:
            for k in range(0, D_Ext_Num_shortest_paths[(i, j)]):
                if (s,t) in D_Ext_Arcos_paths[(i,j)][k]:
                    linea = devuelve_lin((s, t), D_Ext_Shortest_paths[(i, j)][k])
                    D_Ext_Carga_Arco_Linea[(s,t),linea]=D_Ext_Carga_Arco_Linea[(s,t),linea]+\
                                                        D_OrigenDestino[(i,j)]*D_Ext_Prop_shortest_paths[(i,j)][k]
    #print ('Carga arco',D_Ext_Carga_Arco_Linea)

print (". Calculando la carga de personas que lleva cada arco del grafo (para cada par OD extendido).....")
```

```
L_Ext_CargaMax_lineas=[0.0 for p in range(numlineas)]

def Ext_compute_CargaMax_all_lines():

    """Se calcula para cada línea el tramo (arco) más cargado, teniendo en cuenta las dos direcciones up y down.

    Esto servirá para dimensionar el número de trenes que requiere cada línea."""

    for i in range(numlineas):
         Max=0.0
         for (s,t) in L_Arcos_lineas[i]:
             if D_Ext_Carga_Arco_Linea[(s,t),i] > Max:
                 Max=D_Ext_Carga_Arco_Linea[(s,t),i]
         L_Ext_CargaMax_lineas[i]=Max
         print("línea",i,L_Ext_CargaMax_lineas[i])

print (". Calculando la carga máxima de cada línea de la red (path extendidos).......")



def TrasbordoBaj(s,t,i,extpath):

    """Devuelve si se ha producido trasbordo o no entre un determinado arco y el siguiente, ambos pertenecientes
    a un camino extendido concreto. Permitirá computar bajadas de una línea para trasbordar a otra.

    Para ello, recibe como entrada un determinado arco junto con la línea y el camino extendido a los que
    este pertenece."""

    Numarcos=len(extpath)
    valor=0
    for k in range(Numarcos-1):
       if extpath[k]==(s,t,i):
            #si la línea del arco que estamos analizando y el siguiente son diferentes, se realiza trasbordo
            if extpath[k][2]!=extpath[k+1][2]:
                #print(extpath[k])
               valor=1

    return valor
```

```
def TrasbordoSub(s,t,i,extpath):

    """Devuelve si se ha producido trasbordo o no, entre un determinado arco y el anterior pertenecientes
    a un camino extendido concreto. Permitirá computar subidas a una línea habiendo trasbordado desde otra.

    Para ello, recibe como entrada un determinado arco junto con la línea y el camino extendido a los que
    este pertenece."""

    Numarcos=len(extpath)
    valor=0
    for k in range(1,Numarcos):
       if extpath[k]==(s,t,i):
            #si la línea del arco que estamos analizando y el anterior son diferentes, se realiza trasbordo
            if extpath[k][2]!=extpath[k-1][2]:
                 #print(extpath[k])
                valor=1

    return valor


D_Ext_Destino_nodo_up={}
D_Ext_Transbordo_nodo_up={}
D_Ext_Bajadas_nodo_up={} # suma de los dos anteriores

def Ext_compute_bajadas_UP():

    """Calcula la cantidad de personas "que se bajan" (suma de las personas que llegan a destino y que se bajan
    para transbordar hacia otras lineas de la red) en cada nodo t usando el arco (s,t), en sentido up"""

    for i in range(numlineas):
        # para cada arco de la linea i en sentido up
        for (s, t) in D_ArcosLinea_up[i]:
            D_Ext_Destino_nodo_up[i,(s, t)] = 0.0
            for k in Nodos:
                # buscamos en los m caminos existentes para llegar de k a t
                if k!=t:
                     for m in range(0, D_Ext_Num_shortest_paths[(k, t)]):
                    #Seleccionamos el camino de cada par OD que acaba en nodo t usando el arco 'up' (s,t) de la linea i
```

```
                    if (s,t,i) in D_Ext_Shortest_paths[(k, t)][m]:
                       D_Ext_Destino_nodo_up[i, (s, t)] = D_Ext_Destino_nodo_up[i, (s, t)] + \
                       D_OrigenDestino[(k, t)] * D_Ext_Prop_shortest_paths[(k, t)][m]
                       #print("par" ,k,t, D_Ext_Destino_nodo_up)
            D_Ext_Bajadas_nodo_up[i,(s, t)] = D_Ext_Destino_nodo_up[i,(s, t)]

            #Calculamos personas que se bajan en nodo t usando arco (s,t) para transbordar
            D_Ext_Transbordo_nodo_up[i, (s, t)] = 0.0
            for k in Nodos:
                for p in Nodos:
                     if p!=k:
                          for m in range(0, D_Ext_Num_shortest_paths[(k, p)]):
                              #el nodo de fin de viaje p, tiene que ser distinto del de trasbordo t
                              if (s, t, i) in D_Ext_Shortest_paths[(k, p)][m] and p != t:
                                   trasb=TrasbordoBaj(s,t,i,D_Ext_Shortest_paths[(k, p)][m])
                                   if trasb==1:
                                       D_Ext_Transbordo_nodo_up[i,(s,t)] = D_Ext_Transbordo_nodo_up[i,(s,t)] + \
                                       D_OrigenDestino[(k, p)] * D_Ext_Prop_shortest_paths[(k, p)][m]
                                        #print("par", k, p, D_Ext_Transbordo_nodo_up)
            D_Ext_Bajadas_nodo_up[i,(s,t)] = D_Ext_Bajadas_nodo_up[i,(s,t)] + D_Ext_Transbordo_nodo_up[i, (s, t)]
    #print(D_Ext_Bajadas_nodo_up)

print (". Calculando la cantidad de personas que se bajan en cada nodo en sentido UP....")



D_Ext_Destino_nodo_down={}
D_Ext_Transbordo_nodo_down={}
D_Ext_Bajadas_nodo_down={} # suma de los dos anteriores

def Ext_compute_bajadas_DOWN():

    """Calcula la cantidad de personas "que se bajan" (suma de las personas que llegan a destino y que se bajan
    para transbordar hacia otras lineas de la red) en cada nodo t usando el arco (s,t), en sentido down"""

    for i in range(numlineas):
        # para cada arco de la linea i en sentido down
        for (s, t) in D_ArcosLinea_dw[i]:
```

```
            D_Ext_Destino_nodo_down[i, (s, t)] = 0.0
            for k in Nodos:
                # buscamos en los m caminos existentes para llegar de k a t
                if k!=t:
                    for m in range(0, D_Ext_Num_shortest_paths[(k, t)]):
                    #Seleccionamos el camino de cada par que acaba en nodo t usando el arco 'down' (s,t) de la linea i
                       if (s,t,i) in D_Ext_Shortest_paths[(k, t)][m]:
                          D_Ext_Destino_nodo_down[i, (s, t)] = D_Ext_Destino_nodo_down[i, (s, t)] + \
                          D_OrigenDestino[(k, t)] * D_Ext_Prop_shortest_paths[(k, t)][m]
            D_Ext_Bajadas_nodo_down[i, (s, t)] = D_Ext_Destino_nodo_down[i, (s, t)]
            #print(D_Ext_Bajadas_nodo_down)

            #Calculamos personas que se bajan en nodo t usando el arco (s,t) para transbordar
            D_Ext_Transbordo_nodo_down[i, (s, t)] = 0.0
            for k in Nodos:
                for p in Nodos:
                    if p!=k:
                        for m in range(0, D_Ext_Num_shortest_paths[(k, p)]):
                            #el nodo de fin de viaje p, tiene que ser distinto del de trasbordo t
                            if (s, t, i) in D_Ext_Shortest_paths[(k, p)][m] and p != t:
                                trasb=TrasbordoBaj(s,t,i,D_Ext_Shortest_paths[(k, p)][m])
                                if trasb==1:
                                    D_Ext_Transbordo_nodo_down[i,(s, t)] = D_Ext_Transbordo_nodo_down[i,(s,t)] + \
                                    D_OrigenDestino[(k, p)]*D_Ext_Prop_shortest_paths[(k, p)][m]
                                      #print("par", k, p, D_Ext_Transbordo_nodo_down)
            D_Ext_Bajadas_nodo_down[i,(s,t)] = D_Ext_Bajadas_nodo_down[i,(s,t)] + D_Ext_Transbordo_nodo_down[i,(s, t)]
  # print(D_Ext_Bajadas_nodo_down)

print (". Calculando la cantidad de personas que se bajan en cada nodo en sentido DOWN....")



D_Ext_Origen_nodo_up = {}
D_Ext_TransbordoS_nodo_up = {}
D_Ext_Subidas_nodo_up = {}  # suma de los dos anteriores

def Ext_compute_subidas_UP():
```

```
    """Calcula la cantidad de personas "que se suben" (suma de las personas que inician su viaje y que se suben
    para transbordar hacia otras lineas de la red) en cada nodo s usando el arco (s,t), en sentido up"""

    for i in range(numlineas):
        # para cada arco de la linea i en sentido up
        for (s, t) in D_ArcosLinea_up[i]:
            D_Ext_Origen_nodo_up[i,(s, t)] = 0.0
            for k in Nodos:
                # buscamos en los m caminos existentes para llegar de s a k
                if k != s:
                    for m in range(0, D_Ext_Num_shortest_paths[(s, k)]):
                        # Seleccionamos el camino de cada par OD que acaba en nodo t usando el arco 'up' (s,t)
                        if (s,t,i) in D_Ext_Shortest_paths[(s, k)][m]:
                            D_Ext_Origen_nodo_up[i,(s, t)]=D_Ext_Origen_nodo_up[i, (s, t)] + \
                                                           D_OrigenDestino[(s,k)]*D_Ext_Prop_shortest_paths[(s,k)][m]
            D_Ext_Subidas_nodo_up[i,(s, t)] = D_Ext_Origen_nodo_up[i, (s, t)]

            # Calculamos personas que se suben en nodo s para transbordar
            D_Ext_TransbordoS_nodo_up[i, (s, t)] = 0.0
            for k in Nodos:
                for p in Nodos:
                     if p != k:
                         for m in range(0, D_Ext_Num_shortest_paths[(k, p)]):
                             #el nodo de inicio de viaje k tiene que ser distinto del de trasbordo s
                             if (s, t, i) in D_Ext_Shortest_paths[(k, p)][m] and k != s:
                                 trasb=TrasbordoSub(s,t,i,D_Ext_Shortest_paths[(k, p)][m])
                                 if trasb==1:
                                     D_Ext_TransbordoS_nodo_up[i,(s, t)] = D_Ext_TransbordoS_nodo_up[ i,(s, t)] + \
                                                                           D_OrigenDestino[(k,p)]*D_Ext_Prop_shortest_paths[(k,p)][m]
                                     #print("par", k, p, D_Ext_TransbordoS_nodo_up)
            D_Ext_Subidas_nodo_up[i,(s, t)] = D_Ext_Subidas_nodo_up[i,(s, t)] + D_Ext_TransbordoS_nodo_up[i,(s, t)]
    #print(D_Ext_Subidas_nodo_up)
    #print(D_Ext_TransbordoS_nodo_up)

print (". Calculando la cantidad de personas que se suben en cada nodo en sentido UP....")
```

```
D_Ext_Origen_nodo_down = {}
D_Ext_TransbordoS_nodo_down = {}
D_Ext_Subidas_nodo_down = {}  # suma de los dos anteriores

def Ext_compute_subidas_DOWN():

    """Calcula la cantidad de personas "que se suben" (suma de las personas que inician su viaje y que se suben
    para transbordar hacia otras lineas de la red) en cada nodo s usando el arco (s,t), en sentido down"""

    for i in range(numlineas):
        # para cada arco de la linea i en sentido down
        for (s, t) in D_ArcosLinea_dw[i]:
            D_Ext_Origen_nodo_down[i, (s, t)] = 0.0
            for k in Nodos:
                # buscamos en los m caminos existentes para llegar de s a k
                if k != s:
                    for m in range(0, D_Ext_Num_shortest_paths[(s, k)]):
                        # Seleccionamos el camino de cada par OD que acaba en nodo t usando el arco 'up' (s,t)
                        if (s,t,i) in D_Ext_Shortest_paths[(s, k)][m]:
                            D_Ext_Origen_nodo_down[i,(s,t)] = D_Ext_Origen_nodo_down[i,(s,t)]+D_OrigenDestino[(s,k)]*\
                                                              D_Ext_Prop_shortest_paths[(s, k)][m]
            D_Ext_Subidas_nodo_down[i,(s, t)] = D_Ext_Origen_nodo_down[i, (s, t)]

            # Calculamos personas que se suben en nodo t para transbordar
            D_Ext_TransbordoS_nodo_down[i,(s, t)] = 0.0
            for k in Nodos:
                for p in Nodos:
                     if p != k:
                         for m in range(0, D_Ext_Num_shortest_paths[(k, p)]):
                             # el nodo de inicio de viaje k tiene que ser distinto del de trasbordo s
                             if (s, t, i) in D_Ext_Shortest_paths[(k, p)][m] and k != s:
                                 trasb=TrasbordoSub(s,t,i,D_Ext_Shortest_paths[(k, p)][m])
                                 if trasb==1:
                                    D_Ext_TransbordoS_nodo_down[i,(s,t)]=D_Ext_TransbordoS_nodo_down[i,(s,t)] + \
                                                                         D_OrigenDestino[(k, p)]*D_Ext_Prop_shortest_paths[(k, p)][m]
                                    # print("par", k, p, D_Ext_TransbordoS_nodo_down)
            D_Ext_Subidas_nodo_down[i,(s,t)]=D_Ext_Subidas_nodo_down[i,(s,t)] + D_Ext_TransbordoS_nodo_down[i,(s, t)]
    #print(D_Ext_TransbordoS_nodo_down)
```

```
print (". Calculando la cantidad de personas que se suben en cada nodo en sentido DOWN....")



SubidasenPlatUp=[[] for i in range(numlineas)]
SubidasenPlatDw=[[] for i in range(numlineas)]
BajadasenPlatUp=[[] for i in range(numlineas)]
BajadasenPlatDw=[[] for i in range(numlineas)]

for p in range(numlineas):
    SubidasenPlatUp[p] = list([(0) for i in L_Nodos_lineas[p]])
    SubidasenPlatDw[p] = list([(0) for i in L_Nodos_lineas[p]])
    BajadasenPlatUp[p] = list([(0) for i in L_Nodos_lineas[p]])
    BajadasenPlatDw[p] = list([(0) for i in L_Nodos_lineas[p]])

def Compute_sub_baj_plat():

    """Transformación de los diccionarios anteriores a subidas y bajadas en plataformas, imponiendo la condición de
    que: Las subidas en las últimas plataformas en ambos sentidos=0, y bajadas en las primeras plataformas=0
    """

    for l in range(numlineas):
        for o,(s,t) in enumerate(L_ArcosLinea_up[l]):
           SubidasenPlatUp[l][o]=D_Ext_Subidas_nodo_up[l,(s,t)]
           BajadasenPlatUp[l][o + 1] = D_Ext_Bajadas_nodo_up[l, (s, t)]
        SubidasenPlatUp[l][o+1] =0.0
        BajadasenPlatUp[l][0] = 0.0
        for o,(s,t) in enumerate(L_ArcosLinea_dw[l]):
           SubidasenPlatDw[l][o]=D_Ext_Subidas_nodo_down[l,(s,t)]
           BajadasenPlatDw[l][o+1] = D_Ext_Bajadas_nodo_down[l, (s, t)]
        SubidasenPlatDw[l][o+1] =0.0
        BajadasenPlatDw[l][0] = 0.0
    # print(BajadasenPlatUp)
    # print(BajadasenPlatDw)
    # print(SubidasenPlatUp)
    # print(SubidasenPlatDw)
```

```
compute_k_shortest_paths_in_network()
Ext_compute_proportions_k_shortest_paths_in_network()
Ext_compute_arcos_en_paths()
ext_compute_carga_en_arcos()
Ext_compute_CargaMax_all_lines()
Ext_compute_bajadas_UP()
Ext_compute_bajadas_DOWN()
Ext_compute_subidas_UP()
Ext_compute_subidas_DOWN()
Compute_sub_baj_plat()



'MODELO DE OPTIMIZACIÓN'

#Datos de entrada

TB=100                      #Tiempo empleado por cada línea en dar la vuelta en las estaciones finales en segundos
Vmax=120.0*1000/3600        #Velocidad máxima permitida en cada vía en km/h
Vmin=40.0*1000/3600         #Velocidad mínima permitida en cada vía
Cap=[414,607,832,997]       #Capacidad de cada modelo de tren
NuVag=[2,3,4,5]
NuPuertas=2
a=[0.5/(NuPuertas*i) for i in NuVag]              #Tasa de subida empírica según modelo
b=[0.5/(NuPuertas*i) for i in NuVag]              #Tasa de bajada empírica según modelo
H=[120,180,240,300,360,600,720,900,1200,1800]     #Posibles valores para el headway en segundos
F=[30,20,15,12,10,6,5,4,3,2]                      #Posibles valores para la frecuencia en trenes/hora
Tseg=60                                           #Tiempo de seguridad mínimo entre trenes
Aux=0.0

A=30      #máximo valor de la frecuencia
T=1800    #máximo valor del headway
M=100     #máximo valor del tamaño de flota
```

```
#Costes función objetivo
CTrip=22.085                                #Coste de la tripulación (€/hora)
Coper= [6.60,7.53,8.46,9.79]                #Coste de operación según modelo(€/km)
PesoT_Espera=2.5                            #Peso del tiempo de espera en las estaciones
PesoT_entren=1                              #Peso del tiempo a bordo del tren
PesoTransf=10/60.0                          #Peso de realizar un trasbordo (horas)
VOT=24.17                                   #Valor monetario del tiempo (€/hora)


m=Model("modelo")

#Definición de variables enteras y continuas
TRup = {}     #Tiempo de recorrido de cada vía en sentido up
TRdw = {}     #Tiempo de recorrido de cada vía en sentido down
TPup = {}     #Tiempo de parada en cada plataforma de la línea en sentido up
TPdw = {}     #Tiempo de parada en cada plataforma de la línea en sentido up
TC = {}       #Tiempo de ciclo de cada línea
head = {}     #Headway de cada línea
frec = {}     #Frecuencia de cada línea
TF= {}        #Tamaño de la flota de trenes de cada línea

for l in range(numlineas):
   for (i, j) in (L_ArcosLinea_up[l]):
      TRup[l,i,j] = m.addVar(lb=D_Arcos_long[(i, j)]/Vmax, ub=D_Arcos_long[(i, j)]/Vmin, vtype=GRB.CONTINUOUS,
                             name="TRUp_%s_(%s,%s)" % (l,i,j))
      TRdw[l,j,i] = m.addVar(lb=D_Arcos_long[(j, i)]/Vmax, ub=D_Arcos_long[(j, i)]/Vmin, vtype=GRB.CONTINUOUS,
                             name="TRdw_%s_(%s,%s)" % (l,j,i))
   for j in range(len(L_Nodos_lineas[l])):
      TPup[l,j]=m.addVar(lb=10,vtype=GRB.CONTINUOUS, name="TPUp_%s_%s"%(l,j))
      TPdw[l,j]=m.addVar(lb=10,vtype=GRB.CONTINUOUS, name="TPdw_%s_%s"%(l,j))

   TC[l]=m.addVar(vtype=GRB.CONTINUOUS,name="TiempodeCiclo_%s"%(l))
   head[l]=m.addVar(lb=120,ub=1800, vtype=GRB.INTEGER, name="Headway_%s"%(l))
```

```
   frec[l]=m.addVar(lb=2,ub=30, vtype=GRB.INTEGER, name="Frecuencia_%s"%(l))
   TF[l]=m.addVar(vtype=GRB.CONTINUOUS, name="TamañodeFlota_%s"%(l))


#Definición de variables binarias
Delta={}     #selección modelo tren
Landa={}     #selección headway

for l in range(numlineas):
   for j in range(len(Cap)):
      Delta[l,j]=m.addVar(vtype=GRB.BINARY, name="delta_%s_%s"%(l,j))

   for j in range(len(H)):
      Landa[l,j]=m.addVar(vtype=GRB.BINARY, name="landa_%s_%s"%(l,j))


#Variables necesarias para linealizar
Alfa= {}
Tau = {}
Omega= {}

for l in range(numlineas):
   for j in range(len(Cap)):
       Alfa[l,j]=m.addVar(vtype=GRB.INTEGER, name="alfa_%s_%s"%(l,j))
       Omega[l,j]=m.addVar(vtype=GRB.INTEGER, name="omega_%s_%s"%(l,j))
   for j in range(len(H)):
       Tau[l,j]=m.addVar(vtype=GRB.INTEGER, name="tau_%s_%s"%(l,j))


m.update()  #Se integran las variables en el modelo
```

```
condicion=1
cont=0
while(condicion==1):  #el modelo se integra en un bucle, para repetirlo hasta que se alcance la convergencia

  'Restricciones'
  for l in range(numlineas):
    for k in range(len(L_Nodos_lineas[l])):
      m.addConstr(TPup[l,k]>=quicksum(Omega[l,e]*a[e]*SubidasenPlatUp[l][k]/3600.0 for e in range(len(Cap)))\
          +quicksum(Omega[l,e]*b[e]*BajadasenPlatUp[l][k]/3600.0 for e in range(len(Cap))),
name="DwellUp_%s_%s"%(l,k))

    for k in range(len(L_Nodos_lineas[l])):
      m.addConstr(TPdw[l,k]>=quicksum(Omega[l,e]*a[e]*SubidasenPlatDw[l][k]/3600.0 for e in range(len(Cap)))\
          +quicksum(Omega[l,e]*b[e]*BajadasenPlatDw[l][k]/3600.0 for e in range(len(Cap))),
name="DwellDw_%s_%s"%(l,k))

    for n in range(len(L_Nodos_lineas[l])):
      m.addConstr(TPup[l,n]+Tseg <= head[l],name="AcotaDwellUp_%s"%(l))
      m.addConstr(TPdw[l,n]+Tseg <= head[l], name="AcotaDwellDw_%s"%(l))

    m.addConstr(TC[l]==quicksum(TRup[l,i,j]+TRdw[l,j,i] for (i,j) in (L_ArcosLinea_up[l]))+quicksum(
       TPup[l,e]+TPdw[l,e] for e in range(len(L_Nodos_lineas[l])))+2*TB,name="DefineCiclo_%s"%(l))

    m.addConstr(quicksum(Delta[l,o] for o in range(len(Cap))) == 1,name="SumSelTren_%s"%(l))

    m.addConstr(head[l] == quicksum(H[o]*Landa[l,o] for o in range(len(H))), name="DefineHead_%s"%(l))

    m.addConstr(frec[l] == quicksum(F[o]*Landa[l,o] for o in range(len(H))), name="DefineFreq_%s"%(l))

    m.addConstr(quicksum(Landa[l,o] for o in range(len(H))) == 1,name="SumVarHead_%s"%(l))

    m.addConstr(TC[l]==quicksum(H[o]*Tau[l,o] for o in range(len(H))),name="DefineFlota_%s"%(l))
```

```
  for o in range(len(H)):
     m.addConstr(Tau[l,o]<= M*Landa[l,o],name="LinTF1_%s_%s"%(l,o))

     m.addConstr(TF[l]-M*(1-Landa[l,o])<= Tau[l,o],name="LinTF2_%s_%s"%(l,o))

     m.addConstr(Tau[l,o]<=TF[l],name="LinTF3_%s_%s"%(l,o))

  for e in range(len(Cap)):
     m.addConstr(Omega[l,e]<=T*Delta[l,e],name="LinHead1_%s_%s"%(l,e))

     m.addConstr(head[l]-T*(1-Delta[l,e])<=Omega[l,e],name="LinHead2_%s_%s"%(l,e))

     m.addConstr(Omega[l,e]<=head[l],name="LinHead3_%s_%s"%(l,e))

     m.addConstr(Alfa[l,e] <=A*Delta[l,e],name="LinFreq1_%s_%s"%(l,e))

     m.addConstr(frec[l]-A*(1-Delta[l,e])<=Alfa[l,e],name="LinFreq2_%s_%s"%(l,e))

     m.addConstr(Alfa[l,e]<=frec[l],name="LinFreq3_%s_%s"%(l,e))

  m.addConstr(L_Ext_CargaMax_lineas[l]<=quicksum(Cap[o]*Alfa[l,o] for o in range(len(Cap))),name="Cargamax_%s"%(l))


'Función objetivo (€/hora)'
Conjunto1=[(e,l) for e in range(len(Cap)) for l in range(numlineas)]
Conjunto1=tuplelist(Conjunto1)
Conjunto2=[(l,i,j) for l in range(numlineas) for (i,j) in L_ArcosLinea_up[l]]
Conjunto2=tuplelist(Conjunto2)

CosteTrip=m.addVar(vtype=GRB.CONTINUOUS, name="CosteTrip")
CosteOper= m.addVar(vtype=GRB.CONTINUOUS, name="CosteOper")
CostePasEspera = m.addVar(vtype=GRB.CONTINUOUS, name="CostePasEspera")
CostePasTransb = m.addVar(vtype=GRB.CONTINUOUS, name="CostePasTransb")
```

```
  CostePasRec = m.addVar(vtype=GRB.CONTINUOUS, name="CostePasRec")

  m.addConstr(CosteTrip== quicksum(CTrip*TF[l] for l in range(numlineas)), name="Coste_de_trip" )

 m.addConstr(CosteOper == quicksum(Coper[e]*Alfa[l,e]*L_LongTotal[l]/1000.0 for e,l in Conjunto1.select('*','*')),
name="Coste_de_oper")

  m.addConstr(CostePasEspera == VOT * PesoT_Espera * quicksum((D_Ext_Subidas_nodo_up[l, (i, j)] +
D_Ext_Subidas_nodo_down[l,(j,i)])*(head[l]/(2 * 3600.0)) for (l, i, j) in Conjunto2.select('*', '*', '*')),
name="Coste_de_Espera")

  m.addConstr(CostePasTransb == VOT*PesoTransf*quicksum((D_Ext_TransbordoS_nodo_up[l,(i,j)]+
D_Ext_TransbordoS_nodo_down[l,(j,i)]) for (l,i,j) in Conjunto2.select('*','*','*')), name="Coste_de_Transbordos")

  m.addConstr(CostePasRec == VOT * PesoT_entren * quicksum(
       D_Ext_Carga_Arco_Linea[(i, j), l] * TRup[l, i, j] / 3600.0 + D_Ext_Carga_Arco_Linea[(j, i), l] \
       * TRdw[l, j, i] / 3600.0 for (l, i, j) in Conjunto2.select('*', '*', '*')), name="Coste_de_Pas_recorrido")

  m.update()


  Peso1=6
  Peso2=1
  m.setObjective(Peso1*(CosteTrip+CosteOper)+Peso2*(CostePasEspera+CostePasTransb+CostePasRec) ,GRB.MINIMIZE)


  m.write('modelo.lp')   #se escribe el modelo en un fichero
  m.optimize()
  obj=m.getObjective()


  # 'Se muestran por pantalla los resultados'
  for l in range(numlineas):
      print('Headway linea %s :'%(l),head[l].x)
      print('Freq linea %s :'%(l),frec[l].x)
      print('Flota linea %s :'%(l),TF[l].x,)
      print('Modelo tren linea %s :'%(l), sum(Delta[l,o].x*o for o in range(len(Cap))))
      print('Tiempo de ciclo de linea %s:'%(l),TC[l].x)
```

```
        print('Alfa linea %s:'%(l), sum(Alfa[l,o].x*o for o in range(len(Cap))))


        for (i, j) in (D_ArcosLinea_up[l]):
           print('Tiempo recorrido linea %s en via up %s,%s =' % (l,i,j), TRup[l,i,j].x)
           print('Tiempo recorrido linea %s en via dw %s,%s =' % (l,j,i), TRdw[l,j,i].x)

        for j in range(len(L_Nodos_lineas[l])):
           print('Tiempo parada linea %s en estacion up %s =' % (l, j),TPup[l,j].x)
           print('Tiempo parada linea %s en estacion dw %s =' % (l, j),TPdw[l,j].x)

    print('Coste tripulaciones %s:' % (CosteTrip.x))
    print('Coste operacion %s:' % (CosteOper.x))
    print('Coste pas tiempos de espera %s:' % (CostePasEspera.x))
    print('Coste pas tiempos de transbordo %s:' % (CostePasTransb.x))
    print('Coste pas tiempo de recorrido %s:' % (CostePasRec.x))
    print('Peso1: %s: Peso2: %s: Oper: %s: Pas: %s' % (Peso1, Peso2,
(CosteOper.x+CosteTrip.x),(CostePasEspera.x+CostePasRec.x+CostePasTransb.x)))

    D_Ext_Carga_Arco_Linea2={} #Diccionario que va a representar la carga de los arcos en la iteración anterior
    for l in range(numlineas):
        for (i,j) in L_Arcos_lineas[l]:
            D_Ext_Carga_Arco_Linea2[(i,j),l]=D_Ext_Carga_Arco_Linea[(i,j),l]


    Dif=0
    MaxDif=0
    'Comprobación de la convergencia'
    if Dif<1 and cont>0:
        condicion=0
    cont=cont+1


    'Se vuelve a realizar una asignación'
    compute_k_shortest_paths_in_network()
    Ext_update_paths_with_freqs()
    Ext_compute_proportions_k_shortest_paths_in_network()
    Ext_compute_arcos_en_paths()
```

```
  ext_compute_carga_en_arcos()
  Ext_compute_CargaMax_all_lines()
  Ext_compute_bajadas_UP()
  Ext_compute_bajadas_DOWN()
  Ext_compute_subidas_UP()
  Ext_compute_subidas_DOWN()
  Compute_sub_baj_plat()
  update_arcos_tiempos()


  'Cáculo de la diferencia de cargas en las distintas vías, de la interación anterior a la actual'
  for l in range(numlineas):
      for (i,j) in L_Arcos_lineas[l]:
          Dif=D_Ext_Carga_Arco_Linea2[(i,j),l]-D_Ext_Carga_Arco_Linea[(i,j),l]
          #print(i,j,l,Dif)
          if(MaxDif<Dif):
              MaxDif=Dif


'Una vez se alcanza la convergencia, se construyen las gráficas del recorrido de cada línea'
#Antes de su construcción, es necesario definir una serie de listas que contienen el tiempo de llegada y salida
#del primer tren de cada línea a cada una de sus estaciones, partiendo inicialmente desde un tiempo 0.

T_salUp=[[] for i in range(numlineas)]
T_llegUp = [[] for i in range(numlineas)]
T_salDw = [[] for i in range(numlineas)]
T_llegDw = [[] for i in range(numlineas)]
Tiempoll=0
for p in range(numlineas):
   Lista1 = []
   Lista2 = []
   Lista1.append(0)
   for n, (i,j) in enumerate(L_ArcosLinea_up[p]): #sentido up
       Tiemposal=Lista1[n]+TPup[p,n].x
       Lista2.append(Tiemposal)
       Tiempoll=Lista2[n]+TRup[p,i,j].x
       Lista1.append(Tiempoll)
   T_llegUp[p] = Lista1
```

```
    T_salUp[p] = Lista2
    Tiemposal=Tiempoll+TPup[p,n+1].x+TB  #estación final
    T_salUp[p].append(Tiemposal)

    Lista3 = []
    Lista4 = []
    for n, (i, j) in enumerate(L_ArcosLinea_dw[p]):  #sentido down
        Tiempoll = Tiemposal + TRdw[p, i, j].x
        Lista3.append(Tiempoll)
        Tiemposal = Tiempoll + TPdw[p, n].x
        Lista4.append(Tiemposal)
    T_salDw[p] = Lista4
    T_llegDw[p] = Lista3


'Pasamos a construir las gráficas para un solo tren de cada línea'
Recorrido=[[] for i in range (numlineas)]
Estaciones=[[] for i in range(numlineas)]

for n in range(numlineas):
    for p in range(len(L_Nodos_lineas[n])):   #recorrido sentido 'up'
        Recorrido[n].append(T_llegUp[n][p])
        Recorrido[n].append(T_salUp[n][p])

    for p in range(len(L_Nodos_lineas[n])-1):   #recorrido sentido 'down'
        Recorrido[n].append(T_llegDw[n][p])
        Recorrido[n].append(T_salDw[n][p])

    aux=0
    #Se construye una lista con la distancia entre estaciones
    Estaciones[n].append(0)
    Estaciones[n].append(0)
    for (i,j) in (L_ArcosLinea_up[n]):
        aux = aux + D_Arcos_long[(i, j)]
        Estaciones[n].append(aux)
        Estaciones[n].append(aux)
```

```
for (i, j) in (L_ArcosLinea_dw[n]):
       aux = aux - D_Arcos_long[(i, j)]
       Estaciones[n].append(aux)
       Estaciones[n].append(aux)


'Construcción de gráficas para todo el horizonte temporal (1 hora)'
#A los valores obtenidos para el primer tren de cada línea, se le suma el headway correspondiente tantas veces
#como la frecuencia de la línea indique, y así se obtiene el recorrido para una hora completa.
Recorridos = [[] for i in range(numlineas)]
for p in range(numlineas):
   Recorridos[p] = list([(0) for i in range(int(frec[p].x))])   #Se le da tamaño a la lista
   for j in range(int(frec[p].x)):
       Recorridos[p][j] = list([(0) for i in range(len(Recorrido[p]))])   #Se le da tamaño a la lista
       for i in range(len(Recorrido[p])):
            Recorridos[p][j][i]=Recorrido[p][i]+head[p].x*j


'Representación gráficas'
for n in range(numlineas):
    pyplot.title("GRAFICA DE RECORRIDO ")   #Título gráfica
    pyplot.xlabel("Tiempo(s)")              #Etiqueta eje horizontal
    pyplot.ylabel("Plataformas")            #Etiqueta eje vertical
   # pyplot.grid(True)                      #Cuadrículas

    for i in range(int(frec[n].x)):
      pyplot. plot(Recorridos[n][i], Estaciones[n], 'b')      #Dibujar recorrido trenes

    pyplot.show()    #Se muestra por pantalla el recorrido
```