UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS MATEMÁTICAS

TESIS DOCTORAL

Unas aplicaciones de la matemática computacional a la ingeniería del transporte

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Alberto Almech Jiménez

Director

Eugenio Roanes Lozano

Madrid

Universidad Complutense de Madrid
Facultad de Ciencias Matemáticas

Unas aplicaciones de la matemática computacional a la ingeniería del transporte

Memoria para optar al grado de Doctor

presentada por Alberto Almech Jiménez

dirigida por Eugenio Roanes Lozano

Madrid, 2022.

Programa de Doctorado en Ingeniería Matemática, Estadística e Investigación Operativa por la Universidad Complutense de Madrid y la Universidad Politectnica de Madrid

UNAS APLICACIONES DE LA MATEMÁTICA COMPUTACIONAL A LA INGENIERÍA DEL TRANSPORTE

Tesis doctoral presentada por Alberto Almech Jiménez

Dirigida por el Prof. Dr. Eugenio Roanes Lozano

año 2022

Agradecimientos

Deseo expresar mi estima y agradecimiento al profesor Eugenio Roanes Lozano (tutor y director) por su atención, paciencia y guía durante todos estos meses.

También quiero rendir gracias a quien de forma desinteresada me ayudó con las actividades formativas y a quien por su modestia y generosidad mantendré en el anonimato. En general, a toda la diligente organización del programa, y a la institución responsable, mi gratitud por esta oportunidad.

Prólogo

El interés común de este trabajo aborda cómo el valor del tiempo actúa sobre una red de ferrocarril o sus pasajeros, destacando el diseño y la representación gráfica sobre un software matemático de vasta difusión. De forma ilustrativa las cuatro aplicaciones organizan los capítulos y han requerido de la consideración y el esmero de su autor como si de un sólo trabajo se tratara. Aunque los capítulos gestionan problemas distintos, existe una relación de continuidad entre ellos que a modo de sumario se incluye a continuación.

Cada uno de estos capítulos está acreditado por su correspondiente artículo, publicado en una revista científica de comprobada calidad e impacto. El orden de exposición es la secuencia en que fueron desarrollados. Los capítulos tendrán todos la misma estructura, que a grandes rasgos, por su naturaleza, coincide con los de un artículo científico.

Por otro lado todos los trabajos están soportados por un software matemático muy difundido. El software elegido ha sido el potente CAS (Computer Algebra System) *Maple* (*Maple* es una marca registrada de Waterloo Maple Inc.).

El primer capítulo acomete los tiempos de viaje de un tren de ancho variable en una red con tres tipos de infraestructura, dos anchos de vía y cambiadores de ancho. Se consigue reducir la complejidad desde un multigrafo a un grafo multicapa, y en estas condiciones se consigue una ruta mínima con un algoritmo estándar. Este trabajo fue expuesto por primera vez en 2017 en el *17th International Conference on Computational and Mathematical Methods in Science and Engineering* (CMMSE2017 [35]). También ese mismo año se presentó en unas jornadas que la Comunidad de Madrid junto con varias universidades facilita a jóvenes investigadores (FuzzyMad17). Esta aplicación ampliada y completada, ha sido finalmente publicada en 2019 en un artículo de la revista *Mathematical Problems in Engineering*, indexada en SCI-JCR [8] .

En el segundo capítulo se continúa con el uso de la teoría de grafos que tan buenos resultados dieron en el capítulo anterior. Se afrontó el problema de recrear dinámicamente los planos históricos de unas redes sometidas a muchos cambios: las redes de ”metro”. Nuevamente se consiguieron buenos resultados reduciendo la evolución de la red de metro de Madrid a unos pocos grafos por línea, llamados

etapas. Estas conclusiones fueron expuestas en un póster de las Jornadas *FuzzyMAD18* y una contribución oral en la *24th Conference on Applications of Computer Algebra* (ACA2018). Se publicó finalmente en 2019, en un artículo de la revista *Mathematics in Computer Science* (indexada en Scopus y en Emerging Sources Citation Index) [7].

En el tercer capítulo, con la preocupación puesta en la representación gráfica de los tiempos de recorrido en una red de ferrocarril, ocurrirá lo contrario. Se parte de un grafo que representa una cartografía simplificada de nuestra red de ferrocarril. Elevar el grafo con una tercera dimensión nos permite visualizar datos, proporcionando un interesante gráfico. Las aristas se convertirán en superficies rectangulares, cuya superficie nos mostrará información visual de rápida comprensión, en nuestro caso del tiempo de viaje o la velocidad del recorrido. Un póster de esta propuesta fue presentado en la jornada FuzzyMAD 2019 [2]. Posteriormente en 2020 el artículo titulado ”A 3D proposal for the visualization of speed in railway networks” [6], fue publicado en la revista AIMS Mathematics (indexada en SCI-JCR).

El último capitulo ya no tiene relación con teoría de grafos, pero si mantiene la preocupación por la visualización gráfica y el tiempo. Se desarrolla una simulación de las colas de espera en una gran estación de ferrocarril. La propuesta ofrece grandes posibilidades combinatorias de las ventanillas compartidas para los diversos servicios (como cercanías, media y larga distancia o velocidad alta) y la existencia de ventanillas de último minuto que rompen el orden natural de llegada. El modelo es acometido como un autómata finito con una única cadena de llegada de viajeros. El artículo de este trabajo titulado ”An Accelerated-Time Simulation of Queues at Ticket Offices at Railway Stations” fue publicado en el año 2021 en la revista Mathematical Problems in Engineering (indexada en SCI-JCR)[9].

He procurado el principio de simplicidad tanto en el planteamiento como en el código de toda la exposición. He optado por una descripción más ágil, sin detalles técnicos de codificación, separando la descripción del código en los apéndices.

El autor,
febrero, 2022

UNAS APLICACIONES DE LA MATEMÁTICA COMPUTACIONAL A LA INGENIERÍA DEL TRANSPORTE

Resumen

Este trabajo aborda el valor del tiempo actuando sobre una red ferroviaria o sus usuarios, destacando la importancia del diseño y la representación gráfica en este tipo de problemas. Se trata de ilustrarlo con cuatro aplicaciones independientes, cada una en un capítulo, implementadas en software científico de amplia difusión.

El primer capítulo se ocupa de los tiempos de viaje de un tren de ancho variable en una red con tres tipos de infraestructura, dos anchos de vía y cambiadores de ancho. Utilizando la teoría de grafos en estas redes se consigue reducir la complejidad de un multigrafo a un grafo multicapa. En estas condiciones se consigue una ruta mínima con un algoritmo estándar como el de Dijkstra o Bellman-Ford.

En el segundo capítulo, aplicando nuevamente las herramientas de la teoría de grafos, se afrontó el problema de recrear dinámicamente los planos esquemáticos históricos de unas redes ferroviarias sometidas a muchos cambios, como son las redes subterráneas de ”metro”. Nuevamente se consiguió un buen resultado reduciendo la evolución de la red de metro de Madrid a unos pocos grafos llamados etapas de las líneas. La aplicación transforma la cronología de la red en unas pocas etapas, a partir de las que se obtiene la red retrospectiva en cualquier instante.

Con la representación gráfica de los tiempos de recorrido en una red de ferrocarril, se obrará a la inversa. En el tercer capítulo se parte de un grafo que representa una cartografía simplificada de nuestra red de ferrocarril. Elevar el grafo con una tercera dimensión nos permite la visualización de datos, proporcionando un interesante gráfico. Las aristas se convertirán en rectángulos cuya superficie muestra información visual de rápida comprensión, en nuestro caso del tiempo de viaje o la velocidad del recorrido.

El cuarto y último capitulo ya no tiene relación con teoría de grafos, pero si mantiene la preocupación por la visualización gráfica y el tiempo. Se desarrolla una simulación de las colas de espera en una gran estación de ferrocarril. La propuesta ofrece grandes posibilidades combinatorias de ventanillas compartidas para los diversos tipos de servicio (como cercanías, media y larga distancia o velocidad alta), así como la existencia de ventanillas de último minuto que rompen el orden natural de llegadas. El modelo es acometido como un autómata finito a partir de una cadena única de viajeros. La aplicación almacena las entradas, salidas y estados de la máquina en el proceso, permitiendo todo tipo de análisis o gráficos relacionados.

SOME APPLICATIONS OF COMPUTER MATHEMATICS TO TRANSPORTATION ENGINEERING

ABSTRACT

This work addresses the value of time acting on a railway network or its users, highlighting the importance of design and graphic representation in this type of problem. The idea is to illustrate it with four independent applications, each one in a chapter, implemented in widely used scientific software.

- The first chapter deals with the travel times of a variable gauge train in a network with three kinds of infrastructure, two track gauges and gauge changers. Using graph theory in these networks it is possible to reduce the complexity of a multigraph to a multilayer graph. Under these conditions, a minimum path is achieved with a standard algorithm such as Dijkstra or Bellman-Ford.

- Applying again the tools of graph theory, The second chapter addresses the problem of dynamically recreating historical rail network schemes that undergo many changes, such as metropolitan rail networks. Once again, a good result was achieved by reducing the evolution of the Madrid metro network to a few graphs, called stages. The application transforms the chronology of the network in a few stages, from which the retrospective network is obtained at any moment.

- With the graphical representation of the times traveled in a railway network, the opposite will be done. The starting point is a graphic that represents a simplified map of our rail network. Raising the graph with a third dimension allows us to visualize edge data. The edges will become rectangles whose surface shows us visual information that can be quickly understood, in our

case travel time or travel speed. The design of this interesting graph is developed in the third chapter.

- The last chapter is not related to graph theory, but it maintains the concern with graphical visualization and time. A simulation of the queues at a large train station is developed. The proposal offers great combinatorial possibilities of shared windows for the different kinds of service (such as commuter, medium and long distance or high speed), as well as the existence of last minute windows that break the natural order of arrivals. The model is interpreted as a finite automaton with a single chain of travelers. The application stores the inputs, outputs and states of the machine in the process, allowing all kinds of analysis or related graphics.

Introducción

Conocida es la importancia del acceso a una buena red de transportes para el desarrollo social y económico de una región.

En general se puede afirmar que la red de ferrocarriles se fue configurando durante el siglo XIX en relación principalmente al tamaño y a la actividad económica de las poblaciones y, recíprocamente, los nodos de la red ferroviaria pronto influyeron en el crecimiento y desarrollo de esos núcleos de población. El ferrocarril ha sido clave en la distribución y en la vertebración de la población y la industria.

El desarrollo de las vías y medios de transporte avanza rápidamente. El transporte tanto de personas como de mercancías es de sumo interés para las empresas involucradas, tanto públicas como privadas. En particular, la expansión y evolución del transporte público y de las redes ferroviarias ha experimentado en los últimos años un gran impulso.

A partir de la última década del siglo pasado la Unión Europea ha promovido una política de fomento del transporte por ferrocarril (fundamentalmente por razones ecológicas), así como de liberalización del acceso a las infraestructuras a nuevos operadores. Ello ha supuesto un gran reto para la organización de las empresas ferroviarias, generalmente herederas de una estructura de compañías privadas aisladas y dispersas en el siglo XIX y sistemáticamente unificadas y nacionalizadas en el caso europeo, según el país, en el intervalo entre los años 30 y los siguientes a la Segunda Guerra Mundial.

La red de ferrocarriles españoles es particularmente compleja y ha estado sometida a otros factores técnicos y políticos particulares que la dotan de unas características técnicas y geográficas especiales (a mediados del siglo XX: ancho de vía no estándar, gran proporción de vía estrecha, red principal claramente radial, severas rampas con radios de curvatura muy pequeños en zonas de orografía difícil, diversidad de estándares –heredados de las muchas compañías previas a la creación de Renfe (Red Nacional de los Ferrocarriles Españoles)-, etc.).

Desde un punto de vista técnico, existen:

- dos anchos de vía (excluimos aquí la vía estrecha): el tradicional Ibérico, 1.668 mm y el Internacional, 1.435 mm, usado en las nuevas líneas de alta velocidad (LAV) –con excepciones en ciertos tramos accesorios-,

- dos sistemas de electrificación: 3.000 V CC y 25.000 V CC a frecuencia industrial, este último usado en las nuevas LAV.

- cuatro familias de sistemas de señalización (aparte del bloqueo telefónico): ASFA (Anuncio de Señales y Frenado Automático), LZB (del alemán Linienzugbeeinflussung, influencia lineal en el tren) , EBICAB (marca registrada por Bombardier para un sistema de control automático de trenes -ATC-) y ERTMS (del inglés European Rail Traffic Management System).

- dos tecnologías de cambio de ancho automático: la tecnología Talgo RD para los trenes Talgo y la tecnología BRAVA (Bogie de Rodadura de Ancho Variable Autopropulsado), utilizada por los vehículos CAF (Construcciones y Auxiliar de Ferrocarriles). Asimismo los cambiadores pueden constar de una sola plataforma (Talgo o CAF) o de un sistema dual de plataformas, que opera el cambio de ancho sobre material rodante tanto Talgo como CAF.

Desde el punto de vista administrativo, por normativa europea, Adif (Administrador de Infraestructuras Ferroviarias) se encarga de la infraestructura y Renfe es un operador de forma análoga a lo que ha ocurrido con otros servicios básicos liberalizados o privatizados como la electricidad, el agua o la telefonía.

En cuanto a las relaciones internacionales, la conectividad con Portugal es débil, aunque se comparte el mismo ancho (sólo se encuentran activas las conexiones por Badajoz, Salamanca y Vigo). Con Francia sólo tiene lugar con intensidad por los dos lados de los Pirineos, estando las conexiones aragonesas, Canfranc y Puigcerdá, cerrada desde hace años la primera y con un tráfico testimonial la segunda. La expansión de la red española de vía de ancho estándar y la apertura del túnel de El Pertus está facilitando la conexión en el lado Este.

Es de destacar que España, según la UIC (Unión Internacional de Ferrocarriles), ocupa el segundo lugar mundial en kilómetros de LAV (explotadas a velocidades $> 250Km/h$), tanto en servicio como en construcción, sólo por detrás de China (Figura 1). La experiencia adquirida ha convertido a España en líder en materia de I+D+i ferroviario en campos como la infraestructura, la señalización, la electrificación o el material rodante.

Figura 1: Proyectos del Plan de Infraestructura, Transporte y Vivienda en LAV, 2012-2024. Fuente: Ministerio de Fomento.

Se requieren proyectos sobre las posibles mejoras de las características técnicas de las líneas existentes y sobre la construcción de nuevas líneas (para remediar la situación de cuellos de botella en la red existente, complementar la red clásica o la de alta velocidad), y estudiar su impacto sobre la población, el acceso a los recursos y a las industrias; la competencia con otros medios de transporte; el cumplimiento de la regulación pública (incluido el proceso de privatización), etc.

Índice general

Capítulo

1

Ruta mínima en la red de Adif: dos anchos de vía en varias infraestructuras y cambiadores de ancho

Desde un principio, en España, para conectar la red de alta velocidad y la ibérica se utilizaron **cambiadores de ancho**. La reserva de los operadores ferroviarios europeos a implementar trenes de mercancías con cambio de ancho automático ha obligado a considerar el uso de ancho mixto con **tercer carril** (que permite la circulación de trenes de dos anchos de vía diferentes) como una alternativa para que los trenes de mercancías procedentes del resto de Europa puedan circular en la red española. En este ámbito, se han habilitado varios tramos con tercer carril para permitir que los trenes de mercancías que accedan por la línea de alta velocidad Perpiñán-Figueras puedan alcanzar el resto de la peninsula. Según la Declaración sobre la Red que Adif publicó en 2021 [3], el ancho mixto permite el acceso de los trenes internacionales hasta el puerto de Barcelona, estando instalado entre Tardienta y Huesca y entre Valencia y Castellón. Fuera de esto, el uso del ancho mixto

es marginal o experimental, se está instalando en el corredor mediterráneo y se planea extender su utilización. El total de líneas de ancho mixto en la Red Ferroviaria de Interés General (RFIG) comprende unos cinco tramos con 118,8 km.

Existe otra infraestructura ferroviaria que no conviene olvidar, la de vía estrecha, de bajo coste en terrenos abruptos, se extiende principalmente por lugares de difícil acceso en el norte de España. La red de Ferrocarriles Españoles de Vía Estrecha (FEVE, desde 2012 absorvida por Adif/Renfe) ha sido la compañía que ha gestionado la red métrica más extensa de Europa, con 1.192 km de vía estrecha. La FEVE ha llegado a operar de forma paralela sobre cinco anchos de vía distintos (1.435 mm, 1.062 mm, 1.000 mm, 915 mm y 750 mm). El ancho métrico está desconectado mecánicamente del ancho ibérico y del internacional, por ese motivo no se ha considerado introducirlo en el modelo del trabajo.

1.1 Antecedentes

Se ha investigado mucho sobre los problemas que afectan al mundo del ferrocarril: se pueden encontrar muchos trabajos sobre optimizaciones del tráfico rodado [19, 20], redes de ferrocarril [18, 34, 42], transporte[32] e incluso de grado de accesibilidad entre nodos de la red de transporte de la España peninsular [34]. Pero muy poco sobre gestión de redes con vías de ancho mixto.

Centrándonos en la red administrada por Adif, y debido a la complejidad de esta, la Fundación de los Ferrocarriles Españoles encargó en 2010 a investigadores de la Universidad Complutense de Madrid y la Universidad Politécnica de Madrid, entre los que se encuentra el director de esta tesis, el diseñar e implementar un paquete informático de simulación, que se denominó RutasOptiRed, para ayudar a tomar decisiones sobre rutas óptimas, uso del parque ferroviario y extensión de infraestructuras [24]. Este proyecto se amplió en 2011 al cómputo de consumos, emisiones y costes [37]. Aparte de estos últimos, no conocemos un trabajo similar en redes ferroviarias de dos anchos de vía y cambiadores de ancho.

Existen otros estudios sobre proyectos de optimización alternativa, como por ejemplo el dirigido por el profesor Enrique Castillo y galardonado con el XIV Premio Talgo a la innovación tecnológica, y titulado “La vía alternada doble-simple como propuesta para nuevas LAV en áreas periféricas”. En él se optimiza mediante

programación lineal las vías únicas, dobles (para los cruces) y horarios, al objeto de proporcionar las altas prestaciones de una LAV convencional de doble vía a un coste de construcción significativamente inferior. [14].

1.2 Objetivo

Como ya se ha comentado, la búsqueda de la ruta más corta en la red de Adif fue acometido en dos ambiciosos proyectos financiados por la Fundación de los Ferrocarriles Españoles (FFE) y dió lugar a dos artículos en revistas indexadas SCI-JCR. Los autores, desarrollaron desde cero, en 2010 y 2011, un sofisticado software denominado RutasOptiRed. La primera versión encontraba rutas más cortas, evaluando con gran precisión la circulación del amplio material rodante del operador de Renfe [24]. La segunda versión agrega a esto los consumos y emisiones de los trenes [37]. Para atacarlo se consideraron grafos no dirigidos y con peso. Debido al nivel de precisión buscado se necesitó gran detalle documentado de secciones de vía y de material rodante, por ejemplo cada arco de los grafos tenían asociados una serie de atributos como ancho de vía, velocidad máxima, etc...

Las particularidades de la red requería que RutasOptiRed utilizara un algoritmo específico, hecho a medida de las necesidades. Se resolvió con una adaptación compleja y poco intuitiva del algoritmo de Dijkstra. El algoritmo es complicado y llama la atención el uso de los vértices del grafo como secciones de vía y los arcos como posibilidades de que una clase de tren pase entre secciones. Se puede encontrar este curioso algoritmo en [25].

El objetivo ahora es el mismo: calcular la ruta mímima en una red ferroviaria cualquiera con varios anchos de vía y cambiadores de ancho. La importancia de estas rutas mínimas es que establecen una distancia social y comercial entre localidades o regiones. La dificultad de jugar con varios anchos de vía, como se acaba de ver, estaba superada con el software anterior. La pregunta es: ¿Cómo modelizamos esta situación para poder utilizar los algoritmos habituales, de probada eficiencia?, ¿cómo bajamos la complejidad al problema? o ¿cómo podemos extenderlo de forma simple a otros escenarios parecidos?

Se ha tomado como referencia principal el técnico y detallado trabajo previo [24]. Es muy importante desvincularse de aquel enfoque. El planteamiento desa-

rrollado en la siguiente sección, aunque igualmente haga uso de grafos, es completamente original, buscando simplificar el modelo y sobre todo el algoritmo.

1.2.1 El problema de las multiaristas

A pesar de su sencillez, este ejemplo muestra la imposibilidad de usar directamente un algoritmo habitual de ruta mínima en la red ferroviaria española (este ejemplo apareció al probar la primera versión de RutasOptiRed [24]).

Se considera la ruta Madrid – Zaragoza – Tardienta – Huesca – Canfranc de la red ferroviaria Adif. Donde la situación era la siguiente:

- Hay dos conexiones ferroviarias entre Madrid y Zaragoza: una es la clásica línea de vía ibérica y la otra forma parte de la línea de alta velocidad Madrid-Barcelona .

- Zaragoza – Tardienta – Huesca – Canfranc es una de las líneas españolas que comunica ambos lados de los Pirineos, en ancho ibérico.

- Nuevo tramo Zaragoza-Tardienta de velocidad alta ($\leq 200km/h$) construido en ancho internacional.

- La sección Tardienta – Huesca fue elegida para probar nuevas soluciones técnicas como ancho mixto (tres raíles) o catenaria apta para 3kV y 25kV.

- Aunque existe un cambiador de ancho en Huesca, los cambiadores de ancho de esta ruta sólo se encuentran activos en Madrid y en Zaragoza.

En resumen, la situación de esta accidentada ruta y las características que nos ocupan se representan de manera simplificada en la Figura 1.1.

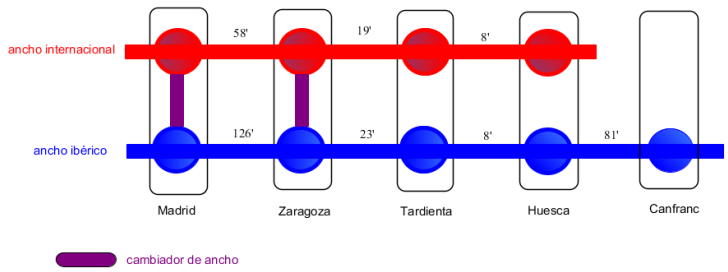

Figura 1.1: Esquema que representa en la línea azul inferior el ancho ibérico hasta Canfranc y en la superior en rojo la línea de ancho internacional que llega hasta Huesca. En morado los cambiadores de ancho activos, el peso del arco son lo minutos en recorrerlo al límite de velocidad máxima.

En este caso, si se ejecuta la ruta más corta entre Madrid y Canfranc en un tren de ancho variable, al ser Huesca un nodo ineludible hasta Canfranc, un algoritmo estándar recorrerá primero el camino más corto hasta Huesca. Proponiendo un recorrido absurdo de ida y vuelta:

1. Madrid – Zaragoza- Huesca: Llegar a Huesca lo antes posible. Ancho estándar, a lo largo de la línea de alta velocidad hasta Tardienta y en doble ancho hasta Huesca.

2. Huesca-Zaragoza-Canfranc: Al no poder continuar , el tren regresa a Zaragoza en ancho estándar, y cambiando el ancho, alcanzar Canfranc en ancho Ibérico.

1.3 Método

El problema es encontrar la ruta más corta en una red ferroviaria con varias infraestructuras y anchos de vía. En la red ferroviaria española opera material rodante dotado de tecnología de cambio de ancho automático, permitiendo cambiar de infraestructura sin cambiar de tren. Para aprovechar las posibilidades de toda la red ferroviaria consideramos que un tren puede circular a lo largo de ambos anchos de vía, necesitando atravesar un cambiador para pasar de una infraestructura a otra no compatible. Las infraestructuras contempladas son las vías de ancho internacional, ibérico y mixto.

La idea en el nuevo enfoque presentado aquí es estratificar las infraestructuras ferroviarias no compatibles, considerandolas en dos niveles o capas disjuntas. Cada capa representa a la red de cada ancho en forma de subgrafos inmersos espacialmente cada uno en un plano. Estos planos, paralelos, están conectados por medio de unos nuevos arcos añadidos que representan a los cambiadores de ancho. Si sólo se consideraran los trenes de ancho ibérico o de ancho estándar, sería suficiente eliminar los cambiadores de ancho, o lo que es lo mismo, penalizarlos con tiempo ilimitado.

La introducción de estas dos capas requerirá replicar tanto las estaciones que admitan los dos anchos de vía como los tramos de ancho mixto (se detalla en la sección siguiente).

De esta forma, se ha transformado el multigrafo inicial en un grafo que representa nuestro escenario y cumple las condiciones de cualquier algoritmo de ruta más corta estándar.

1.3.1 Grafos inducidos e interconexión

De un modo riguroso cada infraestructura forma una subred que induce un subgrafo, estos subgrafos G_i,G_s y G_t representarán respectivamente a la red de ancho ibérico, a la red de ancho internacional y a la red de ancho mixto. No tendrán ni nodos ni aristas comunes, para ello:

- Cada estación e (o punto de interés para la red como puede ser una bifurcación) se desdobla en tantos nodos como infraestructuras distintas tenga. Por ejemplo, en el peor de los casos, en una estación e que tuviera las tres infraestructuras, se construirían tres nodos e_i, e_s y e_t de G_i, G_s y G_t respectivamente. Tres nodos lógicos o plataformas para una localización física, es decir, la estación e, o punto característico, se descompone o proyecta en las tres subredes, en un principio inconexas.

- Los arcos de cada subgrafo son las secciones de vía de dicha infraestructura.

El acceso a toda la red se consigue con un nuevo grafo G, resultado de conectar los estratos o capas (subgrafos anteriores) por medio de los cambiadores de capa,

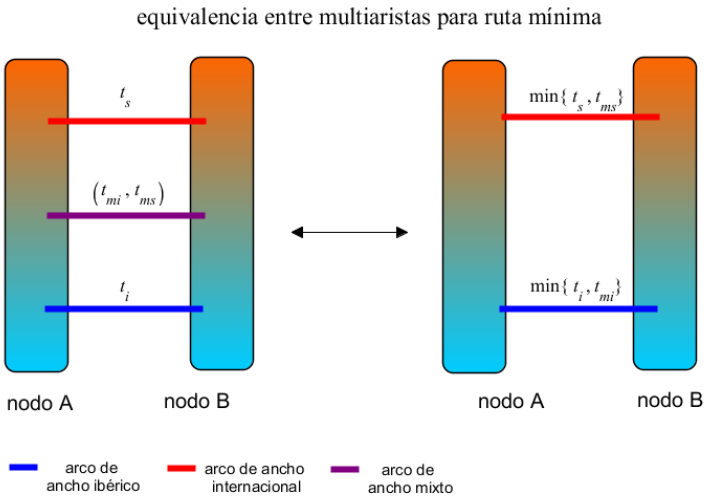

Figura 1.2: Equivalencia en un modelo multiarista. La t es el tiempo de recorrido entre nodos. En el ancho mixto aparecen dos; uno para cada ancho de vía.

reproduciendo la transición automática entre infraestructuras de un tren de ancho variable. Se distinguirían los cambiadores de ancho reales que conectan la infraestructura internacional con la ibérica, y los cambiadores de ancho virtuales, sin penalización, que conectan todos los nodos de G_t con sus homólogos de los otros dos anchos, simulando de esta forma la compatibilidad.

G_t complica el modelo. Para nuestro objetivo de ruta optima lo que interesa son los anchos de vía , no las infraestructuras. En este contexto, las líneas de ancho mixto de un modelo multiarista, se pueden descomponer en las aristas de los dos anchos de vía que la componen. Véase la Figura 1.2, donde cada tipo de arco se puede interpretar como una capa. Añadir de forma común estas secciones de ancho mixto en los dos primeros subgrafos permite prescindir del tercero (G_t) (ver Figuras 1.3 y 1.4).

La simplificación anterior admite velocidades máximas diferentes para cada ancho de vía en un ancho mixto, hecho que no admite el modelo general con G_t sin modificarse. No obstante, en los datos consultados, el arco de ancho mixto aparece sin los otros arcos, al igual que tampoco se cumple que las velocidades máximas medias de ancho mixto sean distintas para cada ancho (ocurre que $t_{mi} = t_{ms}$ en la Figura 1.2).

1. RUTA MÍNIMA EN LA RED DE ADIF: DOS ANCHOS DE VÍA EN VARIAS INFRAESTRUCTURAS Y CAMBIADORES DE ANCHO

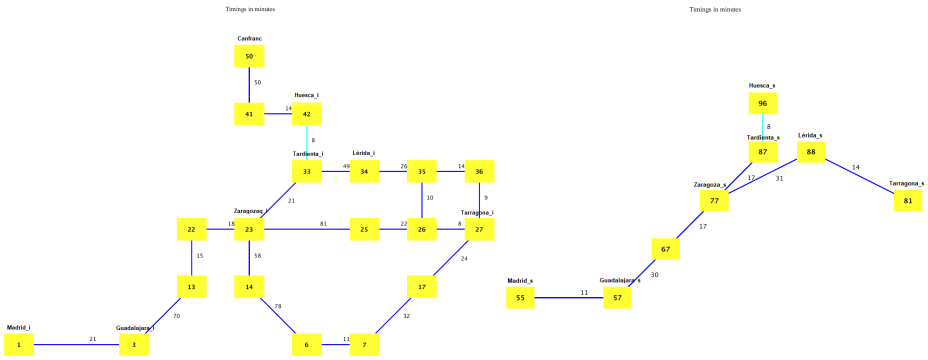

Figura 1.3: A la izquierda subgrafo de la representación del ancho ibérico de la región Centro-NE de Adif. A la derecha el subgrafo de ancho internacional(AV) de esa misma región. El tramo Tardienta-Huesca, en cyan, es de ancho mixto y aparece en las dos capas. El número del interior del nodo es el identificador del vértice. Fuente: [35]

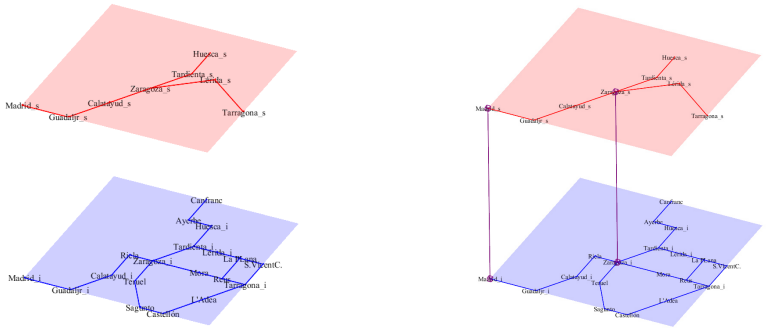

Figura 1.4: Representación 3D de los subgrafos inducidos por los anchos de vía en la región Centro-NE de Adif. Ancho internacional(AV) en rojo y ancho ibérico en azul. A la derecha se obtiene el grafo de toda la red añadiendo los cambiadores de ancho en morado. Fuente: [35]

1.3.2 Optimización

Con este nuevo modelo sólo se debe asignar a cada arco el tiempo estimado o el programado. Como cada arco corresponde a un tramo de la línea férrea, la única complejidad de esta operación es la de la obtención de los datos que la alimentan.

Se introduce el tiempo de recorrido como peso de las aristas al grafo G anterior. También se añade la penalización temporal a los arcos de los cambiadores. Ya se puede utilizar cualquier algoritmo habitual de ruta mínima en G.

Puede verse esto en las Figuras 1.4 y 1.5 del Ejemplo 1.1. Es una versión 2D proyectada aplicada a un esquema de la zona SW de la península.

Ejemplo 1.1 *Madrid-Zafra:* $3h10'$

Se puede ver en la Figura 1.6 dos ejecuciones de ruta mínima. Madrid-Zafra no se alcanza con alta velocidad hasta Córdoba sin imponer el paso por Córdoba. Madrid-Córdoba-Zafra: $3h33'$

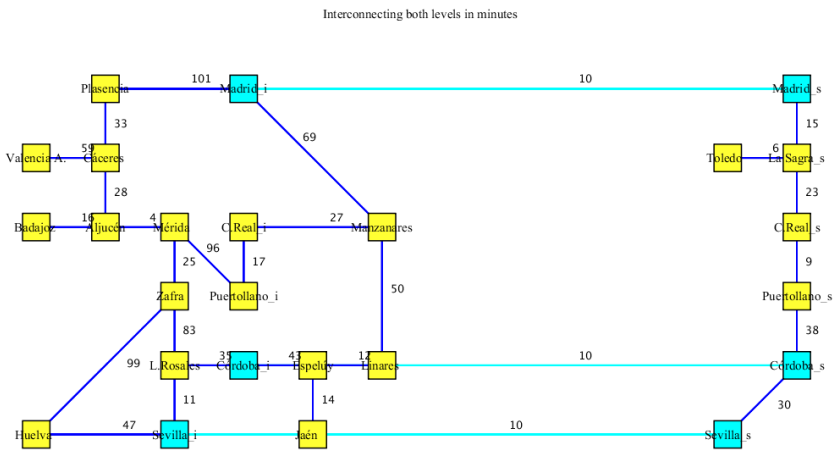

Figura 1.5: Grafo G en 2D de la región SW de Adif, en cyan los cambiadores que conectan los subgrafos inducidos por los dos anchos de vía, izquierda ancho ibérico y a la derecha internacional. Fuente: [35]

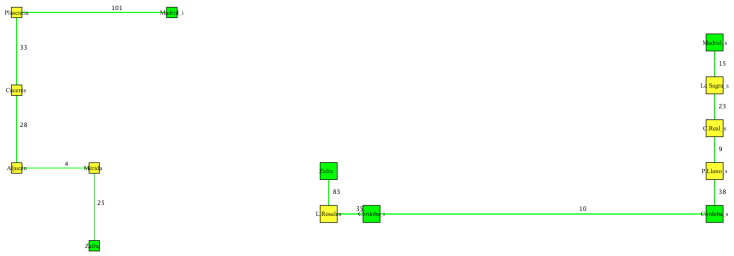

Figura 1.6: Ejemplos gráficos de ruta mínima entre Madrid y Zafra, a la izquierda sin restricciones y a la derecha pasando por Los Rosales (para obligar a usar alta velocidad). Los nodos verdes son origen, destino y cambiador). Fuente: [35]

1.4 Resultados

Reincidiendo en lo mismo, el objetivo del nuevo software descrito en este documento es sólo uno de los objetivos de RutasOptiRed, pero el principal. El nuevo enfoque, que enlaza la red de doble ancho, la red ibérica y la de vía estándar, permite utilizar una codificación y un algoritmo mucho más sencillo.

Con la construcción de este grafo G (Figuras 1.3 y 1.4), el contraejemplo anterior de Canfranc quedaría satisfactoriamente superado. El siguiente comando del paquete de *Maple*:

```
> DijkstrasAlgorithm(G,"Madrid_s","Canfranc");
```

devuelve la ruta más corta Madrid-Canfranc:

```
[["Madrid_s","Guadalajara_s","Calatayud_s","Zaragoza_s","Zaragoza_i",
  "Tardienta_i","Huesca_i", "Ayerbe", "Canfranc"],161]
```

Con el innovador método explicado en la sección anterior, el diseño e implementación del objetivo propuesto, en el software matemático *Maple* y con la ayuda de su paquete GraphTheory resulta relativamente sencillo. Se ha diseñado e implementado en este software un paquete para el cálculo de la mejor ruta en una red ferroviaria de doble ancho usando los algoritmos habituales de ruta óptima. Este trabajo fue expuesto en las Jornadas FuzzyMad17 y también se presentó ese mismo año en el congreso CMMSE2017 [35]. Esta aplicación partía de la red existente y sus características. Inicialmente constaba de un esquema de la zona SW de España, con 25 estaciones destacadas y 3 cambiadores de ancho. El comportamiento

del algoritmo se ha puesto a prueba sin problemas en una red mucho mayor extendiéndola a toda la España peninsular, con 145 estaciones destacadas y 12 de los 13 cambiadores de ancho activos, excluidos los de la frontera francesa (Figura 1.7). Esta aplicación completada, ha sido finalmente publicada en 2019 en un artículo de la revista *Mathematical Problems in Engineering* en SCI-JCR [8] .

1.4.1 Ejemplos

A continuación se verán unos ejemplos de la aplicación del software. En los tres primeros ejemplos se puede apreciar el peso de las Cercanías en las rutas calculadas. Remarcar que son tiempos sin paradas, sin transbordo y a la velocidad media máxima que permite cada sección de vía.

Ejemplo 1.2 $Santiago\,de\,Compostela - Barcelona_s$*:* $6h5'$

Desde Santiago, Barcelona se alcanza con el AVE por Madrid, haciendo uso de la alta velocidad castellana desde Medina del Campo (Figura 1.8)

Cambiemos el destino a las cercanías de Barcelona en el siguiente ejemplo.

Ejemplo 1.3 $Santiago\,de\,Compostela - Manresa$*:* $7h7'$

Para alcanzar Manresa se hace uso, como parece lógico, de la línea de alta velocidad sólo hasta Zaragoza ya que no existen más cambiadores hasta la frontera (Figura 1.9)

En este otro ejemplo se ve como cambia radicalmente la ruta con otra cercanía de Barcelona (incluso con la propia $Barcelona_i$ en lugar de Vich).

Ejemplo 1.4 $Santiago\,de\,Compostela - Vich$*:* $7h43'$

Un rodeo entre Manresa y Vich por Barcelona son suficientes para que el algoritmo cambie de ruta para llegar a Vich. Ahora haciendo uso de la alta velocidad por Valencia y atravesando el rápido corredor mediterráneo se llega a destino (Figura 1.10)

1. RUTA MÍNIMA EN LA RED DE ADIF: DOS ANCHOS DE VÍA EN VARIAS INFRAESTRUCTURAS Y CAMBIADORES DE ANCHO

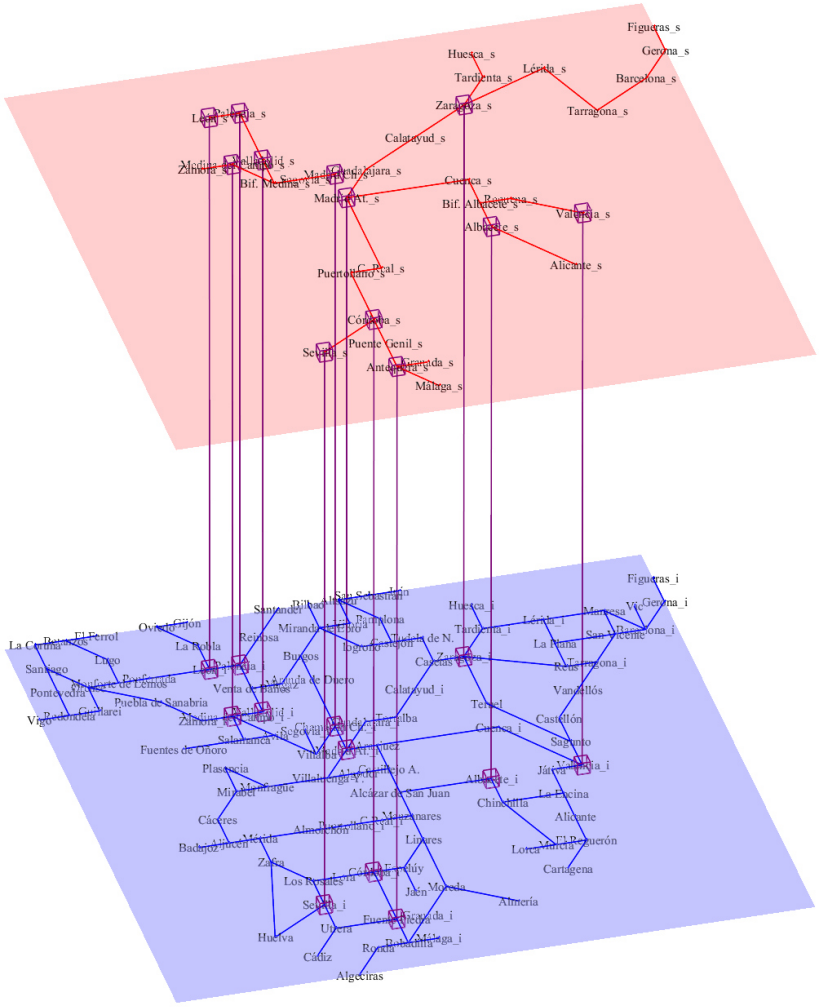

Figura 1.7: Red de FFCC de la España peninsular atendiendo al ancho de vía. Dos subgrafos conexos no dirigidos, cada uno correspondiente a un ancho de vía: ancho internacional (en rojo) y ancho ibérico (en azul). Los cambiadores de ancho son los arcos verticales en morado. Fuente: [8]

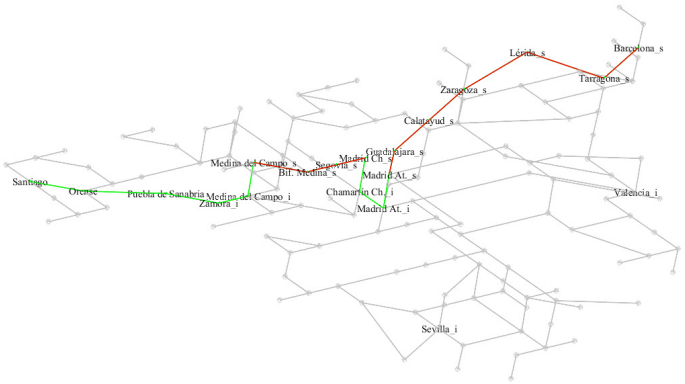

Figura 1.8: Ruta óptima en FFCC de doble ancho $Santiago - Barcelona_s$. Línea verde en ancho ibérico y la roja de alta velocidad en un plano superior. Realizado en *Maple*.Fuente: [8]

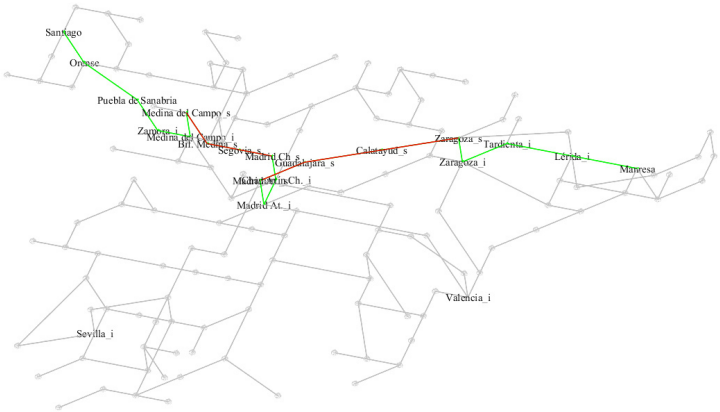

Figura 1.9: Ruta óptima en FFCC de doble ancho entre Santiago y Manresa. Línea verde en ancho ibérico y la roja de alta velocidad en un plano superior. Realizado en *Maple*. Fuente: [8]

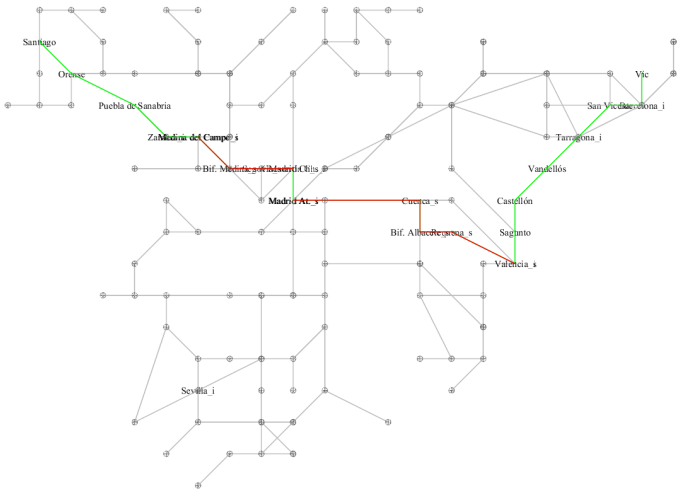

Figura 1.10: Ruta óptima en FFCC de doble ancho entre Santiago y Vich. Vista cenital.Línea verde en ancho ibérico y la roja de alta velocidad en un plano superior. Realizado en *Maple*. Fuente: [8]

Finalmente, y como aplicación a la línea del SW de Adif a la que se hizo referencia, observar la nula incidencia en Extremadura de la alta velocidad a Sevilla. Ya se vio en el Ejemplo 1.1, pero introducir un nuevo cambiador en Puertollano no cambia mucho la situación (Figura 1.11):

Ejemplo 1.5 $Madrid - Zafra$: $2h57'$

A Zafra se llega ahora por Puertollano. A Badajoz no varía la mejor ruta por Plasencia (Figura 1.12).

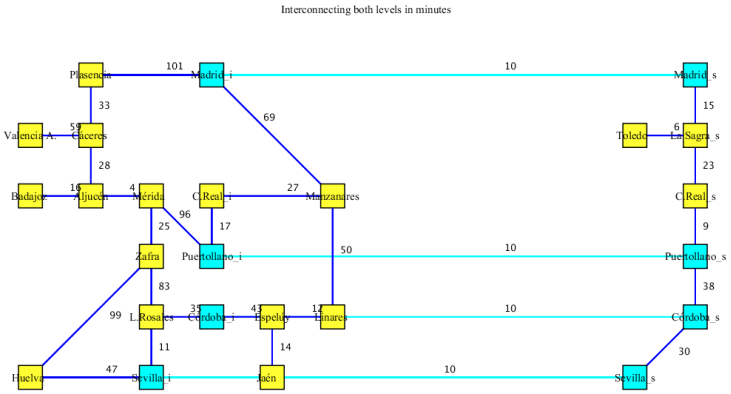

Figura 1.11: Grafo 2D de la red SW de Adif por ancho de vía a la que se ha añadido un nuevo cambiador en Puertollano. En cyan los cambiadores que conectan ambas subredes, peso de los arcos en minutos. Fuente: [35]

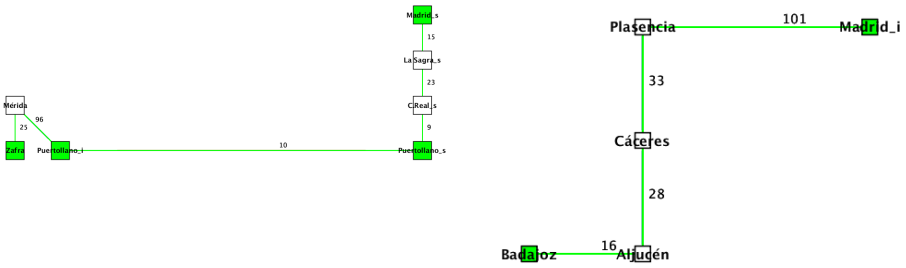

Figura 1.12: Ejecuciones de Ruta óptima añadiendo un hipotético cambiador en Puertollano. A la Izda. Madrid-Zafra y a la derecha la ruta Madrid-Badajoz no se ve afectada. Fuente: [35]

1.4.2 Observaciones

Como se hizo notar en el contraejemplo, el estado de los nodos que originó esta estructura depende del ancho de vía utilizado por el tren. El trabajo de calcular la ruta más corta posible se ha realizado sobre velocidades medias máximas en trenes de ancho variable, pero es extensible a las limitaciones de una composición de ancho fijo. El usuario final sufrirá la clase de tren que preste su servicio (trenes sin ancho variable, o con limitaciones de velocidad en ciertos tramos).

Los extremos de una ruta pueden ser estaciones con vías de los dos anchos (como Madrid en el Ejemplo 1.1). Si esto ocurre, hay que configurar el estado de los trenes en estos extremos para evitar innecesarias penalizaciones de cambio de ancho en origen o destino.

Capítulo

2

Grafo-cronología: mapas esquemáticos retrospectivos de la red de metro mediante grafos

El primer mapa popular de transporte en utilizar un diagrama esquemático para representar las distintas líneas que componen una red no fue diseñado hasta 1933 (el innovador y artístico mapa del metro de Londres de Harry Beck [1]). El éxito de este mapa se debe a su funcionalidad, primando una fácil comprensión topológica ante una detallada exactitud geográfica. Este diseño de mapa es utilizado en redes de transporte de todo el mundo, principalmente en ferrocarriles metropolitanos. Se puede seguir la evolución del diseño de los mapas del metro de Londres en [44].

Los mapas de extensas redes ferroviarias, con tiempos entre paradas muy heterogéneos, suelen ser fieles a las distancias.

En el mapa de una red metropolitana los tiempos entre localizaciones son cortos y más o menos homogéneos. En este último caso, los detalles cartográficos en el mapa pueden entorpecer la comprensión en la búsqueda de caminos. Esto se puede comprobar en muchas de estas redes que ponen a disposición de los usuarios planos

con ambos enfoques. Esta actividad esquemática apunta directamente a la teoría de grafos como herramienta candidata.

La evolución de una red se puede seguir a través de una colección de mapas ordenados cronológicamente. Facilita enormemente el conocimiento histórico disponer dinámicamente de mapas de este tipo, ya hayan sido editados o no. Este es el objetivo de este capítulo.

2.1 Antecedentes

Las posibilidades abiertas por nuevas herramientas en problemas como la evolución histórica de redes pueden abrir nuevos caminos en el desarrollo de utilidades. Los trabajos [40, 41], liderados por el director de esta tesis, son un claro ejemplo de este tipo de apuestas. Este trabajo obtenía mapas ferroviarios en cualquier instante a partir de una cronología. Implementado en el sistema de cómputo algebraico *Maple* y buscando tanta precisión geográfica como fue posible, proporcionaba unos planos que codificaban por colores y estilo de línea las características técnicas que cada sección de vía presentaba en la fecha deseada.

Aunque se encuentran trabajos relacionados, como la tesis [46] donde se dibujan automáticamente mapas esquemáticos, evaluando y optimizando el diseño, otros como [11, 13] donde se revisan en detalle los mapas esquemáticos o incluso en [12] donde se aborda la evolución de los mapas del metro de Melbourne, no se ha encontrado un enfoque para dibujar mapas esquemáticos basados en las transformaciones experimentadas por una red metropolitana a lo largo del tiempo.

2.2 Objetivo

Tanto los mapas actualizados como las animaciones históricas siguen literalmente una cronología, es decir, las secciones o estaciones se agregan o eliminan en las diferentes líneas secuencialmente en un proceso de integración de eventos de una línea temporal. Vamos a utilizar la siguiente notación:

- La red existente en un momento determinado se denominará *estado* de la red o *red acumulada* en ese instante.

- La *evolución de la red* se entenderá como la secuencia de estados de la red que genera la cronología en cada instante.

Los mapas o redes de transporte por ferrocarril nos permite el uso de diagramas esquemáticos en forma de grafos y por lo tanto poder utilizar los recursos que nos brinda determinado software científico.

El objetivo del trabajo es obtener la *red acumulada* y trazar un mapa esquemático del metro en cualquier fecha propuesta a bajo coste computacional, nuevamente con el uso de grafos. Centrando el interés en la topología del grafo, se hace uso de coordenadas para no desatender la orientación geográfica. El presente trabajo se ocupa de la red disponible al público. Otras características más técnicas, como la infraestructura (electrificación, señalización, tipo de vía, etc.) se podrían tratar de forma similar.

La evolución L de una línea, en tiempo discreto, la definimos como una sucesión finita de estados representados en cada instante t_i por un grafo L_i:

$$L = \{L_i : \text{ grafo en el instante } t_i\} \tag{2.1}$$

La evolución de una línea está caracterizada, como veremos, por muy pocos de estos grafos, estados que conservan toda la información de su pasado y a los que llamaremos etapas. Esta orientación también se implementa en *Maple* aprovechando las posibilidades del paquete *GraphTheory*.

Para ilustrarlo se toma como ejemplo el siglo de historia de la red de metro de Madrid. Con aproximadamente 300 estaciones es la tercera más grande de Europa. Por claridad, se han redondeado las fechas anualmente.

2.2.1 El problema de perder información

Viendo la evolución de los mapas esquemáticos, no se tarda mucho tiempo en apreciar que el problema no es tan sencillo como parece, que alguna información se disuelve con el transcurrir del tiempo. Un evento puede eliminar algunas secciones o cambiarlas a otra línea y esta información se olvida en estados posteriores. Esto se manifiesta en los cambios experimentados por las Líneas 2 y 5 del Metro de Madrid- entre 1969 y 1970 (Figura 2.1): estas líneas no sólo se extienden, también se transfieren secciones de una a otra.

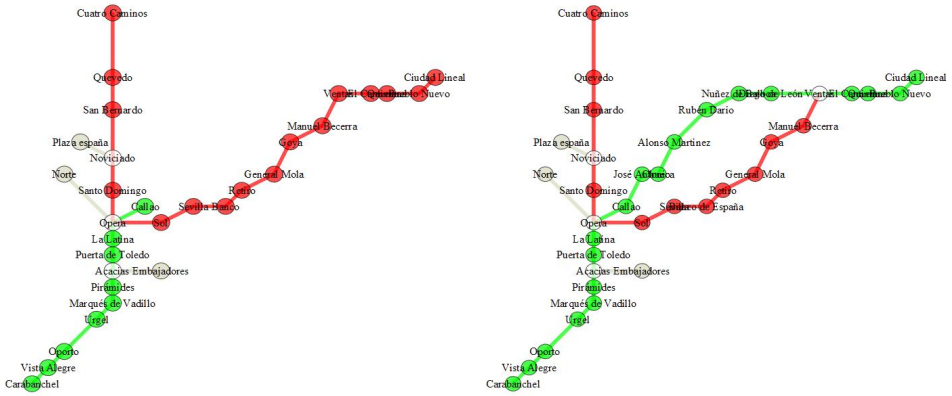

Figura 2.1: línea 2 (roja) y línea 5 (verde) en 1969 (izda.) y 1970 (dcha.). Fuente: [7]

2.3 Método

El mapa histórico esquemático de la red de metro puede considerarse de forma intuitiva como una red formada por grafos no dirigidos con peso:

- los nodos son las estaciones de la red de Metro.
- los arcos son las secciones o tramos de línea entre estaciones.
- el peso es la fecha de apertura de la sección, agrega el valor histórico.

Cumpliendo:

- Un nodo es común a varias líneas si y sólo si representa a estaciones de transbordo para todas las líneas incidentes en este nodo, como habitualmente aparece en los planos.
- las líneas son grafos y su unión con los nodos comunes forman la red.

Las líneas son los grafos predefinidos que forman la red y centrarán nuestra atención. Es importante la distinción de líneas dentro de la red, son entidades en sí mismas y cada una tiene asociado un color al representarlas gráficamente.

En caso de multiaristas en la red hay fechas distintas, una por cada arco de línea. En la actualidad en la red de Metro de Madrid sólo existen tres aristas múltiples: *Avda. América-Diego de Leon (L4 y L6)*, *Plaza Castilla-Chamartin (L1y L10)* y *Arguelles-Moncloa (L3 y L6)*.

La red la dividiremos en líneas y su evolución se podrá representar mediante una lista de grafos que llamaremos *etapas* de la línea. Descompondremos así el problema de la evolución en una pequeña lista de grafos con fechas asociadas a sus arcos (Figura 2.2).

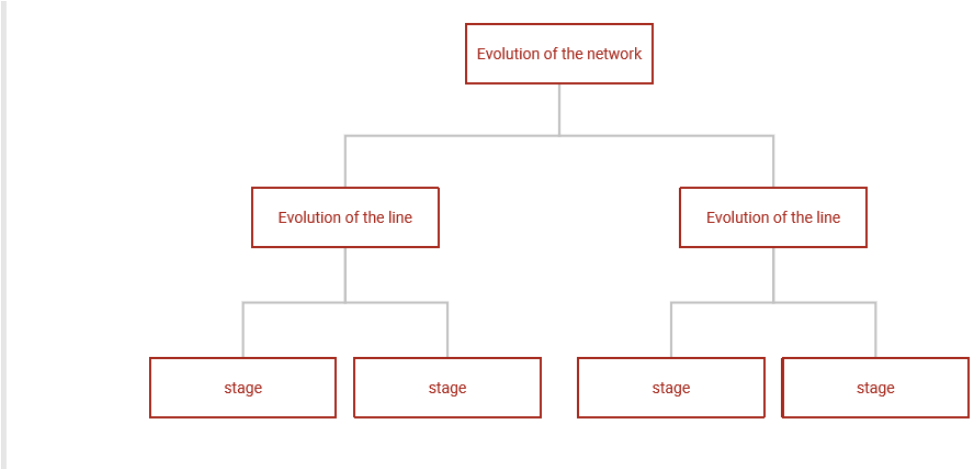

Figura 2.2: Descomposición de la evolución de la red en subgrafos.

2.3.1 Estado de una línea

El efecto que un evento o cadena de eventos provoca sobre una línea consiste en agregar o suprimir aristas, entendiendo que cada arista está asociada a dos nodos (estaciones).

Para una clara exposición de lo que nos interesa, podemos clasificar el estado de una línea L_i según el cambio que haya experimentado en el instante t_i:

- **estado expansivo**: Cuando en una línea no hay cambios o cuando sufre un evento que exclusivamente agrega aristas. No se elimina nada, no se pierde información. L_{i-1} es un subgrafo de L_i .

- **estado crítico**: Cuando no es un estado expansivo, es decir, cuando como consecuencia de un evento se elimina algún arco. Hay información que ha desaparecido del grafo. L_{i-1} no es un subgrafo de L_i .

Al estado anterior a uno crítico se le llamará *fin del período* o simplemente *etapa*:

$$L_i \text{ es una etapa de L} \iff L_i \text{ no es subgrafo de } L_{i+1} \tag{2.2}$$

2.3.2 Etapas de una línea

Las etapas tienen que ser recordadas porque contienen información cronológica que en estados posteriores se pierde. Como los estados entre dos etapas son expansivos y no pierden información, se tiene:

Observación 2.1 *Las etapas conservan toda la información cronológica de la evolución*

Como no conocemos el futuro, al último estado también hay que considerarlo una etapa :

Observación 2.2 *número de etapas= 1 + número de estados críticos*

Señalar que eliminar una estación o interpolar una estación intermedia (añadir o quitar un nodo) crea una etapa, ya que hay que eliminar un arco para formar dos o viceversa. Ocurre de forma parecida cuando una sección cambia de una línea a otra, que aunque la red como multigrafo permanece invariante, hay una línea que pierde arcos y otra que los gana. Como anécdota, el metro de Madrid cuenta con la estación fantasma de Chamberí, fuera de servicio desde 1966.

Ejemplo 2.1 *La evolución de la Línea 2 está formada por cuatro etapas, consecuencia de sus 3 eventos críticos correspondientes (Figura 2.3):*

1958 *Eliminación de un ramal (a favor de la Línea 4).*

1969 *transferencia a Línea 5 de algunas secciones.*

1998 *Nueva estación intermedia de Canal (transbordo con Línea 7).*

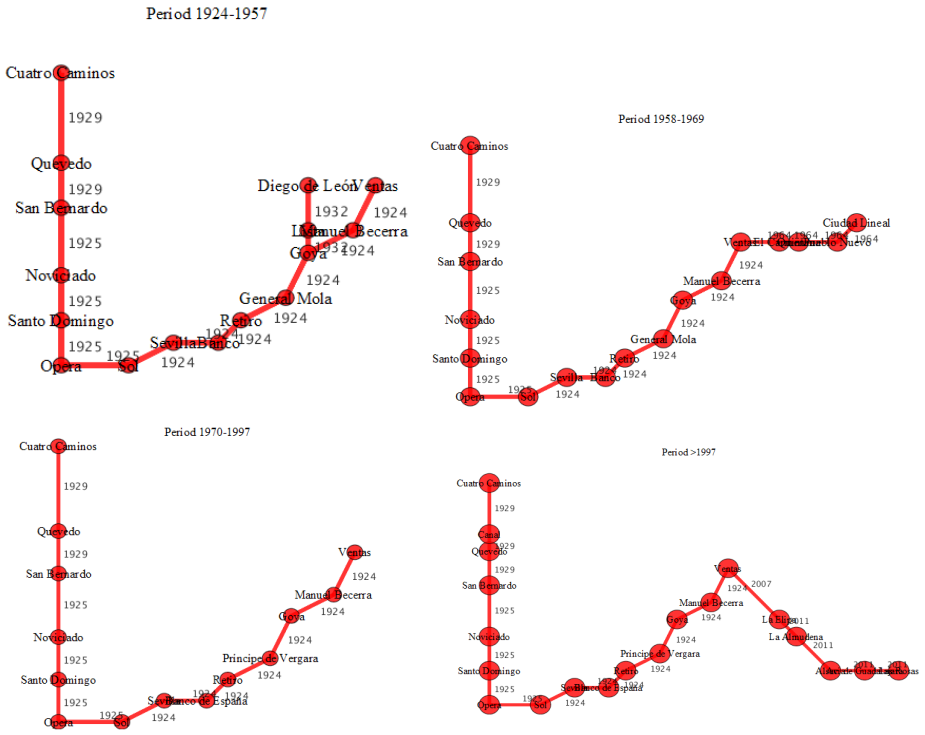

Figura 2.3: Los cuatro grafos (etapas) que forman la evolución de la Línea 4. Fuente: [7]

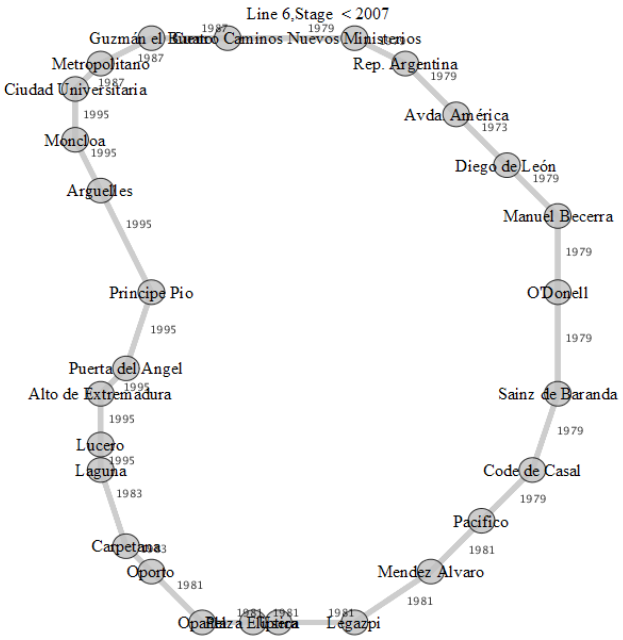

Figura 2.4: La primera etapa de la Línea 6, estado en 2006. Fuente: [7]

Cuadro 2.1: Número de etapas por línea de la red de Metro de Madrid

línea	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11
nº etapas	3	4	1	1	2	2	1	3	1	4	1

Ejemplo 2.2 *La evolución de la Línea 6 consta de dos etapas debido al evento crítico de la nueva estación intermedia del Planetario en 2007, entre Méndez Álvaro y Legazpi . La primera etapa es el estado en 2006 (Figura 2.4), que se corresponde con el período entre la inauguración y esta fecha. La segunda etapa sería el estado a fecha de hoy y se correspondería al período comprendido entre el 2007 y la actualidad (Figura 2.5).*

El número de etapas es pequeño porque las líneas generalmente se extienden sin eliminar aristas (consultar la Tabla 2.1).

Sintetizando, los datos de la cronología se pueden reordenar en una lista de etapas en formato de grafo. El diagrama de flujo del proceso se muestra en la Figura 2.6.

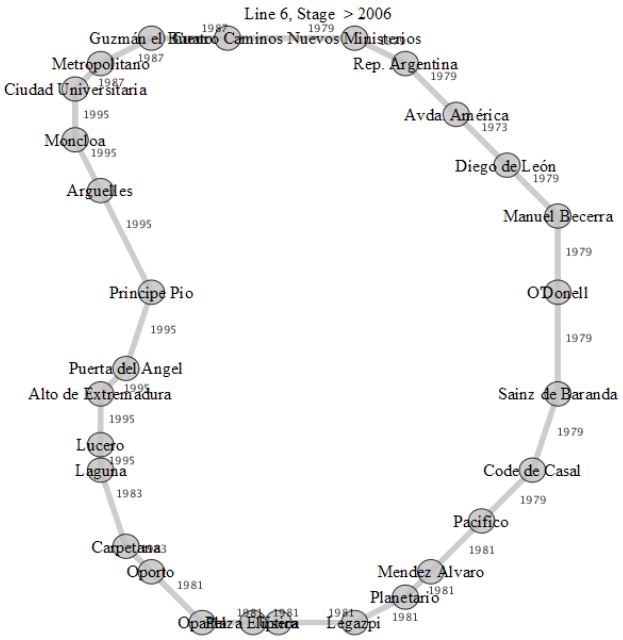

Figura 2.5: La segunda y última etapa de la Línea 6. Fuente: [7]

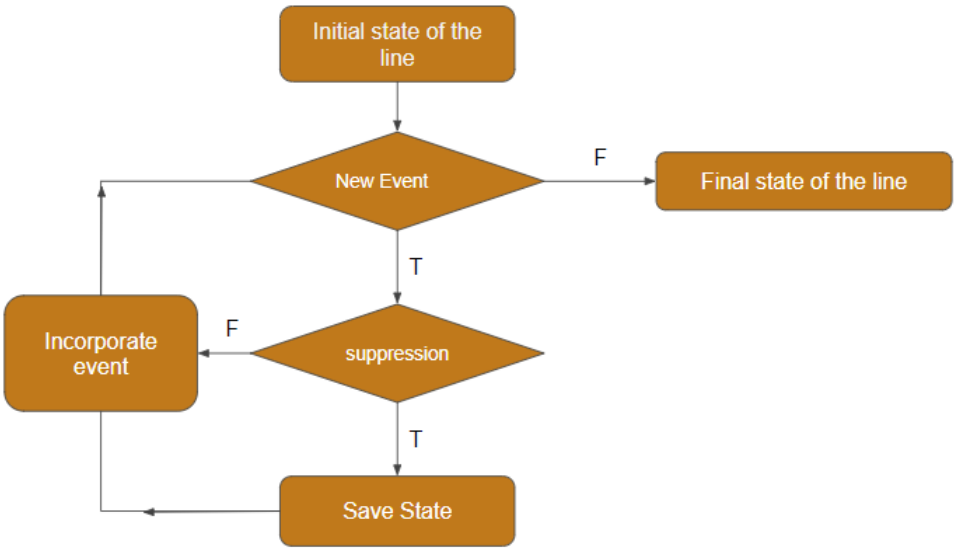

Figura 2.6: Ordinograma de la agrupación de la cronología en etapas. Fuente: [7]

2.3.3 Obtención de la red histórica

Una vez que la cronología se ha reorganizado en grafo-etapas, se obtiene de inmediato, en tres pasos, la red en un instante dado:

Paso 1, elección de etapa: Las etapas forman una partición de la línea temporal. Cada fecha pertenece a una única etapa.

Paso 2, la poda de la línea: Como entre dos etapas todos los estados son expansivos, sus aristas reproducen la cronología en ese período. Por lo tanto, eliminando todos los arcos posteriores a una fecha en la etapa elegida en el paso anterior (y los nodos de grado cero resultantes), se obtiene el estado correspondiente a esa fecha.

Ejemplo 2.3 *La Línea 6, como hemos visto en el Ejemplo 2.2, consta de dos etapas. Si queremos el estado de la L6 en 1982 escogeremos su etapa correspondiente (la* < 2007*), que podándola en esta fecha nos da el estado de la Figura 2.7.*

Figura 2.7: Estado de la Línea 6 en 1982.Fuente: [7]

Ejemplo 2.4 *La Línea 4, sin eventos críticos, está formada por una única etapa. Esta y su poda a 1982 se ven en la Figura 2.8*

Paso 3, la unión de los grafos: Una vez que se calculan las líneas en un momento determinado, sólo es necesario unir los grafos para conseguir la red en un instante dado.

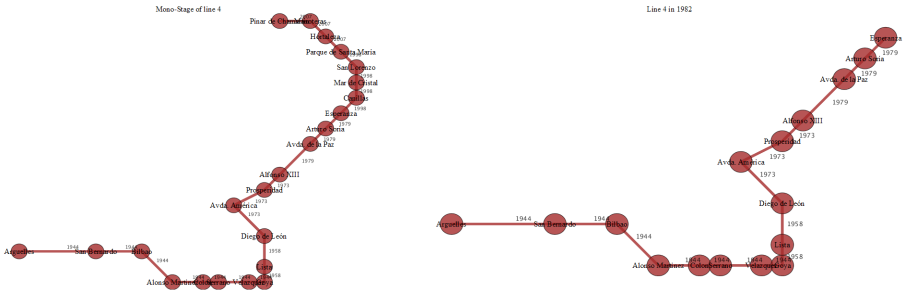

Figura 2.8: La monoetapa de la Línea 4 a la izda. A la dcha. el estado en 1982. Fuente: [7]

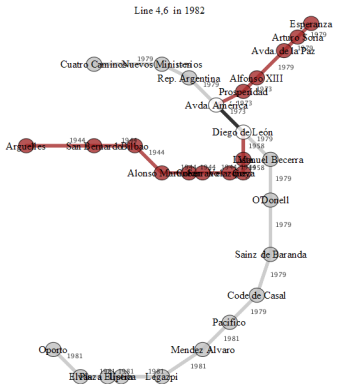

Figura 2.9: La subred unión de la L4 y la L6 en 1982. Fuente: [7]

Ejemplo 2.5 *Con los grafos de los Ejemplos 2.3 y 2.4 se puede ver el grafo unión de las líneas L4 y L6 en 1982. Figura 2.9.*

Recordar que existen multiaristas, al ensamblar los grafos se solapan. En el Ejemplo 2.5 se puede observar este hecho en el arco negro entre las estaciones de *Avda. de América y Diego de León* .

2.4 Resultados

Con la motivación de [40] y un original planteamiento se ha trabajado sobre la red de metro de Madrid. Se ha implementado, en el software ya comentado, un programa capaz de generar un mapa esquemático retrospectivo de todas las líneas de la red de metro de Madrid en un instante dado usando grafos (Figura 2.10).

2. GRAFO-CRONOLOGÍA: MAPAS ESQUEMÁTICOS RETROSPECTIVOS DE LA RED DE METRO MEDIANTE GRAFOS

Se presentó un póster en las Jornadas *FuzzyMAD18* y una contribución oral en la *24th Conference on Applications of Computer Algebra* (ACA2018). Además se publicó en 2019 un artículo en la revista *Mathematics in Computer Science* (citada en Scopus y Emerging Sources Citation Index) [7].

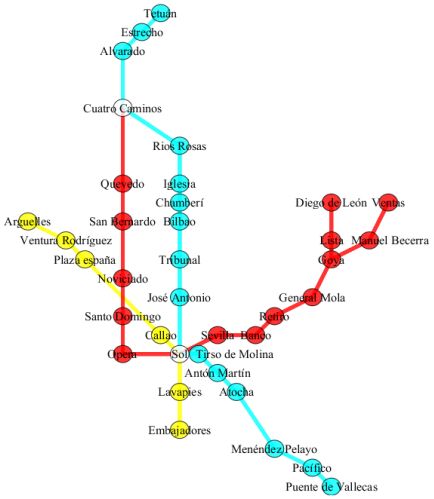

Figura 2.10: Grafo que representa al mapa esquemático retrospectivo de la red de Metro de Madrid en 1941 (los nodos comunes en blanco). Implementado en *Maple*. Fuente: [7]

En este desarrollo destacan de forma natural algunos estados, llamados etapas, que caracterizan toda la evolución de una línea. Cada etapa conserva toda la información de los estados intermedios hasta su etapa precedente (o el origen de la línea). Las etapas admiten la sencilla caracterización 2.2. Con estas etapas se puede recrear la red en cualquier instante, una vez que el histórico de los datos se almacena en etapas, viajar en el tiempo es recortar y pegar estos grafos. (Figura 2.11).

Por la necesidad de expansión de estas redes, un pequeño número de etapas puede caracterizar una gran secuencia de estados. Unas pocas etapas hacen posible llegar a cualquier estado intermedio por medio de la poda. La red en la actualidad se puede ver en la Figura 2.12. Aunque la mayor parte del plano actual es la extensión del año 1980, salta a la vista en este caso que ni el estado de la línea 5 (verde) ni de el de la línea 10 (azul oscuro) en 1980 se pueden inferir de su estado en la actualidad. La L10 en 1980 conectaba en Aluche con la L5, ahora es la L5 la que conecta en *La Casa de Campo* con la L10, que cedió secciones a la L5 y se

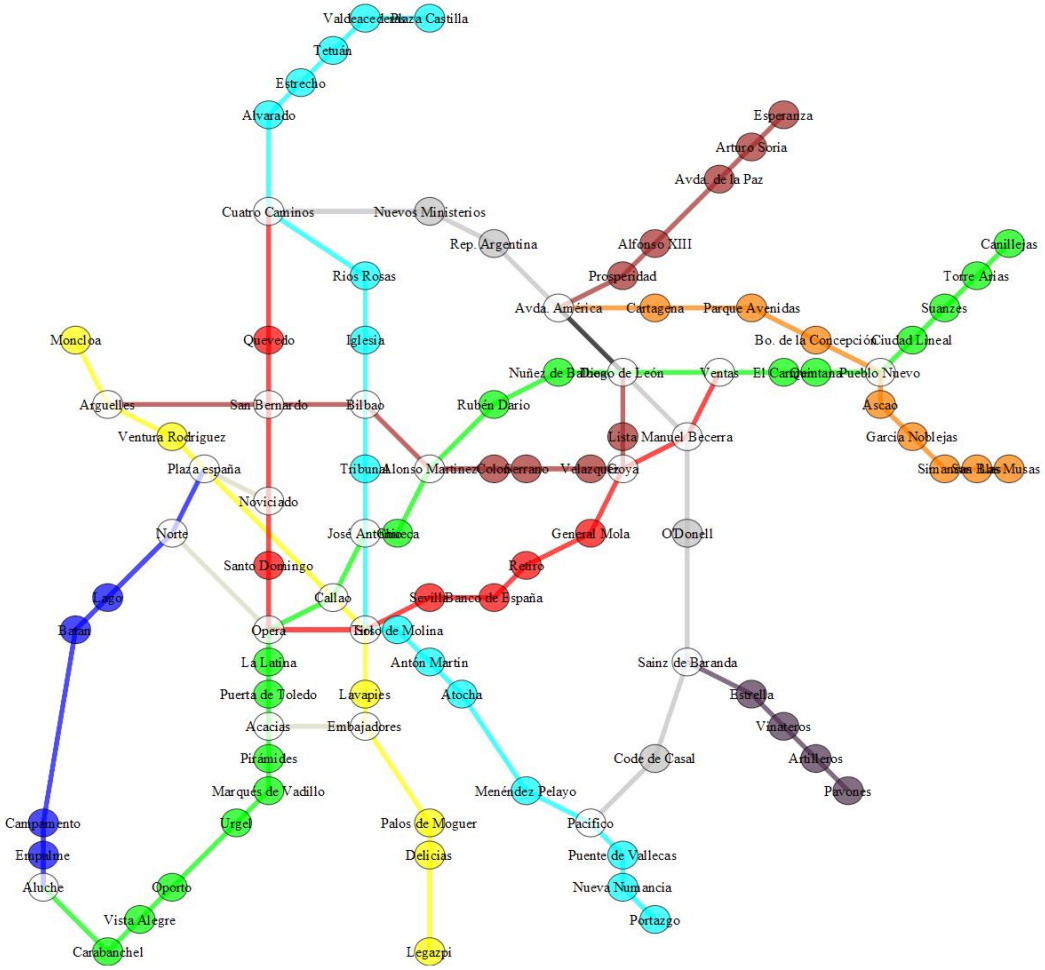

Figura 2.11: Grafo que representa al mapa esquemático de la red de Metro de Madrid en 1980 implementado en Maple. Fuente: [7]

redirigió a *Puerta del Sur*. Pero han ocurrido otros cambios aún más bruscos como la extinción de la antigua L8 (inaugurada para el Mundial82) en favor de la L10, para años más tarde renacer y llegar hasta *Barajas*.

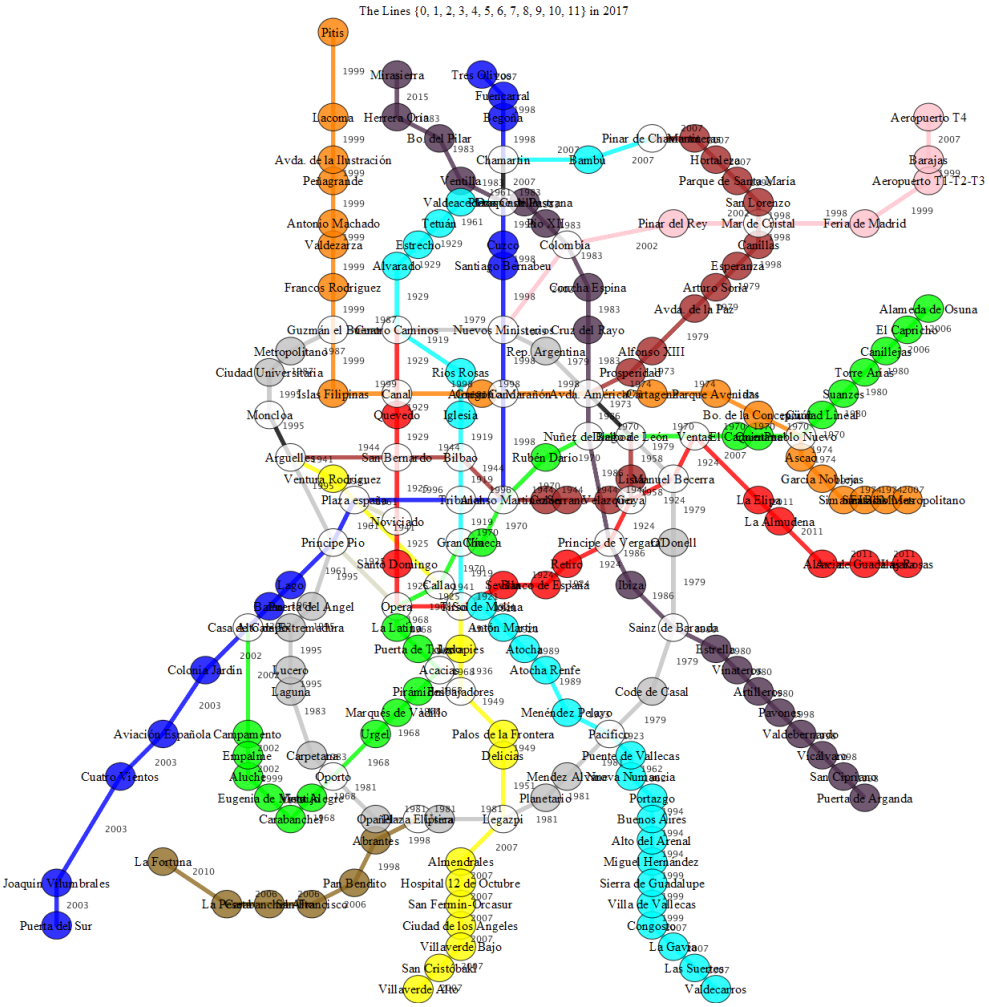

Figura 2.12: Grafo que representa al mapa esquemático de la red de Metro de Madrid en 2017. Implementado en Maple. Excluido Metro Ligero, MetroSur, MetroNorte, MetroEste y el cambio de tren de Puerta de Arganda. Fuente: [7]

Capítulo

3

Propuesta de un Gráfico 3D para representar la rapidez de las conexiones por FFCC

Con la diversidad de medios y velocidades en el transporte ha aumentado la competitividad entre estos servicios. En un mundo globalizado, al pasajero no le preocupa la distancia, le importa el tiempo invertido. Esto afecta directamente al ferrocarril. Con la llegada de distintos operadores y la oferta de alta velocidad, obviamente la clave está en la velocidad. Los mapas donde la distancia se transforma en tiempo muestran mediante deformaciones problemas globalmente. Basándose en esto, con un aumento de dimensión, este trabajo presenta gráficamente a través de superficies poligonales, como si fuesen perfiles de orografía, la rapidez de la conexión sin descuidar el contexto geográfico. El trabajo se ha programado en el software matemático *Maple* y consideramos que el tipo de diagrama obtenido puede ser un buen instrumento gráfico. Se aborda con dos transformaciones de la velocidad. El soporte matemático es elemental, pero la implementación es compleja, explotando las posibilidades del software utilizado.

3.1 Antecedentes

Una muestra reciente e ilustrativa de la preocupación por el diseño de gráficos aplicados a las redes ferroviarias se encuentra en la creación de mapas de isócronas radiales en 2012 [36, 39], fusión de dos esquemas muy utilizados desde el siglo XIX: los mapas de anamorfosis y los gráficos circulares (uno de los autores de estas referencias, y director de esta tesis, fue el impulsor de estos trabajos). La introducción de ideas o técnicas de otros campos en la ingeniería del transporte puede ser muy productiva.

Los gráficos circulares hacen uso de las superficies para comparar la información. El original y artístico[1] diagrama de área polar [43] es el ”dual” del universal ”queso” o gráfico de sectores. Éste último, el más conocido, pondera la superficie de los sectores, a amplitud variable y radio fijo. En cambio, el de área polar pondera la superficie de los sectores a radio variable y amplitud fija (Figura 3.1). Ambos gráficos circulares representan la frecuencia relativa de una variable cualitativa (en categorías o sectores) enfrentando superficies sectoriales.

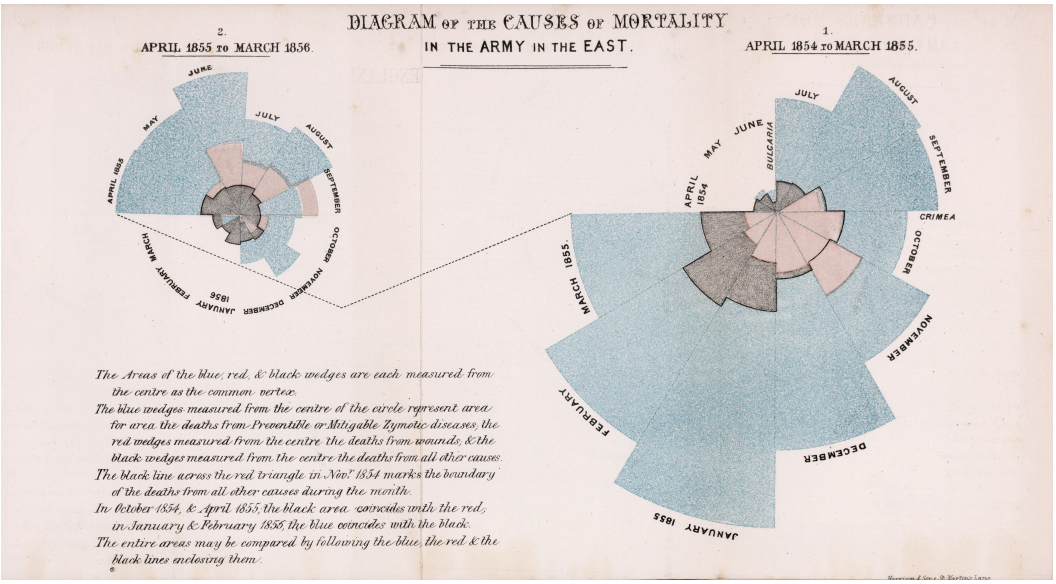

Figura 3.1: Diagrama fechado en 1858 por Florence Nightingale de un gráfico circular coloreado para ilustrar las causas de muerte en el ejército británico. Fuente: dominio público

[1] Utilizando el término artístico en el sentido de agitar el sentimiento o la imaginación

En los mapas de anamorfosis [16, 47, 49], los puntos geográficos conservan el rumbo geográfico desde un origen central o polo, pero sus distancias espaciales son transformadas a tiempos de viaje (Figura 3.2). El cambio de perspectiva en estos procesos de anamorfosis en el diseño de mapas presentan numerosas y llamativas variantes en múltiples campos (por ejemplo [17, 21, 22]).

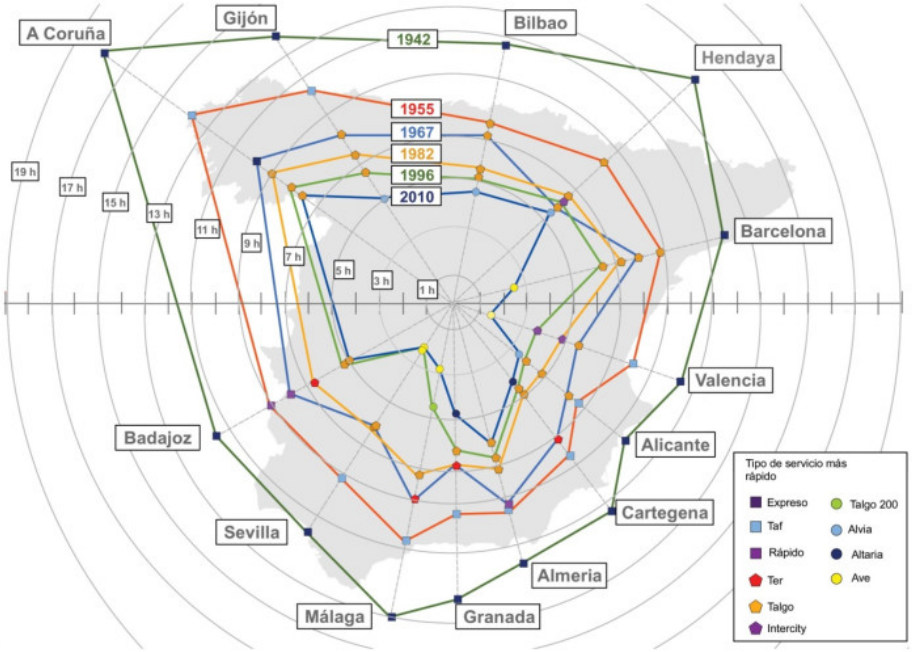

Figura 3.2: Mapa de anamorfosis de la red ferroviaria española (1942-2010). Fuente: Fundación de los Ferrocarriles Españoles.

Volviendo a los gráficos isócronos mencionados, las ciudades periféricas en torno a un polo constituyen las categorías del gráfico circular, la amplitud de los sectores es ponderada por el volumen de viajeros, mientras que el radio del sector, fijo en el gráfico de sectores, pasa a ser variable de forma que: el rumbo de la población orienta aproximadamente la posición del sector mientras que el radio de ese sector depende del tiempo. Estas representaciones gráficas han sido adoptadas por la Fundación de los Ferrocarriles Españoles en sus informes más recientes (Figura 3.3). Este original gráfico tiene un inconveniente, su radialidad le confiere un carácter local e impide comparaciones entre poblaciones periféricas.

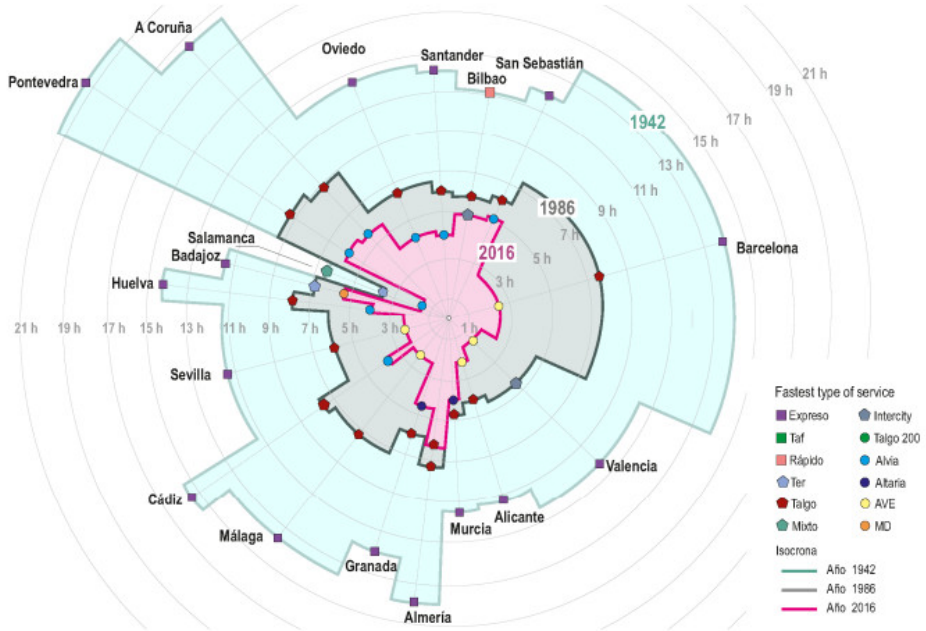

Figura 3.3: Tres mapas de sectores isócronos superpuestos (1942, 1986, 2016) de la red ferroviaria española. Fuente: Fundación de los Ferrocarriles Españoles.

Existen otras propuestas de mapas distorsionados para visualizar la velocidad en una red de ferrocarril [15, 23]. Recordemos también el uso de la simulación como herramienta para generar diversos esquemas y diagramas [4, 5, 26, 38].

Por último, los enfoques 3D para la visualización de grafos, como el que tratamos aquí, se usan generalmente para representar grafos de grandes dimensiones [30, 50] y visualización de grafos multicapa [31], enfoque utilizado en el primer capítulo de esta tesis[8]. No conocemos ningún software comparable que utilice la tercera dimensión para representar pesos de aristas en teoría de grafos.

3.2 Objetivo

La necesidad expuesta al principio del capítulo y la atracción por el diseño de la ”información rápida” proporcionada por un gráfico, ha motivado el diseño de un gráfico 3D para valorar visualmente el gran contraste de velocidades de conexión

por ferrocarril que existe en España. La idea es representar mediante perfiles escalonados unos niveles de penalización sin deformar nuestra orientación geográfica gracias a la tercera dimensión. Permite comparar cualquier ruta entre puntos arbitrarios del plano. La velocidad se ha transformado con una función h que levanta el grafo, y se han propuesto dos alternativas. Esta tercera dimensión, como veremos, nos permite pensar en dar valor a las superficies engendradas.

En el ejemplo tratado se tomarán siete líneas de trenes de Renfe en el area SW de España, cinco de ancho ibérico y dos de alta velocidad.

3.3 Método

Se ha programado en el software matemático *Maple*. Se usa el comando **polygonplot3d** del paquete **plots** que dibuja uno o más polígonos en el espacio.

La construcción es sencilla y consiste en dar relieve al mapa de la red. En el plano $z = 0$ se localizan las estaciones (nodos) a partir de sus coordenadas y se forma la red uniendo los nodos correspondientes a las líneas en servicio mediante aristas. En otras palabras, se crea en $z = 0$ una cartografía simplificada de la red de ferrocarril a estudiar (Figura 3.4).

Una vez se tiene este mapa base, se levanta en cada arista una altura h función de la magnitud a comparar, que al tomarse constante en cada sección de vía forma un rectángulo vertical entre nodos. El resultado son unos caminos poligonales tridimensionales (Figura 3.5). Es en esta magnitud h donde se diferencian los dos criterios alternativos propuestos para la obtención del gráfico y que a continuación se detallan.

3.3.1 Uso de la inversa de la velocidad

Es necesario partir de un plano geográfico, con cierta precisión en sus nodos, como se expondrá a continuación. El mapa esquemático de la red debe tener una oportuna precisión geográfica.

La altura que se levanta en este caso, para cada rectángulo, es $h = v^{-1}$, donde v es la velocidad media en esa arista . Su elección no es casual y está motivada por la integración discreta, en nuestro caso el valor de $h = \frac{\Delta t}{\Delta s}$ es constante en

3. PROPUESTA DE UN GRÁFICO 3D PARA REPRESENTAR LA RAPIDEZ DE LAS CONEXIONES POR FFCC

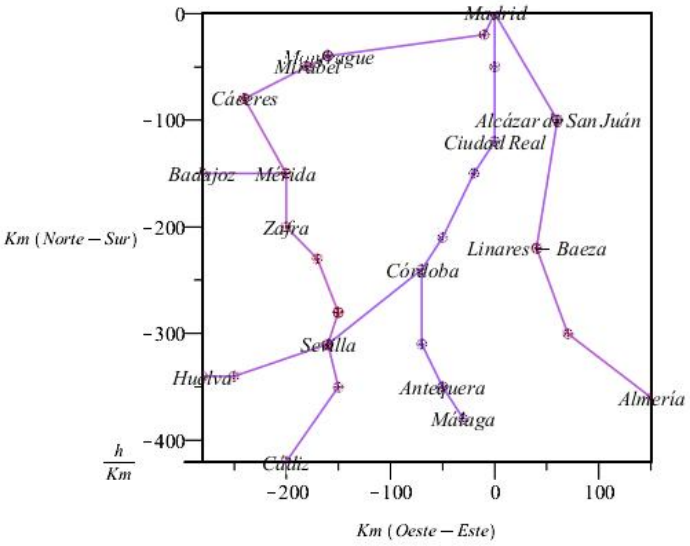

Figura 3.4: Vista cenital (z=0) donde se aprecia la parte de la red de Adif considerada. Fuente: elaboración propia (realizado en *Maple*).

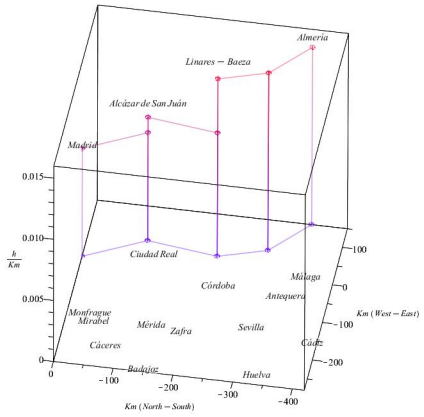

Figura 3.5: Representación 3D propuesta para el Talgo Madrid-Almería que realiza cuatro paradas. Fuente: elaboración propia (realizado en *Maple*).

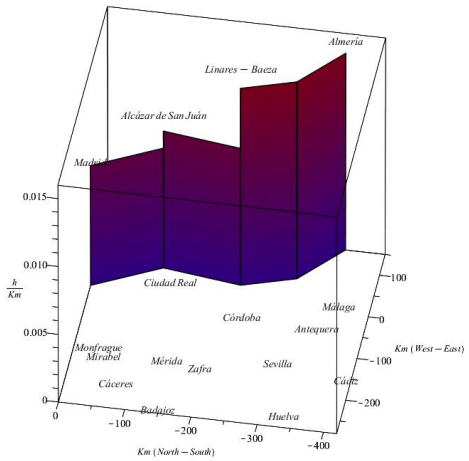

Figura 3.6: Representación 3D propuesta para el Talgo Madrid-Almería de la Figura 3.5 . Las alturas de los rectángulos son las inversas de las velocidades medias y la suma de las áreas de los rectangulos el tiempo empleado. Fuente: elaboración propia (realizado en *Maple*)

.

cada tramo. Esta altura h es el tiempo empleado por unidad de longitud. Sumando las áreas de los rectángulos de una ruta se obtiene el tiempo total empleado en recorrerla (Figura 3.6).

3.3.2 Uso de la diferencia de velocidades

Alternativamente tomando la diferencia $h = v_{max} - v$ como altura en esta construcción, se resaltan los perfiles con infraestructuras más degradadas sin deformar la velocidad. Como en el caso anterior v es la velocidad media entre nodos del camino y v_{max} es la velocidad media máxima alcanzada en los tramos de la red analizada y, en consecuencia, marca el plano $z = 0$ de la Figura. Es decir se representa para cada tramo la diferencia de velocidades entre la velocidad del tramo y la máxima alcanzada en la red.

3.4 Resultados

El paquete está implementado en el sistema de álgebra computacional *Maple*. Un póster de una primera versión de esta propuesta fue presentado en las Jornadas FuzzyMAD 2019 celebradas en la Universidad Complutense de Madrid. Posteriormente en 2020 el artículo titulado "A 3D proposal for the visualization of speed in railway networks" [6], con el contenido expuesto aquí, fue publicado en la revista AIMS Mathematics (citada en SCI-JCR, 83/330 Q2 T1 Mathematics 2020).

El artículo está ilustrado con gráficos elaborados a partir de datos reales obtenidos de los horarios de Renfe Operadora para siete líneas del SO de España, incluyendo la línea de alta velocidad Madrid-Sevilla (1992). La matemática subyacente es simple, como sucede en los artículos [36, 39], pero es una interesante herramienta visual en 3D que perfila la anatomía temporal de las rutas. La presentación gráfica de los datos, visual, es la forma más rápida de transmitir información compleja a un humano. No sabemos de ningún trabajo similar.

Los dos métodos presentados emplean distintas funciones h de la velocidad (Figura 3.7). El decrecimiento de h indica una penalización, proporcionando mayores perfiles a las velocidades más bajas. Se distingue:

- En el primer método es necesaria la exactitud de las distancias en las aristas del plano $z = 0$ para dar precisión a los tiempos como área de los rectángulos. En el segundo método no es necesaria esa exactitud, se pierde esa información temporal.

- Atendiendo a la pendiente de h, en el primer método se observa un decrecimiento rápido a velocidades bajas junto a un decrecimiento lento a velocidades altas . Todo esto no ocurre en el segundo caso que con una pendiente uno (negativa) la imagen de h de la velocidad conserva los incrementos ($\Delta h = -\Delta v$) (Figura 3.7).

Con una primera elección de $h = v^{-1}$, inspirada en la integración, se obtiene dinamismo en las superficies ya que el área determinada por cada segmento es el tiempo empleado en recorrerlo. El decrecimiento rápido de h a baja velocidad junto al decrecimiento lento a alta remarca una pauta para mejorar los tiempos. Fruto de

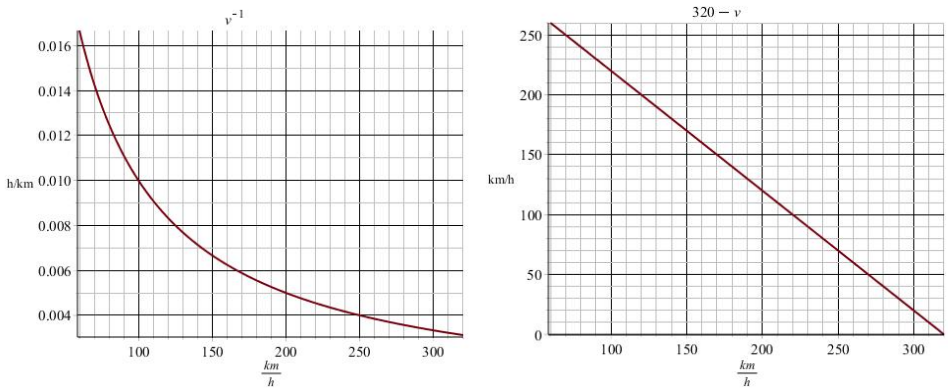

Figura 3.7: Transformaciones h utilizadas para la velocidad. Fuente: elaboración propia (realizado en *Maple*)

la proporcionalidad inversa, pequeños incrementos de baja velocidad ahorran más tiempo que grandes a alta (Figura 3.8).

Con la segunda elección de $h = v_{max} - v$ se mantiene un compromiso con la velocidad. La pendiente constante y unitaria conserva los incrementos de la velocidad de los trenes $\Delta h = -\Delta v$. Cada salto en el perfil es el incremento de velocidad en el camino (Figura 3.9).

3. PROPUESTA DE UN GRÁFICO 3D PARA REPRESENTAR LA RAPIDEZ DE LAS CONEXIONES POR FFCC

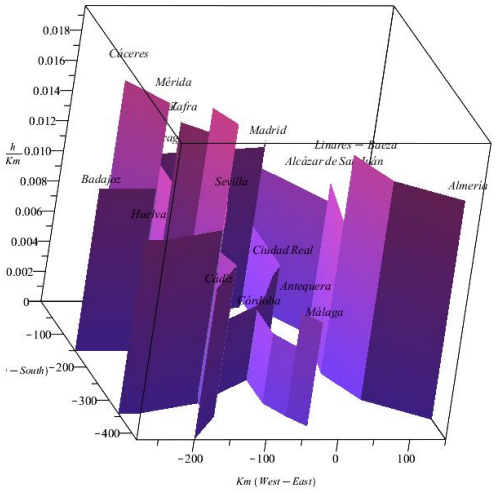

Figura 3.8: Vista 3D de la inversa de la velocidad. Estilo surface más apropiado para esta h. Fuente: elaboración propia (realizado en *Maple*)

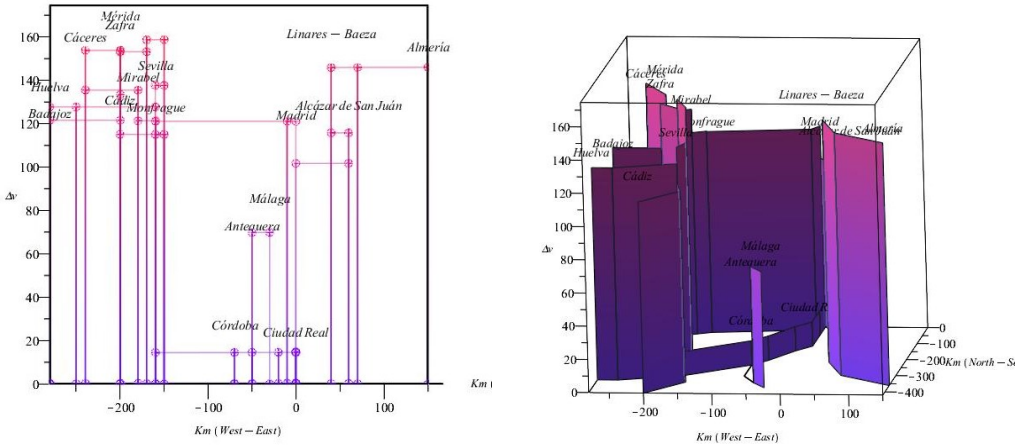

Figura 3.9: Otras dos vistas 3D para $h = v_{max} - v$. A la izda. proyectando en x=0 con estilo pointline y a la dcha. con estilo surface. Fuente: elaboración propia (realizado en *Maple*)

Capítulo

4

Una simulación en tiempo acelerado de colas de ventanillas de una gran estación de ferrocarril

La venta de billetes de tren ”on line” o de abonos ha agilizado el acceso a este transporte. Aún así, en muchos casos se sigue expendiendo en la propia estación, ocasionando largas esperas en determinadas ocasiones o circunstancias. Hay abundantes trabajos sobre simulaciones tratando los movimientos de pasajeros en las estaciones de ferrocarril, como se detalla más adelante, pero este trabajo se centra en concreto en el modelado de las colas de venta de billetes en una gran estación de ferrocarril española, donde se prestan cuatro servicios diferenciados: “Alta Velocidad”, “Larga Distancia”, “Media Distancia” y “Cercanías”. Se consideraran el uso de ventanillas de ”último minuto” y compartidas (dónde se venden billetes para varios servicios). Se realizan simulaciones con ejemplos de distintas composiciones de las ventanillas. Desde el punto de vista teórico no hay novedades, pero por su singularidad y su capacidad de calcular fácilmente las ventanillas necesarias para

garantizar tiempos de espera razonables, consideramos que este trabajo tiene interés práctico. Se mencionarán además otras posibles extensiones de este trabajo. No hemos encontrado otro software dedicado específicamente a este objetivo en particular.

4.1 Antecedentes

Existe una gran bibliografía de estudios y soluciones computacionales aplicadas a las terminales aeroportuarias. Restringiéndose al flujo de pasajeros, también los hay desde distintas perspectivas: [10], [27], [51], [45]...

El flujo de pasajeros en las estaciones de ferrocarril es más simple que en un aeropuerto, ya que, por ejemplo, no se asignan mostradores de facturación, no hay fuertes controles de seguridad y embarque ni, en consecuencia, una segunda línea de tiempos de espera en controles y puertas de acceso.

Como en los capítulos anteriores, no existen muchos trabajos en el campo del estudio del flujo de pasajeros aplicado a las estaciones de ferrocarril. Por la proximidad a nuestro objetivo se pueden mencionar las siguientes referencias:

- 'Train station passenger flow study'[28]: Enfocado a un diseño eficiente en las líneas de espera se desarrolla un modelo típico de simulación de estación de tren, siguiendo los movimientos reales de los pasajeros.

- 'Microscopic simulation on ticket office of large scale railway passenger station'[29]: Se modeliza con el software de simulacion SIMIO la emisión de billetes en una gran estación de trenes de pasajeros en una ciudad china. Con varios modelos experimentales se valora eficiencia y servicio.

- 'Modeling pedestrian movement at the hall of high-speed railway station during the check-in process'[48]: Se utiliza un modelo de autómata celular para simular la agitación de los pasajeros en el vestíbulo de una estación de ferrocarril durante la expedición de billetes para un tren de alta velocidad. Permite estudiar el complejo comportamiento del movimiento de los pasajeros hasta el embarque.

- 'Possibilities of Simulation Tools for Describing Queuing Theory and Operations Service Lines in Railway Passenger Transport'[33] : Modelo estocástico de teoría de colas con tiempos de espera en las diferentes colas ante ventanillas de venta de billetes. Se cuantifica la eficiencia con ayuda de la simulación.

El trabajo presentado aquí tiene en común con estas referencias el conseguir una simulación realista del flujo de pasajeros en una estación de tren durante la tramitación de billetes, pensando principalmente en la previsión y la eficiencia. El proceso, la implementación y el marco es propio y original, estando enfocado al caso español.

En una gran estación de ferrocarril la venta de billetes para los servicios de alta velocidad (AVE), larga distancia (LD), media distancia (MD) y cercanías (C), pueden tener un amplio abanico de ventanillas variables asignadas a cada servicio, donde una ventanilla puede atender varios servicios, además de las ventanillas de "último minuto" (UM) que rompen el orden natural de llegada. Dado que no conocemos ningún enfoque que reproduzca esta situación, tratamos de obtener una herramienta original de simulación en tiempo acelerado. Esta herramienta es útil para predecir, conocido el número y la composición de las ventanillas compartidas, la evolución de las colas en los mostradores de venta de billetes.

Por último citamos dos trabajos más a tener en consideración por la experiencia directa del director de la tesis en este campo. El primero es un proyecto de investigación desarrollado para el aeropuerto de Málaga 'An accelerated-time simulation of departing passengers' flow in airport terminals' [38]. El segundo es 'An accelerated-time microscopic simulation of a dedicated freight double-track railway line'[26], desarrollado en cooperación con la Fundación de los Ferrocarriles Españoles.

4.2 Objetivo

Existen las tarjetas de transportes y la compra de billetes ”on line”, que sólo precisan validación, pero sigue siendo posible y habitual la venta de billetes tanto en taquilla como en máquinas expendedoras. Es por tanto muy útil poder modelizar el flujo de pasajeros en el acceso a este transporte de masas, especialmente en horas punta, en fechas de extraordinaria afluencia o con medidas excepcionales como una pandemia.

Se plantea el problema como una máquina virtual en tiempo discreto. El tiempo discreto sólo limita la precisión que se desee, rebajando la complejidad. En cuanto al término de máquina, se puede formular el problema como un autómata finito $A = \langle E, S, Q, f, g \rangle$:

1. E es el conjunto de entrada: cada uno de sus elementos viene dado por la lista de pasajeros que llegan a por su billete, independientemente de su tren (cola única virtual). Cada pasajero adjunta sus atributos.

2. S es el conjunto de salida: cada elemento es la lista de pasajeros que salen al andén. Cada uno con sus atributos.

3. Q es el conjunto de estados: cada estado es una lista de colas de ventanilla (Cada pasajero mantiene unos atributos de estado).

4. $f : E \times Q \longrightarrow Q$ función de transición (paso de reloj en ventanilla): incorpora las entradas a las colas de ventanillas, elimina de las colas a los pasajeros atendidos y cambia de cola a ciertos pasajeros. Modificando en consecuencia los atributos de los pasajeros.

5. $g : E \times Q \longrightarrow S$ función salida (paso de reloj en andenes): genera la lista de salida, los pasajeros atendidos. Cambia los atributos de los pasajeros.

6. Si en el instante t, A recibe la entrada $e_t \in E$ en un estado $q_t \in Q$, entonces:

$$q_{t+1} = f(e_t, q_t) \tag{4.1}$$

$$s_{t+1} = g(e_t, q_t) \tag{4.2}$$

Un esquema de funcionamiento de esta máquina en un paso temporal se puede ver en la Figura 4.1. f estaría definida como el estado resultante de la concatenación de operaciones $elección\ de\ ventanilla \longrightarrow reloj \longrightarrow urgencia$ sobre un estado q_t cuando recibe una entrada e_t. Por su parte, g sería una función de la cadena de instrucciones $elección\ de\ ventanilla \longrightarrow reloj$. La instrucción $urgencia$ modifica las colas con cambios de 'último minuto'.

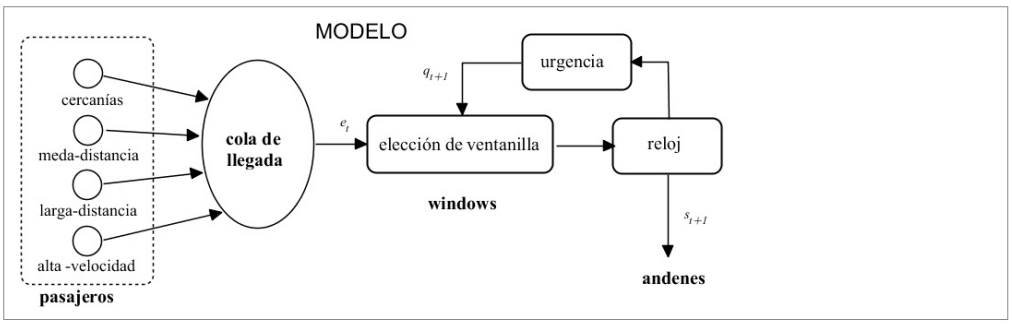

Figura 4.1: Esquema de la dinámica de expedición de billetes al acceder a una estación de tren

Utilizando como entradas las curvas de afluencia de pasajeros, los servicios por ventanillas y sus tiempos de atención, se hace correr la máquina sobre un horizonte temporal. Así se pueden obtener toda clase de estadísticas y consecuencias en lo referente al flujo de pasajeros por las taquillas: tiempos de espera, pasajeros que pierden el tren, ineficiencia de las ventanillas, pasajeros de último minuto, ventanillas necesarias, intensidades, etc..

La simulación se implementa en el software *Maple*.

4.3 Método

Esta formulación del problema necesita ser precisada para entender, programar y explorar sus capacidades. El anterior algoritmo hay que transformarlo en un programa. Se empezará definiendo los principales tipos de datos:

- pasajero: lista de atributos. Un atributo puede ser cualquier característica de interés del pasajero, como la hora de llegada a la estación, identificador de su

tren, servicio requerido, tiempo restante de atención, hora en ser despachado, ventanilla, etc. Puede haber pasajeros con idénticos datos.

- cola de entrada: lista de pasajeros. En nuestro caso es una única cadena que contiene de forma ordenada a todos los pasajeros que llegan a adquirir un billete en un instante dado, independientemente de su tren.

- cola de ventanilla: lista de pasajeros. Los pasajeros son usuarios de un tipo de servicio que cubre la ventanilla. Puede estar vacía.

- estado de la máquina: lista de colas de ventanillas. Hay tantas colas como ventanillas .

4.3.1 Instrucciones

El programa se divide en los siguientes bloques de instrucciones:

- Llegada de pasajeros. En cada instante se suma la llegada de los pasajeros de todos los servicios, cada uno conforme a una distribución o curva de afluencia. Los pasajeros tienen unos atributos que permite su seguimiento. Es la entrada de datos.

- Fila única: Todos los pasajeros que llegan en un mismo instante son aleatoriamente ordenados en una fila virtual única. Se supone que no existen prioridades antes de la elección de ventanilla.

- Movimiento de ventanillas: Realiza los cambios en la configuración de las ventanillas a lo largo de la simulación. Están registrados previamente en un conjunto de movimientos. Cambia el entorno de la máquina.

- Elección de ventanilla: Los servicios que se asignan a una ventanilla puede ser único, compartido o ninguno. El criterio de elección de ventanilla utilizado por el pasajero es el habitual: cada individuo de la fila única, por orden, se coloca en la ventanilla con la menor cola que cubra su servicio. Se añade el atributo de tiempo de atención en ventanilla a los nuevos pasajeros en cola.

- reloj: Cada tipo de pasajero tiene un tiempo medio de atención. Cada paso de reloj se modifica el atributo de tiempo restante de despacho de cada pasajero atendido en ventanilla. Se extrae a los pasajeros que han sido completamente atendidos.

- urgencia: Algunos pasajeros de las colas, por sus circunstancias particulares, pueden cambiar de cola. Sólo se ha contemplado la situación de último minuto, con ventanillas especiales habilitadas para los usuarios cuyo tren tenga salida inminente. Quedan excluidos, en principio, los usuarios de cercanías por la alta frecuencia de estos trenes.

El diagrama de flujo de la Figura 4.2, con las instrucciones anteriores, representa el programa de la máquina ya diseñada.

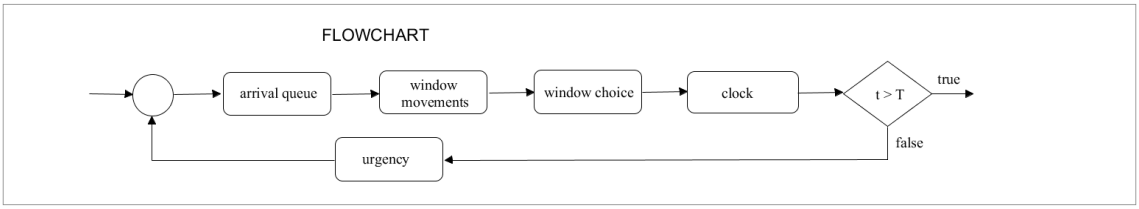

Figura 4.2: Diagrama de flujo de expedición de billetes al acceder a una estación de tren

El código del intérprete (en este caso de muy alto nivel) y la simulación hacen el resto. Se ajustan parámetros como número de ventanillas, asignación de estas, tiempos de atención, etc. Tras realizar una simulación, los datos obtenidos se pueden mostrar o graficar.

4.4 Ejemplo

En este ejemplo se ha considerado el intervalo de tiempo que va desde las 19:00 hasta las 20:00, con un paso de reloj de medio minuto. Hay en esta hora, nueve trenes de cercanías, dos de media distancia, uno de largo recorrido y un AVE. Se puede cambiar sin dificultad el número de trenes o ventanillas (son parámetros, véase Apéndice D). Se consideran 20 ventanillas numeradas con la siguiente distribución (los números entre llaves son los identificadores de ventanilla):

- cercanias={1,2,4,5,6,7,8,9}
- media distancia={10,11,12,13}
- larga distancia={11,12,13,14}
- velocidad alta={13,14,15,16}
- último minuto={17,18,19,20}

Nótese que los conjuntos anteriores no son disjuntos, hay ventanillas que comparten servicios (obviamente no para último minuto). La ventanilla número 3 no aparece por no estar operativa.

4.4.1 Curvas de afluencia y fila única

La carga de pasajeros de un tren es una lista que hay que introducir específicamente para cada tren, el valor k de la posición i-ésima en esta lista indica el número de pasajeros que llegan en el intervalo i-ésimo anterior a la salida de su tren (se detalla en el Apéndice D). No obstante en este ejemplo se hace uso de un procedimiento que genera automáticamente los pasajeros de cada tren mediante simulación de ciertas distribuciones estadísticas.

La obtención de estas curvas se sale del interés del trabajo y, pese a su importancia, no han sido objeto de estudio. No obstante, a continuación se explica brevemente cómo se ha generado la curva de afluencia de los pasajeros a cada tren. Se ha tomado la misma distribución de afluencia de pasajeros para los trenes que prestan el mismo tipo de servicio. Cada pasajero de cada tren viene dado por una distribución discreta y finita (Figura 4.3) de la antelación con la que llega en los instantes previos a la partida del tren. La curva de afluencia de cada tren se genera con una multinomial de esta distribución de tamaño el número de pasajeros que subirán a ese tren, número también aleatorizable. En el caso real los gestores proporcionan estas curvas a partir de encuestas en la propia terminal, como ocurrió en [38].

Primero se reúne la llegada de pasajeros de todos los trenes en un instante a través de sus correspondientes curvas de afluencia (Figura 4.4). Después se aleatorizan en una fila única virtual que permite el proceso de elección de ventanilla en

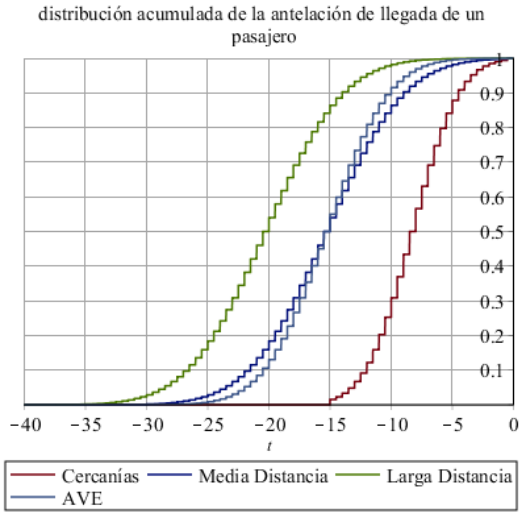

Figura 4.3: Funciones de distribución de la antelación de un pasajero, en minutos. Cero es el momento en que parte el tren. Cada color corresponde a un tipo de servicio.

cada instante. Por ejemplo en el instante simulado de las 19:39:30 (minuto 1179.5), donde llegaban tres viajeros de C, dos de MD y otros dos de LD, se crea la lista de llegada de pasajeros siguiente (almacenada en la entrada $e[t] \in E$, siendo e una tabla indexada en minutos):

```
> e[1179.5];

[["M0002", 1179.5, 1190.0], ["C0004", 1179.5, 1185.0], ["L0001", 1179.5, 1200.0],
 ["C0041", 1179.5, 1190.0], ["C0041", 1179.5, 1190.0], ["M0002", 1179.5, 1190.0],
 ["L0001", 1179.5, 1200.0]]
```

4.4.2 Transición de estados

Como hemos visto en la formulación, el cambio de estado consta de tres fases, a continuación se detalla el proceso entre el estado de las 19:39:30 y el siguiente de las 19:40:00. Señalemos que el tren de MD de las 19:50:00 entra en 'último minuto'. El estado a las 19:39:30 es el de la Figura 4.5.

fase 1 Elección de ventanilla: al estado de la Figura 4.5 se le añaden las nuevas incorporaciones de viajeros, en este caso como se ve en la Figura 4.6, tres

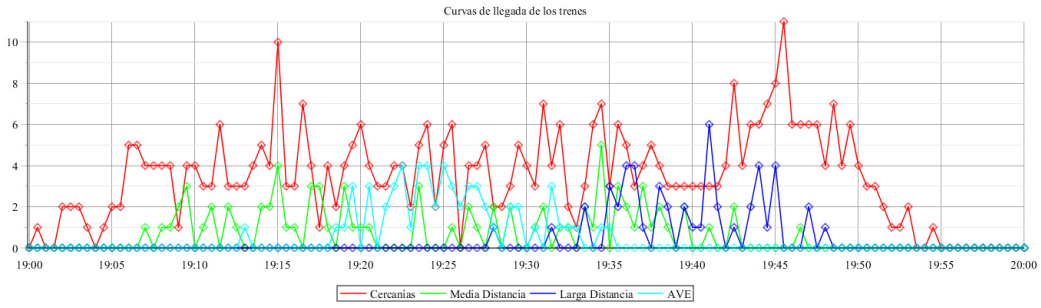

Figura 4.4: Curva de afluencia instantánea, por tipo de servicio, a lo largo de la franja de simulación. Cada tipo de servicio suma la afluencia de pasajeros de todos sus trenes.

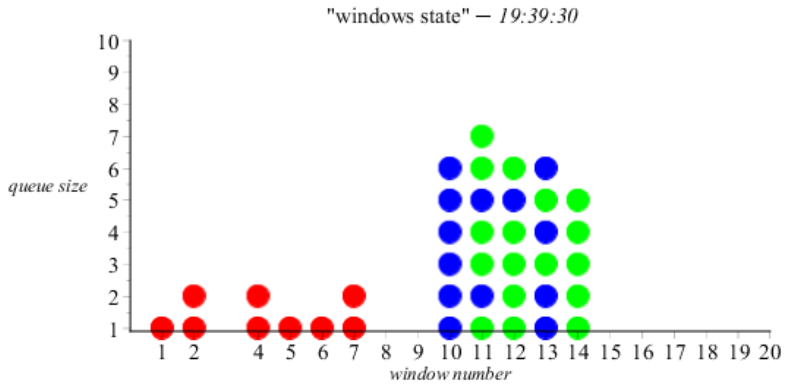

Figura 4.5: Estado de las ventanillas a las 19:39:30. Gráfico que representa en el eje horizontal el identificador de ventanilla y en el eje vertical el tamaño o número de personas en la cola de esa ventanilla. Cada bola representa a un pasajero en espera. En rojo los usuarios de C, en azul los de MD, en verde los de LD y en cyan AVE

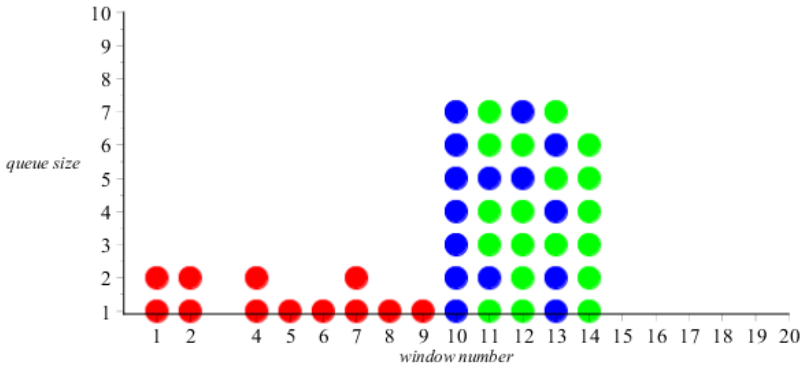

Figura 4.6: ventanillas en la fase1.

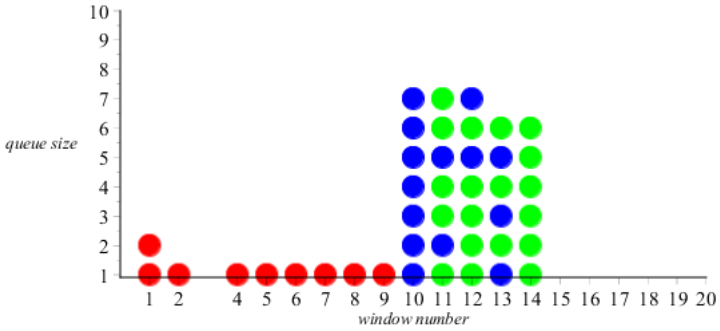

Figura 4.7: ventanillas en la fase2.

de cercanías, dos de MD y otros dos de LD. Aparece el atributo de pasajero ”tiempo restante de atención” a estas nuevas incorporaciones.

fase 2 Reloj: ahora corre el reloj un paso extrayendo a los despachados , saliendo tres de C y uno de MD (Figura 4.7). Salen los pasajeros cuyo atributo ”tiempo restante de atención” es nulo y reduce un paso al resto de las cabezas de cola.

fase 3 Urgencia: una vez adelantado el reloj y salido a los andenes los pasajeros con billete, los que quedan en cola, si procede, cambian a la cola de ”último minuto” antes de la entrada de nuevos pasajeros. El nuevo estado 19:40:00 es el que aparece en la Figura 4.8. En este caso hay un traspaso de 10 viajeros del MD de las 19:50:00 a las ventanillas de ”último minuto” (17,18,19,20).

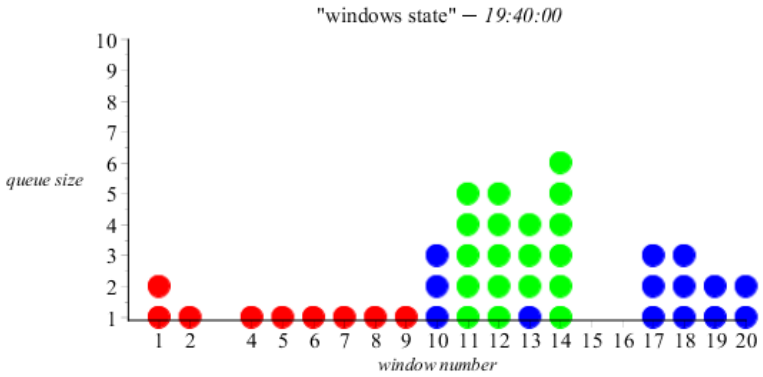

Figura 4.8: ventanillas en la fase3 o nuevo estado 19:40:00. Algunos pasajeros ya han pasado a "último minuto"

4.4.3 Información y Gráficos

Todo el proceso que se vio en 4.2 queda registrado en cada instante en el conjunto $\{(e_t, q_t, s_t) : \forall t\} \subset E \times Q \times S$ que se almacena en tres tablas indexadas en minutos: e, q y s. Ya hemos visto un ejemplo $e_t \in E$ de entrada de datos, s_t tiene la misma forma aunque pueda variar algún atributo. En cuanto a q, en la subsección anterior ya hemos visto varios gráficos que representan los estados q_t de las ventanillas (q_t incluye todos los atributos, conteniendo una información más completa, se puede ver su forma en el Apéndice D).

En cuanto a la evolución de las colas, sin entrar en análisis, se pueden graficar agrupando oportunamente los datos anteriores. En la Figura 4.9 se compara el tamaño total de las colas por tipo de tren. En este gráfico se aprecia el peso global de cada tipo de tren en las colas y se comprueba en este caso (aunque se puedan solapar viajeros del mismo tipo de servicio y distinto tren) que ningún pasajero de MD y LD ha perdido su tren, gracias a la actuación de las ventanillas de UM, aunque no se aprecie en este gráfico.

En el gráfico de la Figura 4.10 se compara el tamaño de la cola por tipo de ventanilla, en este caso las de MD frente a las de UM. Recordar que en este ejemplo las ventanillas de MD tienen en común tres de ellas con LD y una con AVE y que existía otra ventanilla más para LD. Se observa cómo influye un tren sobre el tipo de ventanilla cuando se dispara la alarma de las ventanillas de UM. En particular se

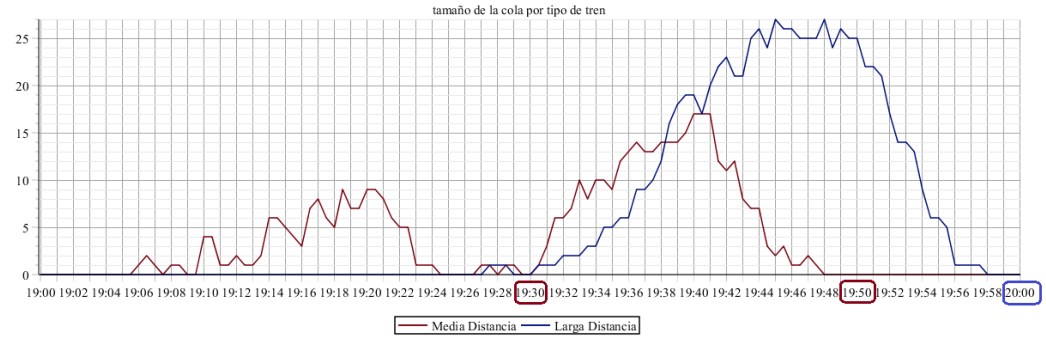

Figura 4.9: Tamaño total de las colas de los trenes de media y larga distancia a lo largo de la hora simulada. Se han encuadrado las horas de salida de los trenes involucrados.

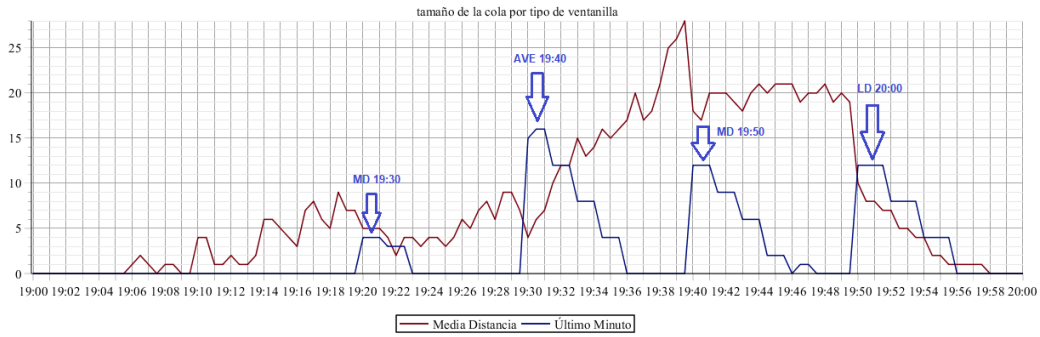

Figura 4.10: Tamaño total de las colas de las ventanillas que cubren MD, con ventanillas comunes a LD y AVE, a lo largo de la simulación. Se contrasta con el tamaño total de las colas de las ventanillas de Último Minuto, señalando el tren que provoca el disparo de UM.

nota el peso del tren de LD en el pico de estas ventanillas a las 19:40 (el tren AVE ya había salido y sus viajeros no podían ocupar las ventanillas comunes con MD).

4.5 Resultados y conclusiones

El paquete está implementado también en *Maple*. Este trabajo ha sido publicado en la revista Mathematical Problems in Engineering en 2021 (citada en SCI-JCR, 84/108 Q4 T3 Mathematics, Interdisciplinary Applications 2020) bajo el título de ”An Accelerated-Time Simulation of Queues at Ticket Offices at Railway Stations” [9].

Simulando la franja horaria se almacenan las entradas, los estados y las salidas. En cada instante los pasajeros de entrada, las listas de pasajeros de las colas y los pasajeros de la salida, con sus atributos, nos dan toda la información necesaria para el propósito deseado. Al igual que ocurre con la obtención de los datos de entrada, el análisis queda fuera del objetivo de este trabajo, que es la máquina como herramienta generadora de datos y gráficos.

Puede ser un paquete útil para planificar la composición de ventanillas en los vestíbulos u oficinas de venta de billetes en grandes estaciones de ferrocarril, considerando las ventanillas compartidas. Es un enfoque simple (autómata finito en un entorno variable) que utiliza una lista de listas, se hace posible tratar cualquier combinación de ventanillas y las ventanillas de último minuto (UM).

Permite componer o distribuir las ventanillas para minorar el tiempo de espera máximo, equilibrando el sistema de intereses entre la empresa de transporte y los viajeros.

Pensando en la eficiencia, las ventanillas UM resultan reveladoras. Sirven de válvula de seguridad para que un pasajero no pierda su tren. Son un indicador para añadir o eliminar ventanillas, según la necesidad y el instante, como se veía en la Figura 4.10. Pero a pesar del control que ejercen las ventanillas de UM, no garantizan menores tiempos de espera, sólo no perder el tren. Los tiempos de espera se garantizarían con límite de cola, en este sentido cabe pensar como complemento al presente trabajo, y a efectos de optimización, en el uso de ventanillas de saturación que aligeren a las ventanillas que alcancen cierta longitud de cola.

Capítulo

5

Conclusiones y perspectivas futuras

A riesgo de resultar repetitivo, en estas breves secciones se recapitulan las conclusiones y se proponen posibles trabajos futuros surgidos a raíz de la aportación correspondiente.

5.1 Conclusiones

El desarrollo de las aplicaciones que forman los tres primeros capítulos está basado en la teoría de grafos. Esta flexible herramienta cuenta con paquetes en los software científicos más extendidos y se ajusta muy bien a la actividad esquemática, y en particular, como se ha visto, a una red ferroviaria o de transporte. Las respuestas a los problemas planteados de optimización y gestión en estas redes se han obtenido reduciendo una estructura superior a otras estructuras más sencillas, los grafos.

Primero se ha mostrado cómo se reduce un multigrafo, formado por nodos que admiten varios estados, a un grafo multicapa. Con esta descomposición se logra, con un algoritmo estándar, una ruta óptima para un tren de ancho variable en una red con tres tipos de infraestructura, dos anchos de vía y cambiadores de ancho.

Después se ha tratado la estructura de una red metropolitana, formada por una ”unión” de grafos distinguidos (las líneas) a través de unos nodos comunes (estaciones donde se pueden producir transbordos). La cronología de la evolución de cada línea se puede transformar a una secuencia de ciertos grafos que se llamaron etapas. Con estas etapas se consigue el estado de esta red en cualquier instante. Por la naturaleza generalmente expansiva de estas redes muy pocas etapas (grafos) caracterizan toda la evolución.

En el tercer capítulo se actúa al contrario, partiendo de un grafo que representa un mapa esquemático de nuestra red de ferrocarril, se crea un original diseño elevando el grafo con una tercera dimensión, lo que permite visualizar intuitivamente las velocidades medias en las distintas secciones de las líneas.

El último capitulo no se ha tratado con teoría de grafos. Es una simulación de colas que nos permite gestionar las ventanillas de una gran estación para, entre otros usos, minorar el tiempo de espera máximo. Las ventanillas de último minuto suministran un seguro de no perder el tren pero no garantizan menores tiempos de espera.

5.2 Perspectivas futuras

El trabajo descrito en el primer capítulo, ”Ruta mínima en la red de Adif: dos anchos de vía en varias infraestructuras y cambiadores de ancho”, detalla una nueva aproximación al problema tratado en [24, 37], y cuya principal aportación respecto del citado trabajo es la propuesta de un algoritmo subyacente totalmente distinto y muy simplificado. Desde el punto de vista de la teoría computacional no se prevé pues intentar nuevas mejoras. En cuanto a considerar más anchos de vía, aunque hay redes con tres anchos de vía, como es el caso, por ejemplo, de España, Argentina o Australia, no sabemos de la explotación en ningún país de material móvil apto para tres anchos de vía (posiblemente no hay espacio físico para poder recolocar las ruedas y su timonería de freno en tres posiciones distintas) o del uso en la misma red de distinto material móvil apto para dos pares de anchos diferentes. Sí sería razonable intentar portar la implementación a un CAS gratuito, por las facilidades de acceso al software desarrollado que proporcionaría. El CAS Maple

fue elegido por su potencia, facilidad de programación y diversidad de paquetes complementarios incorporados.

El trabajo detallado en el segundo capítulo, ”Grafo-cronología: mapas esquemáticos retrospectivos de la red de metro mediante grafos”, está también relacionado con un trabajo previo para redes de ferrocarril [40]. La adaptación a redes de ferrocarril metropolitano ha hecho recomendable incluir un enfoque distinto, basado en considerar ”líneas”, que ha supuesto el diseño, desarrollo e implementación de nuevos algoritmos. Como ya se ha explicado adecuadamente en el capítulo correspondiente, el proceso no es directo por la desaparición, absorción o reorganización de líneas. Como en el caso anterior, sería interesante intentar portar la implementación a un CAS gratuito y hacer más amable la introducción de datos.

El trabajo resumido en el tercer capítulo, ”Propuesta de un Gráfico 3D para representar la rapidez de las conexiones por FFCC”, presenta cómo se pueden apreciar inmediatamente a simple vista las velocidades medias en los distintos tramos de un red ferroviaria. Dado que el representar en 2D un gráfico 3D hace que unos rectángulos verticales oculten a veces parcialmente a otros, sería interesante evaluar otras posibles representaciones, como por ejemplo que los lados horizontales superiores de los rectángulos fueran las ”divisorias de aguas”de un mapa 3D. Por otra parte, hay diverso software como el CAS GeoGebra que incorpora importantes capacidades de representación geométrica y de realidad virtual, y resultaría interesante poder ver ”desde dentro” una gráfica de rectángulos como las propuestas, teniendo bajo los pies un mapa usual de la red ferroviaria.

El trabajo analizado en el cuarto capítulo, ”Una simulación en tiempo acelerado de colas de ventanillas de una gran estación de ferrocarril”, está relacionado con un trabajo anterior para terminales aeroportuarias [38], pero al tener el contexto (ferroviario) unas características totalmente distintas, la aproximación y la implementación son completamente distintas. Como posible extensión futura podría considerarse la existencia de ”ventanillas de saturación”, que se abrieran de forma dinámica con el propósito específico de ayudar al tipo de ventanilla que alcanzara una cierta longitud de cola preestablecida.

Capítulo

6

Recursos

- Software matemático de amplia difusión.
- Informes y publicaciones de Adif y Renfe.
- Datos y mapas proporcionados por la Fundación de los Ferrocarriles Españoles, el Consorcio Regional de Transportes de Madrid y Google Maps.
- Otra bibliografía matemática, ferroviaria y sobre software matemático.

El software utilizado ha sido *Maple* y algunos de sus paquetes, destacando el *GraphTheory*.

APPENDIX

A

Ruta mínima en la red de Adif: dos anchos de vía en varias infraestructuras y cambiadores de ancho con *Maple*

En este apéndice se detalla el código del software matemático *Maple* desarrollado para la aplicación de nuestro problema. Siguiendo los pasos explicados en el método, se exponen y explican los procedimientos que cubren y resuelven nuestras necesidades. Los procedimientos no desplegados en el curso de la exposición se añadirán en la última sección del apéndice.

Se empezará cargando los dos paquetes:

```
restart;
with(GraphTheory):
with(plots):
```

A.1 Diseño del grafo en tres capas

Aquí se ajusta el tamaño de la rejilla de las capas y se presentan las tres capas del grafo (aunque finalmente se usarán sólo dos). Cada capa es una plantilla plana de nodos que potencialmente representa a cada subred. Ahora se puede identificar a cada nodo tanto de forma coordenada como por su número de vértice. Las últimas instrucciones nos muestran estas tres capas en la Figura A.1. Es una versión 2D, el comando `SetVertexPositions` del paquete de grafos utilizado no admite una recolocación 3D de los vértices. En el array `stations` se guardaran todos los nombres de los nodos (estaciones). La función `nodo` convierte las coordenadas a número de vértice.

```
rows:=14:# +2(la 0 y la -1)
columns:=18:# +4 (de la -3 a la 0)

nodes:=(rows+2)*(columns+4):
stations:=Array([seq(1 .. 3*nodes)]):
AB1 := Array(-3 .. columns, -1 .. rows, (i, j) -> [i, j] ):
l1 := convert(AB1, list):
AB2 := Array(-3 .. columns, -1 .. rows, (i, j)->AB1[i, j]+[columns+2, 0]):
l2 := convert(AB2, list):
AB3 := Array(-3.. columns, -1 .. rows, (i, j)->AB2[i, j]+[columns+2, 0]):
l3 := convert(AB3, list):
l := [op(l1), op(l2), op(l3)]:
G := Graph(3*nodes):
SetVertexPositions(G, l):

AB := Array(1 .. 1, 1 .. 3):
AB[1, 1] := DrawGraph(InducedSubgraph(G, [seq(1 .. nodes)]), title = "Iberian Gauge nodes"):
AB[1, 2] := DrawGraph(InducedSubgraph(G,[seq( nodes+1..2*nodes)]) ,title = "Standard Gauge nodes"):
AB[1, 3] := DrawGraph(InducedSubgraph(G, [seq(2*nodes+1 .. 3*nodes)]), title = "Dual Gauge nodes"):
display(AB);

nodo:=(x,y)-> (y+1)*(columns+4)+x+4:
```

A.2 Introducción de pesos y aristas

Como se ha explicado se ha utilizado un modelo de grafo no dirigido con peso. Se crea por medio de una matriz de adyacencia (simétrica). Tenemos dos datos por cada sección o arco: la distancia (KM) y su velocidad máxima permitida media (MAX). El tiempo se recogerá calculado en la matriz W.

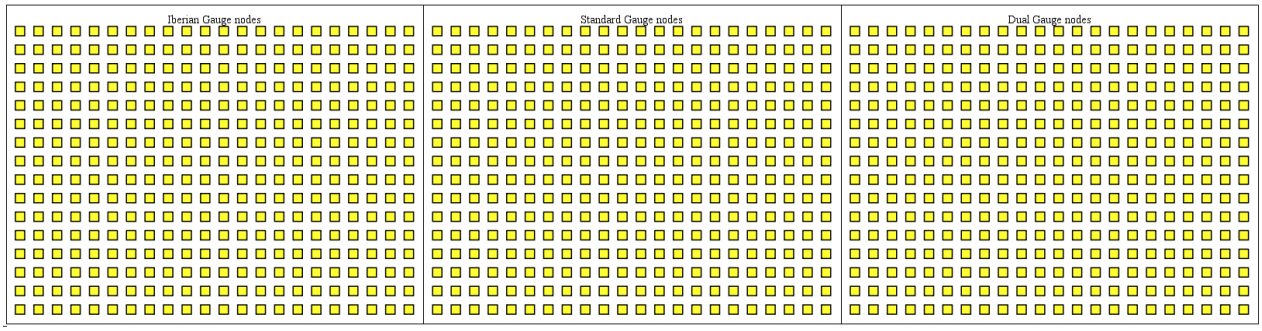

Figura A.1: Plantilla 2D del grafo de tres niveles .

```
KM := Matrix(3*nodes, shape = symmetric):
MAX := Matrix(3*nodes, shape = symmetric):
W := Matrix(3*nodes, shape = symmetric):
```

Se van introduciendo todas las aristas en las matrices anteriores con un básico subprocedimiento llamado `introdata` [1]. En este ejemplo de ancho ibérico se obtiene la arista entre Madrid y Guadalajara, con una distancia de 57Km y una velocidad máxima media en el tramo de 160Km/h, los nodos y los datos de las aristas se introducen junto con las coordenadas:

```
introdata(nodo(6, 8), "Madrid At._i", nodo(7, 9), "Guadalajara_i",  57, 160):
```

Para los otros anchos se actúa igual, por ejemplo ese mismo recorrido en alta velocidad se introduce como se ve a continuación (destacar en el código el nodo trasladado un nivel superior y la diferencia de subíndices en el nombre de la estación):

```
introdata(nodes+nodo(6, 8), "Madrid At._s", nodes+nodo(7, 9), "Guadalajara_s", 64, 300):
```

Una vez introducidos los datos se pueden visualizar por niveles, el procedimiento `GraphTimeLevel`[2] nos muestra cada capa gracias a los comandos del paquete para unir, recortar y dibujar grafos. Aparte de mostrar la capa, el procedimiento también calcula la matriz W a partir de las matrices KM y MAX.

[1] añadido al final del apéndice

[2] Hace uso de otros tres sencillos subprocedimientos: `SiDividir`, `names` y `LimpiarNodos`, que respectivamente calculan tiempos, colocan etiquetas de estación y quitan los nodos de grado cero de la plantilla. Se incluyen al final del apéndice.

```
GraphTimeLevel:=proc(level)
  local G3,r,TXT;
  global Gl,W;
  r:=nodes*level+~[seq(1 .. nodes)]:
  W [r,r]:= round~(SiDividir(KM[r,r], MAX[r,r])*~60):
  Gl := Graph(W [r,r]):
  G3 :=Graph(W [2*nodes+~[seq(1 .. nodes)],2*nodes+~[seq(1 .. nodes)]]):
  AddEdge(Gl, Edges(G3, weights)):
  SetVertexPositions(Gl, l1):
  if Edges(G3) <> {} then HighlightEdges(Gl, Edges(G3), cyan) end if:
  Gl := RelabelVertices(Gl,r):
  LimpiarNodos(Gl): TXT:=names(Gl):
  if Edges(Gl) <> {}
     then display(DrawGraph(Gl, title = "Timings in minutes", showlabels = false),TXT)
  end if:
end proc:
```

El nivel 0 del ejemplo corresponde al ancho ibérico y da la salida de la Figura A.2.

```
GraphTimeLevel(0);
```

El nivel 1 correspondiente al ancho internacional se consigue igual, trasladando los nodos al nuevo nivel (Figura A.3).

```
GraphTimeLevel(1);
```

Aunque, como se ha ido viendo, esta tercera capa correspondiente al ancho mixto es innecesaria para nuestros propósitos (incluso limitativa), no se le puede quitar importancia, al menos comercial. Sólo advertir que la velocidad máxima es la misma para los dos anchos y por lo tanto está bien definida con un sólo peso por arco.

```
GraphTimeLevel(2);
```

A.3 Interconexión

Ha llegado el momento de montar todo el grafo. El procedimiento `gauge_changeovers` lo hace todo: elimina la tercera capa añadiéndola a los dos anchos de vía y conecta ambos niveles con la localización de los cambiadores (el parámetro `location`

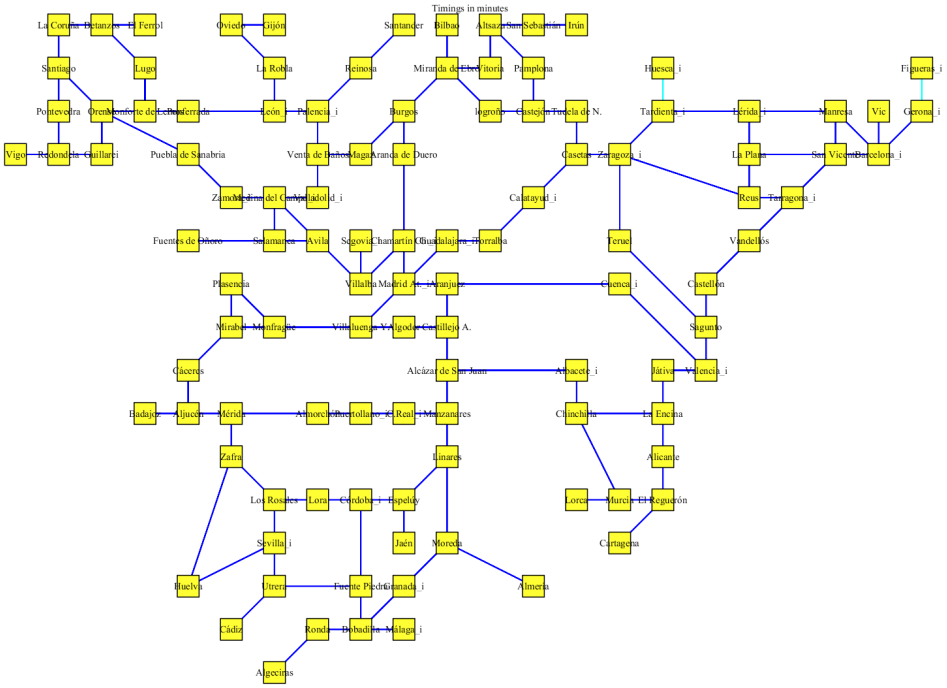

Figura A.2: Subgrafo de ancho ibérico inducido de la primera capa del grafo, se ha añadido en cyan las aristas de la tercera capa que corresponden al ancho mixto

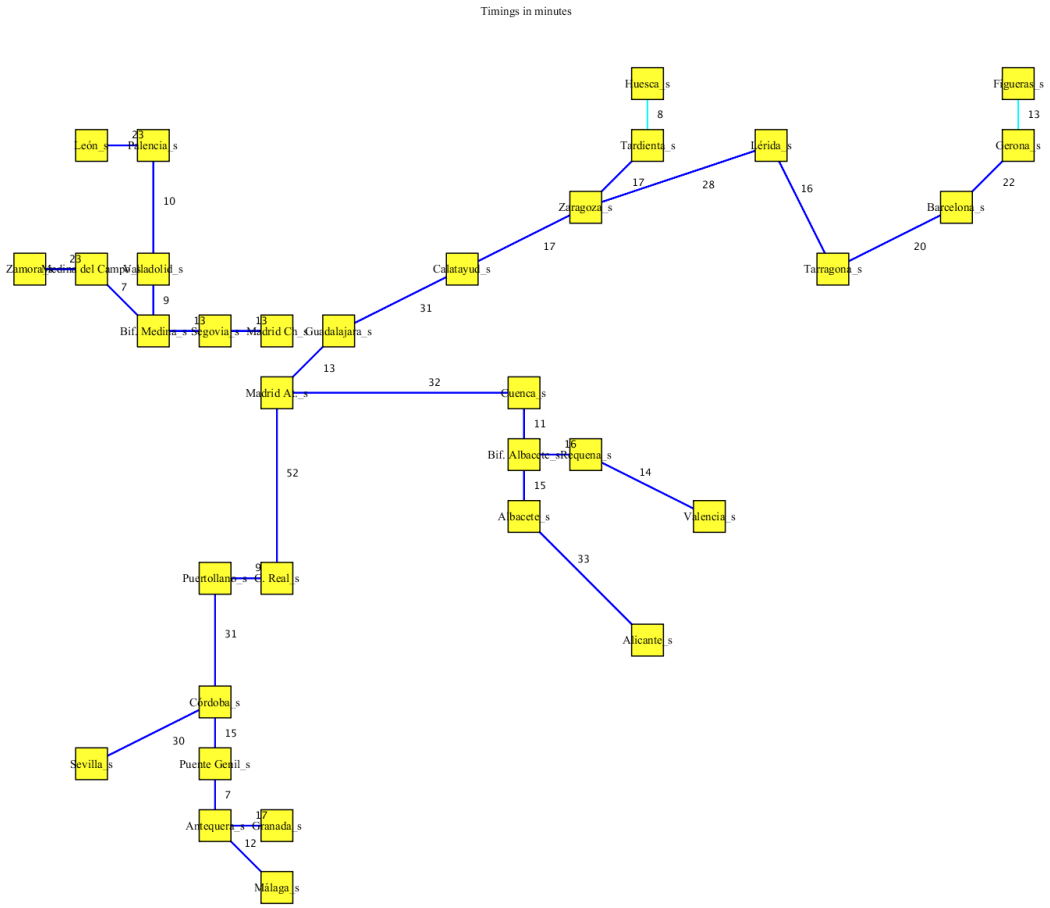

Figura A.3: Subgrafo de ancho internacional inducido de la segunda capa del grafo, se ha añadido en cyan las aristas de la tercera capa que corresponden al ancho mixto

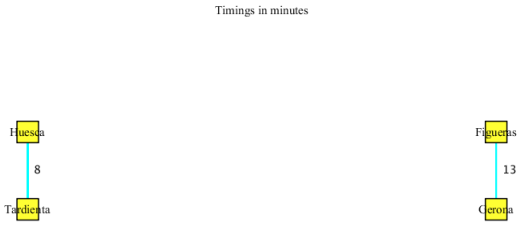

Figura A.4: Subgrafo de ancho mixto inducido de la tercera capa del grafo

es un conjunto de coordenadas), la penalización del intercambiador se fija con el parámetro `delay`. Destacar que la única entrada del procedimiento, aparte de los argumentos indicados, es la matriz W (nuestro grafo de tres capas con tiempos, variable global), que a su vez reducirá su tamaño. Los grafos que representan a toda la red serán las variables globales H y G, en G con los nodos enumerados y en H renombrados con nombres de estación.

```
gauge_changeovers:=proc(delay,location)
   local G3,M,VInter1,VInter2,i,L;
   global H,G,G1,G2,GI,VInter,EInter,W,IT;
   L:=[seq(1 .. nodes)]:
   G3:=Graph(W[ 2*nodes+~L,2*nodes+~L]):
   G2:=Graph(W[ 1*nodes+~L,1*nodes+~L]):
   AddEdge(G2, Edges(G3, weights)):
   G1:=Graph(W[ 0*nodes+~L,0*nodes+~L]):
   AddEdge(G1, Edges(G3, weights)):
   W:=Matrix(2*nodes, shape = symmetric):
   W[1..nodes,1..nodes]:= WeightMatrix(G1):
   W[nodes+1..2*nodes,nodes+1..2*nodes]:= WeightMatrix(G2):
   VInter1 := location:
   VInter2 := VInter1+~nodes:
   VInter := VInter1 union VInter2:
   for i to numelems(VInter1) do W[VInter1[i], VInter2[i]] := delay[i] end do:
   G := Graph(W);  G2:= RelabelVertices(G2,1*nodes+~L):
   GI := InducedSubgraph(G, VInter);
   EInter := Edges(GI) minus Edges(G1) minus Edges(G2);
   SetVertexPositions(G, [op(l1), op(l2)]);
   HighlightVertex(G, VInter, cyan);
   HighlightEdges(G, EInter, cyan);
   LimpiarNodos(G);
   E:=convert(stations,list):
   H:=RelabelVertices(G,E[Vertices(G)]):
   NULL;
end proc:
```

Ajustando la penalización y los cambiadores podemos ver toda la red, lista para aplicar cualquier algoritmo de ruta optima(Figura A.5). Como el número de nodos activos es bastante elevado en este desarrollo y ya que *Maple* limita las etiquetas y los pesos que aparecen en la visualización de los grafos, es más ilustrativa la versión del SW de Adif, y el resultado de estas líneas en aquel caso ya lo hemos visto en la Figura 1.5. Recordar que esta red 2D sólo es una representación de nuestro grafo.

```
location:={nodo(3,2),nodo(5,1),nodo(5,3),nodo(6,8),nodo(10,6),
           nodo(6,9),nodo(3,12),nodo(4,12),nodo(4,10),nodo(3,10),nodo(11, 11),nodo(13,6)}:
delay:=[seq(10,i=1..numelems(location))]:
```

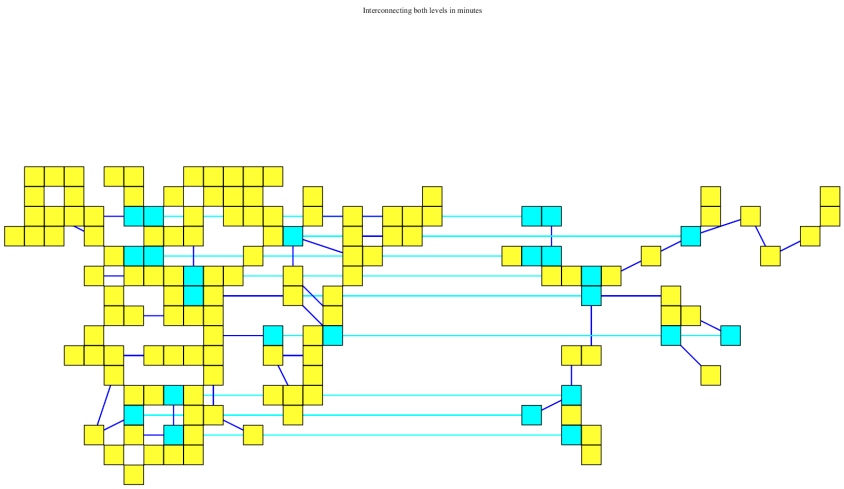

Figura A.5: Toda la red simplificada en España, 2D, a la derecha el ancho internacional y a la izquierda la capa de AV, en cyan los intercambiadores

```
gauge_changeovers(delay,location):
```

A.4 Ruta mínima

En la sección anterior se formó el grafo H de toda la red con la matriz W. Para encontrar la ruta mínima en el software empleado se pueden utilizar los comandos que nos proporciona el paquete, ya sea el `DijkstrasAlgorithm` o bien el `BellmanFordAlgorithm`. Entre las estaciones de Madrid Atocha y Zafra, escribiendo el comando:

```
BellmanFordAlgorithm(H,"Madrid At._i","Zafra");
```

se obtiene la siguiente ruta seguida de los 190 minutos empleados en recorrerla. Estos tiempos no descuentan paradas y son a velocidad máxima de vía. Y con esto se acaba todo el trabajo operativo. Lo curioso es que gran parte del código obedece, como veremos, a las necesidades gráficas.

```
[["Madrid At._i","Villaluenga Y.","Monfragüe","Mirabel","Cáceres","Aljucén",
  "Mérida", "Zafra"], 190]
```

La salida gráfica del comando `DrawGraph` de este paquete sufre de algunas limitaciones para nuestras necesidades. Principalmente debido al orden de los nodos, tamaño y la longitud de los nombres de estación. Se necesita un poco de atención para obtener algo más presentable. El procedimiento `PlottingRoute2d`, corta, pega y enlaza comandos de atributos gráficos de los grafos (que proporciona el propio paquete) con otros comandos gráficos más generales del sofware como `textplot`:

```
PlottingRoute2d := proc (Ruta::list)
   local M, M1, M2, FInter, z, CodigoRuta, TXT1, TXT2, E;
   global H, G, stations;
   CodigoRuta := map(proc (x) options operator, arrow; CodigoEstacion(x, Vertices(H),
                     Vertices(G)) end proc, Ruta[1]);
   E := convert(stations, list);
   M1 := InducedSubgraph(G, `intersect`(convert(CodigoRuta, set), convert(Vertices(G1), set)));
   z := zip(proc (x, y) options operator, arrow; [op(x), y] end proc,
            l1[Vertices(M1)], E[Vertices(M1)]);
   TXT1 := textplot(z, 'axes' = 'none');
   M2 := InducedSubgraph(G, `intersect`(convert(CodigoRuta, set), convert(Vertices(G2), set)));
   M2 := RelabelVertices(M2, `~`[`-`](Vertices(M2), nodes));
   z := zip(proc (x, y) options operator, arrow;[op(x), y] end proc,
            l1[Vertices(M2)], E[Vertices(M2)]);
   TXT2 := textplot(z, 'axes' = 'none');
   M := GraphUnion(M1, M2);
   SetVertexPositions(M, l1[Vertices(M)]);
   HighlightSubgraph(M, M1, blue, cyan);
   HighlightSubgraph(M, M2, red, pink);
   HighlightVertex(M, `intersect`(convert(VInter, set), convert(Vertices(M1), set)), magenta);
   display(DrawGraph(M, showlabels = false), TXT1, TXT2)
end proc
```

El procedimiento anterior nos muestra una versión 2D de ruta óptima en la Figura A.6, indicando con colores el ancho de vía y el paso por los cambiadores .

```
Route:=DijkstrasAlgorithm(H, "La Coruña","Cartagena"):
PlottingRoute2d(Route);
```

A.5 Representación tridimensional

Si en la sección anterior se encontraba algún contratiempo en los comandos gráficos del paquete de grafos para una vista 2D, en 3D los problemas aumentan al no poder

A. RUTA MÍNIMA EN LA RED DE ADIF: DOS ANCHOS DE VÍA EN VARIAS INFRAESTRUCTURAS Y CAMBIADORES DE ANCHO CON *MAPLE*

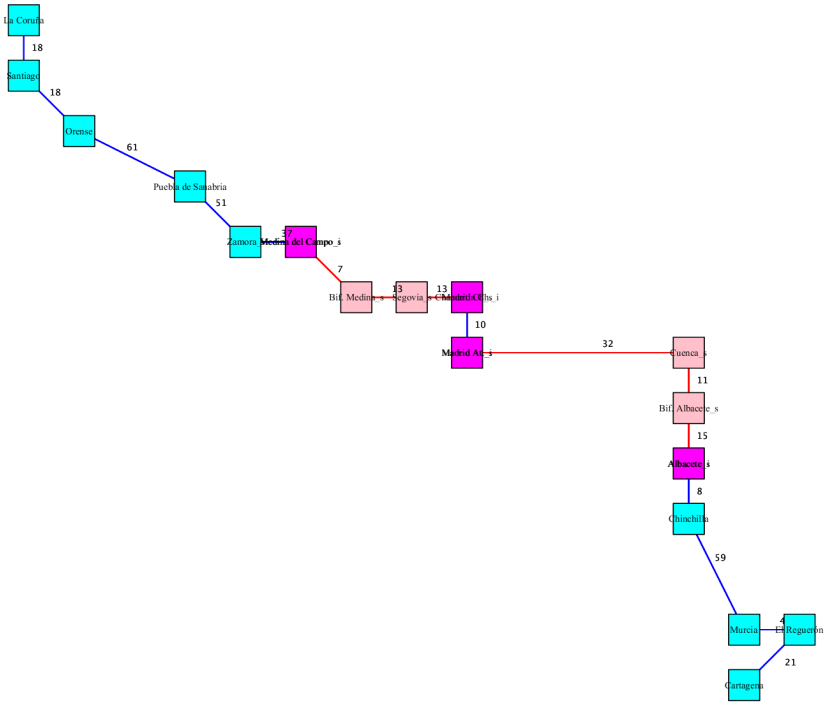

Figura A.6: Ejemplo de salida gráfica para una ruta optima usando comandos del paquete GraphTheory de Maple. En azul ancho ibérico, rojo internacional y cambiadores (que incluyen penalización) en magenta. Tiempo en minutos.

posicionar de forma voluntaria los nodos en el espacio. Esto obligará a trabajar con código a un nivel inferior.

El primer gráfico representará a toda la red de forma tridimensional en dos capas paralelas. El procedimiento se llamará `nice_3D` [1]. El resultado de este código se vio en la Figura 1.7.

```
nice_3D:=proc()
   local A, B, C,E,P0, P1,TXT,NomiNodo,OrdiNodo;
   global arco1, nodos1, arco2, nodos2, arco3, nodos3,Codigo2d,W,G1,G2;
   Codigo2d := [op(l1), op(l2)];   LimpiarNodos(G1): LimpiarNodos(G2):
   arco1 := Edges(G1); nodos1 := Nodos3d(Vertices(G1), Codigo2d, nodes);
   arco2 := Edges(G2); nodos2 := Nodos3d(Vertices(G2), Codigo2d, nodes);
   arco3 := EInter; nodos3 := Nodos3d(VInter, Codigo2d, nodes);
   A := op(arcos3d(arco1, Codigo2d, nodes, blue)),
           pointplot3d(nodos1, color = blue, symbol = point, symbolsize = 10);
   B := op(arcos3d(arco2, Codigo2d, nodes, red)),
           pointplot3d(nodos2, color = red, axes = none, symbol = point, symbolsize = 10);
   C := op(arcos3d(arco3, Codigo2d, nodes, purple)),
           pointplot3d(nodos3, color = purple, axes = none, symbol = box, symbolsize = 10);
   P0 := implicitplot3d(z = 0, x = -4 .. columns+1, y = -2 .. rows+1, z = 0 .. 1,
                        style = surface,color = "Blue", transparency = .8);
   P1 := implicitplot3d(z = 1, x = -4 .. columns+1, y =-2 .. rows+1, z = 0 .. 1,
                       style = surface,color = "Red", transparency = .8,orientation=[-71,57,1] );
   OrdiNodo := Vertices(G);
   E:=convert(stations,list): NomiNodo := E[[op(OrdiNodo)]]:
   TXT := etiketas(OrdiNodo, NomiNodo,black);
   display({A, B, C, P0, P1,TXT});
end proc:
```

`PlottingRoute3d`[2] nos presenta el itinerario tridimensional dentro de toda la red.

```
PlottingRoute3d := proc (RutaL, OtherStations::set)
   local arco0,arco4,arco5,nodos0,nodos4,nodos5,FatPoints,OtherNodes,CodigoRuta,
      A,B,C,D3, E3,F3,P0,P1,TXT,NomiNodo, OrdiNodo,Ruta;
   global H,G;
   Ruta:=RutaL[1];
   CodigoRuta := map(x-> CodigoEstacion(x, Vertices(H), Vertices(G)), Ruta);
   arco0 := Camino2Arco(CodigoRuta);
   arco4 := `intersect`(arco0, arco2);
```

[1] Llamará a otros tres cortos subprocedimientos de carácter técnico: `arcos3d`, `Nodos3d` y `etiketas`, se pueden encontrar al final del apéndice.

[2] LLama a `Camino2Arco`, un nuevo subprocedimiento auxiliar. El argumento `OtherStations` añade los nombres de otras estaciones fuera de la ruta.

```
   arco5 := `intersect`(arco0, arco3);
   nodos0 := Nodos3d(CodigoRuta, Codigo2d, nodes);
   nodos4 := `intersect`({op(nodos0)}, {op(nodos2)});
   nodos5 := `intersect`({op(nodos0)}, {op(nodos3)});
   CodigoRuta := map(x-> CodigoEstacion(x, Vertices(H), Vertices(G)), [op(OtherStations)]);
   OtherNodes:=Nodos3d(CodigoRuta, Codigo2d, nodes);
   FatPoints:=convert(nodos1, set) minus convert(nodos0, set) minus convert(OtherNodes, set);
   A := op(arcos3d(arco1, Codigo2d, nodes, grey)), pointplot3d([op(FatPoints)], color = grey);
   FatPoints:=convert(nodos2, set) minus convert(nodos0, set);
   B := op(arcos3d(arco2, Codigo2d, nodes, grey)), pointplot3d([op(FatPoints)], color = grey);
   C := op(arcos3d(arco3, Codigo2d, nodes, grey));
   D3 := op(arcos3d(arco0, Codigo2d, nodes, "blue")),
            pointplot3d(nodos0, color = "blue", symbol = point, symbolsize = 10);
   E3 := op(arcos3d(arco4, Codigo2d, nodes, "red")),
            pointplot3d(nodos4, color = "red", symbol = point, symbolsize = 10);
   F3 := op(arcos3d(arco5, Codigo2d, nodes, "green")),
            pointplot3d(nodos5, color = "green", symbol = point, symbolsize = 10);
   P0 := implicitplot3d(z = 0, x = -4 .. columns+1, y = -2 .. rows+1, z = 0 .. 1,
                        style = surface,color = "Blue", transparency = .9);
   P1 := implicitplot3d(z = 10, x = -4 .. columns+1, y =-2 .. rows+1, z = 0 .. 11,
                        style = surface,color = "Red", transparency =.9);
   NomiNodo := [op(Ruta), op(OtherStations)];
   OrdiNodo := map(x->CodigoEstacion(x, Vertices(H),Vertices(G)), NomiNodo);
   TXT := etiketas(OrdiNodo, NomiNodo, black);
   display(A, B, C, D3, E3,F3, TXT,P0,P1);
end proc:
```

Así, con estas dos líneas, se consiguen las Figuras A.7 y A.8 , de la que ya se han visto varios ejemplos.

```
Route:=DijkstrasAlgorithm(H,"San Sebastián","Cádiz"):
PlottingRoute3d(Route, {"La Coruña","Madrid At._s","Barcelona_i","Barcelona_s",
                "Valencia_i","Valencia_s"});
```

A.6 Resto de subprocedimientos

Para completar el apéndice, y el script, se añaden los códigos de los subprocedimientos mencionados. Estos tres primeros son de uso general:

```
introdata:=proc(nodo1,station1,nodo2,station2,distance,speed)
   description "Coloca arco y peso en las matrices de los grafos
                de velocidades medias y distancias":
   global KM,MAX,stations:
   stations[nodo1]:=station1:
   stations[nodo2]:=station2:
   KM[nodo1,nodo2]:=distance:
   MAX[nodo1,nodo2]:=speed:
```

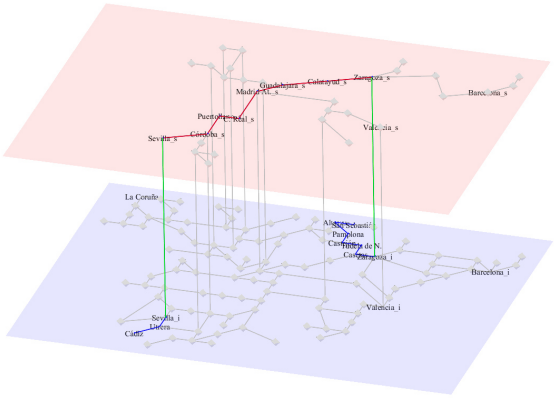

Figura A.7: Ejecución de ruta optima entre San Sebastián y Cádiz. Vista en la que se diferencian los dos niveles, en rojo la AV del ancho internacional y en azul el ancho peninsular. En verde se representa el salto de nivel del cambiador.

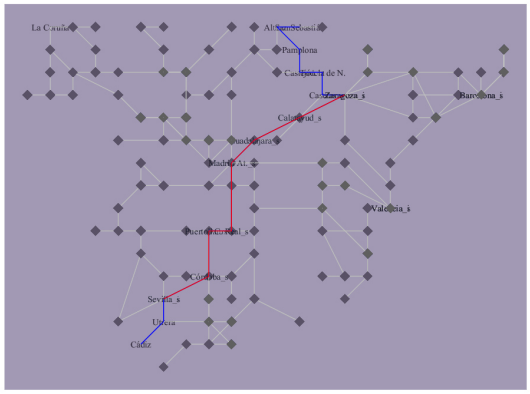

Figura A.8: Ejecución de ruta optima entre San Sebastián y Cádiz. Los distintos colores de los anchos de vía nos permite utilizar la vista cenital.

```
end proc:

SiDividir := proc(M, N)
   description "Opera dos matrices, diviendo elemento a elemento donde exista
                arco (elementos no nulos). Se usa para calcular los tiempos";
   local SiDiv;
   SiDiv := (D, d) -> if d = 0 then 0 else evalf(D/d) end if;
   SiDiv~(M, N):
end proc:

LimpiarNodos := proc(G::Graph)::Graph; # Eliminate the unused nodes
   description " Elimina todos los nodos de grado cero de la plantilla":
   local g, V, v;
   V := Vertices(G);
   g := DegreeSequence(G);
   v := [ListTools:-SearchAll(0, g)];
   G := DeleteVertex(G, V[v])
end proc:
```

El procedimiento siguiente se utiliza para las figuras 2D, creando un plot de etiquetas de estación alternativas. Evita, deshabilitando las etiquetas en el comando *DrawGraph*, que los nombres largos deformen las cajas de los nodos y arruinen el dibujo del grafo.

```
names:=proc(G)
   global l,stations;
   local z,E,TXT;
   E:=convert(stations,list):
   z:=zip((x,y)->[op(x),y],l1[(Vertices(G)-~1) mod nodes +~1],E[Vertices(G)]);
   TXT:=textplot(z,'axes' = 'none');
end proc:
```

La opción gráfica 3D del paquete, sin admitir recolocaciones espaciales de los vértices, mostraba una figura difícil de interpretar. Estos subprocedimientos son solicitados para obtener figuras 3D de fácil comprensión, facilitando o colocando cada arco en su plano.

```
arcos3d := proc(CodigoArco, Codigo2d, Nodos, kolor)
   local Linea, i, f, n;
   description "Auxiliary proc.: returns set graphic commands regarding arcs of nice 3D plot";
   Linea := {}:  n := numelems(CodigoArco):
   f := proc (i::integer)
      if i <= Nodos then [op(Codigo2d[i]), 0] else [op(Codigo2d[i-Nodos]), 1] end if;
   end proc;
   if 0 < n then
     for i to n do
       Linea := Linea union {plottools:-line(f(CodigoArco[i][1]),
```

```
                              f(CodigoArco[i][2]), color = kolor)}
     end do
   else Linea := {}
   end if;
end proc:

Nodos3d := proc(CodigoNodos, Codigo2d, Nodos)
   local f,n;
   description "Auxiliary proc.: raises the nodes to one of the two planes in the 3D plot";
   f := proc (i)
     if i <= Nodos then [op(Codigo2d[i]), 0] else [op(Codigo2d[i-Nodos]), 1] end if;
   end proc;
   map(f, CodigoNodos);
end proc:

etiketas:=proc(OrdiNodo,NomiNodo,kolor)
   local z;
   description "Auxiliary proc.: put the names of the nodes in the space";
   z:=zip((x,y)->[op(x),y],Nodos3d(OrdiNodo, [op(l1), op(l2)], nodes),NomiNodo):
   textplot3d(z,'axes' = 'none',color=kolor)
end proc:

Camino2Arco := proc(camino)
   local i, n, lineas;
   description "Transforms a path (list of nodes) into a set of arcs";
   lineas := {}: n := numelems(camino):
   if 1 < n then for i to n-1 do lineas := `union`(lineas, {{camino[i+1], camino[i]}}) end do
   else lineas := {}
   end if
end proc:
```

APPENDIX

B

Grafo-cronología: mapas esquemáticos retrospectivos de la red de metro mediante grafos con *Maple*

En este apéndice se detalla el código del software matemático *Maple* desarrollado para la aplicación de nuestro problema. Siguiendo los pasos explicados en el método, se exponen y explican los procedimientos que cubren y resuelven nuestras necesidades.

Se empezará cargando los paquetes:

```
restart; with(GraphTheory);
with(plots);
with(ListTools)
```

B.1 Diseño de la red

La red se formaba como unión de líneas, y cada línea como una lista de grafos. Los nodos de la red del suburbano pueden ser situados en una cuadriculación plana

de coordenadas enteras. En nuestro caso de $50 \times 60 = 3000$ posibles nodos. Las estaciones se colocan sobre estos nodos en virtud de su posición geográfica, estableciendo la correspondencia biyectiva entre coordenadas e identificador de vértice (mediante las funciones `coordinates` y `C2N`). Los nombres de estación son tratados en paralelo en un array disperso de caracteres, del mismo tamaño (3000), llamado `StationNames`. Comenzando:

```
Filas := 50;
Columnas := 60;
Nodos := Filas*Columnas;
A := Array(1 .. Filas, 1 .. Columnas, proc (i, j) options operator, arrow; [i, j] end proc);
coordinates := convert(A, list);
C2N := proc (c, f) options operator, arrow; (f-1) . Filas+c end proc;
StationNames := Array(1 .. Nodos, datatype = string);
OldStations := {};
```

Las estaciones renombradas[1] a lo largo de la historia guardan su viejo nombre en un conjunto de listas de la forma `[DateExtinction, vertex, "StationName"]` llamado `OldStations`. El cambio de nombre forma parte de los eventos en la formación de las líneas del metropolitano. Se tendrá en cuenta sólo a la hora de graficar. El contenido es como el que aparece a continuación.

```
{[1931, 1067, "Isabel II"], [1939, 1071, "Progreso"], [1939, 1227, "Principe de Vergara"],
 [1940, 1220, "Gran vía"], [1970, 1124, "Banco"], [1983, 566, "Elvas"], [1983, 1220, "José Antonio"],
 [1983, 1227, "General Mola"], [1986, 772, "Palos de Moguer"], [1994, 1214, "Norte"],
 [1998, 1872, "Lima"], [2007, 2092, "Aeropuerto "], [2017, 1341, "Estadio Olimpico"],
 [2017, 1989, "Campo de las Naciones"]}
```

B.2 Génesis de las líneas

Esta sección es clave ya que la introducción de los datos cronológicos incluye la reordenación de las líneas en grafo-etapas.

La etapa está representada por una lista de la forma [Grafo, FechaInicioEtapa]. La evolución de la línea es tratada como una lista de etapas. Las dos etapas del Ejemplo 2.2 de la línea 6 es procesado en `L[6]` :

```
> L[6];
[[Graph 1: an undirected weighted graph with 27 vertices and 27 edge(s), 1919],
 [Graph 2: an undirected weighted graph with 28 vertices and 28 edge(s), 2007]]
```

[1] Usando el procedimiento `renombrar` que se puede encontrar al final del apéndice.

La L6 entró en servicio en 1979. Aclarar que 1919 es fecha inicial para cualquier primera etapa de la red y genera un grafo vacío para fechas anteriores a la inauguración de las líneas.

Cada línea comienza creando un grafo, cada evento es añadido a este grafo, salvándole como una etapa ante un evento crítico. El primer evento histórico de la red de metro de Madrid, la inauguración de 1919, se codifica así [1] :

```
G := Graph(Nodos, Matrix(Nodos, shape = symmetric)):
StationNames[C2N(17, 35)] := "Cuatro Caminos";
StationNames[C2N(20, 33)] := "Rios Rosas";
StationNames[C2N(20, 31)] := "Iglesia";
StationNames[C2N(20, 30)] := "Chamberí";
StationNames[C2N(20, 29)] := "Bilbao";
StationNames[C2N(20, 27)] := "Tribunal";
StationNames[C2N(20, 25)] := "Gran vía";
StationNames[C2N(20, 22)] := "Sol";
tramo := camino([C2N(17, 35), C2N(20, 33), C2N(20, 31), C2N(20, 30), C2N(20, 29),
                 C2N(20, 27), C2N(20, 25), C2N(20, 22)]);
AddEdge(G, tramo);
for e in tramo do SetEdgeWeight(G, e, 1919) end do
```

Un ejemplo de proceso [2] de un evento crítico se puede ver en el siguiente script de la reordenación de datos. En 1989, se inauguró una nueva estación en la Línea 1, Atocha-Renfe (acceso a la estación de tren de Atocha), sin eventos críticos desde 1967:

```
L[1] := [op(L[1]), [LimpiarNodos(G), 1967]];
DeleteEdge(G, {C2N(23, 20), C2N(25, 17)});
StationNames[C2N(24, 19)] := "Atocha Renfe";
tramo := camino([C2N(23, 20), C2N(24, 19), C2N(25, 17)]);
AddEdge(G, tramo);
for e in tramo do SetEdgeWeight(G, e, 1989) end do
```

B.3 Elección de etapa y poda

En la sección anterior hemos reordenado la cronología de las líneas en etapas. La etapa de una línea correspondiente a un instante dado la encuentra el procedimiento `ChooseStage`, primer paso de la subsección 2.3.3:

[1] El subprocedimiento `camino` transforma una lista de nodos consecutivos de un camino en la lista de sus arcos, se puede encontrar al final del apéndice.

[2] `LimpiarNodos` es un procedimiento que elimina los nodos de grado cero al almacenarlo, da mucha fluidez al programa. Se puede encontrar al final del apéndice.

```
ChooseStage := proc (Linea, fecha)
   local n;
   n := nops(Linea);
   while fecha < Linea[n][2] do n := n-1 end do;
   Linea[n][1]
end proc
```

El procedimiento `prune` poda la etapa, segundo paso visto en la subsección 2.3.3 :

```
prune := proc (RedCompleta, fecha)
  local H, A, a, b, c, s, arcos, peso;
  H := RedCompleta;
  A := [op(Edges(H))];
  peso := proc (G, x) options operator, arrow;GetEdgeWeight(G, x) end proc;
  a := `~`[peso](H, A);
  b := Array(1 .. numelems(A), fill = fecha);
  c := `~`[evalb](`~`[`<=`](a, b));
  s := SearchAll(true, c);
  arcos := {op(A[[s]])};
  H := Subgrafo(RedCompleta, arcos)
end proc
```

Estos dos pasos los recoge el procedimiento `OldNetwork` aplicándolo a varias líneas de metro:

```
OldNetwork := proc (Lines, date)
   local n, G, list;
   list := {};
     for n in Lines do
     G := ChooseStage(L[n], date);
     G := prune(G, date);
     list := {[n, G]} union list
   end do;
   list:
end proc
```

B.4 Salida gráfica

Una vez que se obtiene las líneas en un instante dado, se agregan los nombres a los nodos, y los subgrafos (líneas de metro) se ensamblan y colorean. Todo esto y el dibujo del gráfico se realiza mediante el procedimiento `Graphic`. Los pasos o subprocedimientos seguidos se enumeran a continuación:

1. `OldNetwork`: Explicado arriba.

2. `AssembledGraphs`: enlaza los subgrafos de las diferentes líneas a través de los nexos (correspondencias entre líneas) para obtener un gráfo global de la red, se añade al final.

3. `DistinguishLines`: Colorea subgrafos y nodos, se añade al final.

4. `Stations`: Obtiene los nombres de las estaciones en esa fecha, se añade al final.

5. Dibuja el grafo con el comando `DrawGraph`.

El código es el que sigue:

```
Graphic := proc (lines, date)
   local G, H, z, TXT, position, names, stations;
   global coordinates;
   G := OldNetwork(lines, date);
   H := AssembledGraphs({seq(G[i][2], i = 1 .. nops(G))});
   DistinguishLines(H, G);
   position := coordinates[Vertices(H)];
   stations := Stations(date);
   names := map(x --> stations[x], Vertices(H));
   z := zip((x, y) --> [op(x), y], position, names);
   TXT := textplot(z, 'axes' = 'none');
   SetVertexPositions(H, position);
   display(DrawGraph(H, showlabels = false, showweights = true,
           stylesheet = [edgethickness = 5], transparency = .2,
           title = cat("The Lines ", lines, " in ", date)), TXT)
end proc
```

Como ejemplo, tecleando lo siguiente, se conseguía la Figura 2.10

```
Lineas := {1, 2, 3}:
fecha := 1941:
Graphic(Lineas, fecha);
```

B.5 Resto de subprocedimientos

```
LimpiarNodos := proc (G::Graph)::Graph;
  local H, g, V, v;
  description "elimina nodos de grado cero";
  V := Vertices(G);
  g := DegreeSequence(G);
  v := [ListTools:-SearchAll(0, g)];
  H := DeleteVertex(G, V[v])
end proc
```

B. GRAFO-CRONOLOGÍA: MAPAS ESQUEMÁTICOS RETROSPECTIVOS DE LA RED DE METRO MEDIANTE GRAFOS CON *MAPLE*

```
camino := proc (s)
  local N, n, c;
  description "descompone un camino de nodos en sus arcos";
  N := nops(s); c := {};
  if N = 1 then c := {s[1]}
  else for n to N-1 do c := `union`(c, {{s[n+1], s[n]}}) end do
  end if
end proc

Subgrafo := proc (G::Graph, edges)::Graph;
  description "versión del comando Subgraph adaptada a las necesidades";
  local e, N, M, H;
  N := nops(Vertices(G));
  M := Matrix(N, shape = symmetric);
  H := Graph(Vertices(G), M);
  for e in edges do SetEdgeWeight(H, e, GetEdgeWeight(G, e)) end do;
  H := Graph(Vertices(G), M);
  H := LimpiarNodos(H):
end proc

AssembledGraphs := proc (GG)::Graph;
  description "versión del comando GraphUnion adaptada a las necesidades";
  local e, G, H, nodos;
  nodos := {};
  for G in GG do nodos := `union`(nodos, {op(Vertices(G))}) end do;
  nodos := convert(nodos, list);
  H := Graph(nodos, Matrix(nops(nodos), shape = symmetric));
  for G in GG do for e in Edges(G) do
     AddEdge(H, e);
     if GetEdgeWeight(H, e) = 1 then SetEdgeWeight(H, e, GetEdgeWeight(G, e))
     else SetEdgeWeight(H, e, min(GetEdgeWeight(H, e), GetEdgeWeight(G, e)))
     end if
  end do end do;
  H
end proc

Stations := proc (fecha)
  description "obtiene los nombres coetáneos de las estaciones";
  local estaciones, n;
  global OldStations, StationNames;
  n := numelems(StationNames);
  estaciones := Array(1 .. n, datatype = string);
  estaciones[1 .. n] := StationNames[1 .. n];
  n := nops(OldStations);
  while 1 <= n and fecha < OldStations[n][1] do
    estaciones[OldStations[n][2]] := OldStations[n][3];
    n := n-1
  end do;
  estaciones:
end proc
```

```
Renombrar := proc (vertice::integer, NewStation::string, date)
  description "Actualiza el nombre, almacenando el obsoleto";
  global OldStations, StationNames;
  OldStations := `union`(OldStations, {[date, vertice, StationNames(vertice)]});
  StationNames(vertice) := NewStation
end proc

DistinguishLines := proc (Red::Graph, Lineas)
  description "colorea subgrafos(líneas) y nodos";
  local n, i, k, kolor, nodos, NodosComunes, ArcosComunes;
  n := nops(Lineas);
  nodos := {op(Vertices(Red))};
  NodosComunes := {}; ArcosComunes := {};
  kolor := [wheat, cyan, red, yellow, brown, green, grey, coral, pink, violet, blue, sienna];
  for i to n do
    if Edges(Lineas[i][2]) <> {} then
      HighlightEdges(Red, Edges(Lineas[i][2]), kolor[Lineas[i][1]+1]);
      HighlightVertex(Red, Vertices(Lineas[i][2]), kolor[Lineas[i][1]+1])
    end if;
    for k from i+1 to n do
      NodosComunes := `union`(NodosComunes,
                      `intersect`({op(Vertices(Lineas[i][2]))},{op(Vertices(Lineas[k][2]))}));
      ArcosComunes := `union`(ArcosComunes,
                      `intersect`({op(Edges(Lineas[i][2]))}, {op(Edges(Lineas[k][2]))}))
    end do
  end do;
  HighlightVertex(Red, NodosComunes, "white");
  if ArcosComunes <> {} then HighlightEdges(Red, ArcosComunes, black) end if
end proc;
```

APPENDIX

C

Propuesta de un Gráfico 3D para representar la velocidad de las conexiones por FFCC con *Maple*

En este apéndice se detalla el código en el software matemático *Maple* desarrollado para la aplicación de nuestro problema. Siguiendo los pasos explicados en el método, se exponen y explican los comandos y procedimientos que resuelven sus necesidades. Los subprocedimientos que se mencionan sin código se añaden en la última sección del apéndice.

Se comienza cargando los paquetes necesarios:

```
restart; with(plots): with(ListTools):
```

Como se parte de un mapa simplificado en un sistema coordenado, se ajustan algunos parámetros iniciales. `Unitime` es un conversor de unidades y `CotaI` es la referencia de velocidad máxima alcanzada en toda la red (puede cambiar con la entrada de datos). `IB` y `AV` son las listas que almacenan de forma ordenada las

rutas en ancho ibérico e internacional respectivamente. Las etiquetas se procesan de forma paralela con el conjunto `estaciones`:

```
particion := 10; # tamaño de la partición, escala en km
origen := [40, 45];
abcisa := 60;
ordenada := 45;
estaciones := {};
Unitime := 1/60;
Escala := proc (x) options operator, arrow; [particion*x[1], particion*x[2],
         Unitime*x[3]] end proc # funcion de escala de coordenadas y tiempo
CotaI := 1/(200*Unitime);
roll := 2*rand(1 .. 5)*(1/100):
IB:=[]:AV:=[]:
```

C.1 Introducción de datos

No se utiliza el paquete de teoría de grafos por no poder representar exactamente las gráficas deseadas. Desde un punto de vista teórico partimos de un grafo donde las distintas líneas de trenes forman a través de sus estaciones comunes la red. Con las aclaraciones inferiores ya podremos introducir las líneas:

- Los nodos tienen cuatro componentes, las dos primeras componentes son coordenadas cartesianas centradas en Madrid , la tercera es el nombre y la cuarta es el estado de la etiqueta (1 activada y 0 desactivada).
- Los caminos o rutas de los trenes son introducidos como listas donde alternan nodos con el peso de las aristas que los une.
- La v^{-1} es el peso y está introducido como el cociente entre el tiempo de recorrido y la distancia correspondiente al arco. Cuando no exista parada en nodos intermedios, el peso es común entre sus arcos e igual al $\Delta t/\Delta s$ entre las paradas. Al igual que aparecen nodos sin paradas también hay paradas sin nodo y se actúa de forma similar. La distancia está en km y el tiempo en min.
- El servicio o ruta del tren (`trail`) lo guarda de forma escalonada el procedimiento `Introtrail` en las listas`IB` y `AV`.

Este es un ejemplo de obtención de ruta en ancho ibérico (”ib”), del Talgo Madrid-Almería: 6h 28 min, 4 paradas (Figura C.1) . Nótese en el código que hay una parada sin nodo (Guadix) y un nodo sin parada (Moreda):

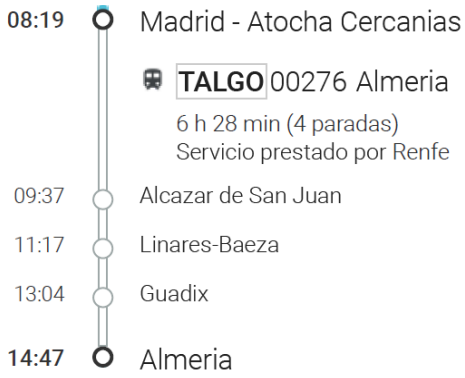

Figura C.1: Ruta ofertada por los servicios de Renfe entre Madrid y Almería. Fuente: Google Maps (12/4/2019)

```
trail := [[0, 0, 'Madrid', 1], 78*(1/147), [6, -10, 'Alcázar*de*San*Juán', 1],
     100*(1/165), [4, -22, 'Linares-Baeza', 1], 210/241, [7, -30, 'Moreda', 0],
     210/241, [15, -36, 'Almería', 1]];
Introtrail(trail, 'ib'):
```

Con ancho estándar se actúa igual, por ejemplo para el AVE Córdoba-Málaga (55min, 2 paradas) se teclearía:

```
  trail := [[-7, -24, 'Córdoba', 1], 31/111, [-7, -31, 'Puente*Genil', 0],
            31/111, [-5, -35, 'Antequera', 1], 24*(1/58), [-3, -38, 'Málaga', 1]];
  Introtrail(trail, 'av'):
```

C.2 El gráfico

El trabajo principal lo realiza el comando `polygonplot3d` del paquete *plots* , pero antes hay que transformar las rutas en polígonos y, de forma paralela, crear otro plot de etiquetas. El procedimiento `Grafico` es el que lo hace, creando un conjunto de dos elementos plots: uno de polígonos y otro de etiquetas. El primer plot se auxilia de `polygons`[1], el segundo, por su parte, de `InMatrix`[2]:

```
Grafico := proc (trails)
  local i, trailEsc, noceroM, L, Listapolygons, estacionesh, D0, D1;
  global estaciones, M, roll, graficos, ordenada, abcisa;
  M := Matrix(ordenada+1, abcisa+1, fill = 0);
```

[1] Subprocedimiento de ordenación de datos para aplicar polygonplot3d

[2] Subprocedimiento de creación de una matriz para facilitar la colocación de etiquetas

```
  # se guarda la matriz auxiliar y se escalan  los puntos
  for i to numelems(trails) do
    InMatrix(M, trails[i], origen);
    trailEsc[i] := map(Escala, trails[i])
  end do;
  # se añaden etiquetas y se crean los plots
  estacionesh := map(proc (x) options operator, arrow; [particion*x[1],
           particion*x[2], M[x[2]+origen[2]+1, x[1]+origen[1]+1]+max(M)*roll(),
           x[3]] end proc, estaciones);
  graficos := {textplot3d(estacionesh)};
  Listapolygons := [];
  for i to numelems(trails) do
    Listapolygons := [op(Listapolygons), op(polygons(trailEsc[i]))]
  end do;
  graficos := `union`(graficos, {polygonplot3d(Listapolygons)})
end proc:
```

El procedimiento `Grafico` ha de tener al menos dos ”trails”, uno puede ser el ”trail” vacío []. Teclear estas dos líneas selecciona todas las rutas de los dos anchos, crea los dos plots y los monta, dando lugar a la vista de la Figura C.2 :

```
Grafico([op(AV), op(IB)]):
display(%);
```

Con el procedimiento `Output` se puede elegir estilo y orientación (la ”A” hace referencia a que el tipo de altura h utilizada ha sido la v^{-1}) , las siguientes líneas son las responsables de la Figura 3.8

```
Grafico([op(AV), op(IB)]):
Output(%, [-90, 65, 19], 'surface', 'A');
```

C.3 Otra opción para h

Como se vio en su capítulo, una alternativa a la $h = v^{-1}$ era $h = v_{max} - v$. Como los pesos se han tomado de la primera forma, pero en la misma red, sólo hay que cambiar los pesos y tener cuidado con las unidades del eje z. El procedimiento `DataDif` transforma este peso.

$$v^{-1} \rightarrow v_{max} - v$$

Las listas de rutas `IB` y `AV` ahora se llaman `IBv` y `AVv` respectivamente por el cambio de la función asignada a h.

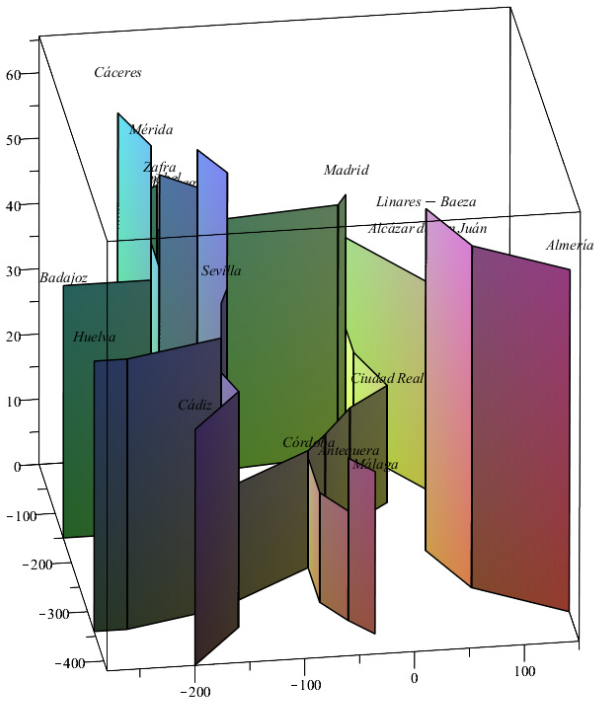

Figura C.2: Gráfico 3D de una red de líneas ferroviarias. El área de cada polígono es el tiempo empleado en recorrer el tramo. Fuente: elaboración propia (realizado en *Maple*)

No cambia nada respecto al caso anterior, sólo la 'B', que ahora se entiende que es para las leyendas y unidades del eje z de esta otra opción. Estas líneas dan la gráfica de la izquierda en la Figura 3.9.

```
DataDif():
Grafico([op(AVv), op(IBv)]):
Output(%, [-90, 90, 0], 'pointline', 'B');
```

C.4 Resto de procedimientos

```
Introtrail := proc (lista, ancho)
  local n, p3, i, nodos, pulgares, WhoStation;
  global IB, AV, CotaI, estaciones;
  description "Almacena  las etiquetas y  escalona los trails de la entrada de datos";
  n := (1/2)*numelems(lista)+1/2;
  nodos := [];
  for i from 2 to n do
    nodos := [op(nodos), [op(lista[2*i-3][1 .. 2]), lista[2*i-2]],
          [op(lista[2*i-1][1 .. 2]), lista[2*i-2]]]
  end do;
  if ancho = 'av' then AV := [op(AV), nodos] else IB := [op(IB), nodos] end if;
  p3 := proc (x) options operator, arrow; x[3] end proc;
  CotaI := min(CotaI, min(map(p3, nodos)));
  pulgares := SearchAll(1, map(proc (x) options operator, arrow; x[4] end proc, lista));
  WhoStation := map(proc (x) options operator, arrow; x[1 .. 3] end proc, lista[[pulgares]]);
  estaciones := `union`(estaciones, {op(WhoStation)})
end proc

polygons := proc (trail)
  local listado, i, up, down, n;
  description "Lista ordenada de los polígonos atómicos de tiempo .";
  listado := [];
  up := proc (x) options operator, arrow; [[x[1], x[2], 0], x] end proc;
  down := proc (x) options operator, arrow; [x, [x[1], x[2], 0]] end proc;
  n := numelems(trail);
  if 0 < n then
    for i to n-1 do
      listado := [op(listado),[op(up(trail[i])),op(down(trail[i+1]))]]
    end do
  else listado := {}
  end if
end proc
```

Este procedimiento usa una matriz como una malla de coordenadas donde introduce el mayor peso en las coincidencias. Así se obtiene la altura de las etiquetas para que queden por encima del perfil del gráfico 3D (evitando que pisen los rectángulos).

```
InMatrix := proc (M, trail, origen)
local x; global Unitime;
description "introduce el trail en la matriz M";
for x in trail do M[x[2]+origen[2]+1, x[1]+origen[1]+1] :=
                  max(M[x[2]+origen[2]+1, x[1]+origen[1]+1], Unitime*x[3])
end do
end proc

Output := proc (DD, orientacion, estilo, tipo)
  description "formatea el gráfico":
  local OptionsPlot, EjeZ, colores;
  if tipo = 'A' then EjeZ := h/Km; colores := z
  else EjeZ := `&Delta;v`; colores := z
  end if;
  OptionsPlot := shading = colores, orientation = orientacion, view = 0 .. 1.1*max(M),
               style = estilo, symbol = sphere, thickness = 1;
  display(DD, labels = [Km*'West-East', Km*'North-South', EjeZ], OptionsPlot)
end proc

DataDif := proc ()
  description"transforma la alternativa A en la B":
  local i, velocidad;
  global AV, IB, AVv, IBv, Unitime, CotaI;
  Unitime := 1/Unitime;
  velocidad := proc (x) options operator, arrow; [x[1], x[2], 1/CotaI-1/x[3]] end proc;
  AVv := AV; IBv := IB;
  for i to numelems(AV) do AVv[i] := map(velocidad, AV[i]) end do;
  for i to numelems(IB) do IBv[i] := map(velocidad, IB[i]) end do
end proc
```

APPENDIX

D

Una simulación en tiempo acelerado de colas de ventanillas de una gran estación de ferrocarril con *Maple*

En este apéndice se detalla el código en el software matemático *Maple* desarrollado para la aplicación de nuestro problema. Siguiendo los pasos explicados en el método, se explican los comandos y procedimientos que resuelven sus necesidades. Los subprocedimientos que se mencionan sin código se incluyen en la última sección del apéndice. La aplicación ha sido probada y depurada.

Como en los temas anteriores se cargan los paquetes:

```
> restart:
> with(Statistics):
> with(ListTools):
> with(plots):
> with(combinat, randperm):
```

D.1 Parámetros, trenes y ventanillas

Se empezará por definir la franja horaria y el paso temporal. Las unidades de tiempo son los minutos. En este ejemplo [1] se tomará la franja horaria desde las 18:45:00 hasta las 20:00:00, con un salto de reloj de medio minuto:

```
> step := .5:
> initial := minute(18, 45, 0):
> duration := 75:
```

Los distintos servicios se introducen como listas de trenes. Cada tren viene dado por una lista de la forma [HoraDePartida, #PasajerosEsperados, IdentificadorTren]. Hay que considerar la posibilidad de aleatorizar el número de pasajeros esperados. El siguiente código define directamente[2] los diferentes servicios, es el utilizado en el ejemplo para simular la llegada de pasajeros:

```
> SuburbanTrain := [[minute([19, 15, 0]), 40, "C0001"], [minute([19, 20, 0]), 40, "C0011"],
                    [minute([19, 25, 0]), 40, "C0002"], [minute([19, 30, 0]), 40, "C0021"],
                    [minute([19, 35, 0]), 40, "C0003"], [minute([19, 40, 0]), 40, "C0031"],
                    [minute([19, 45, 0]), 40, "C0004"], [minute([19, 50, 0]), 40, "C0041"],
                    [minute([19, 55, 0]), 90, "C0005"]]:
> MDTrain := [[minute([19, 30, 0]), 40, "M0001"], [minute([19, 50, 0]), 40, "M0002"]]:
> LDTrain := [[minute([20, 0, 0]), 50, "L0001"]]:
> HSTrain := [[minute([19, 40, 0]), 60, "AVE0001"]]:
```

Los servicios admiten cualquier combinación de subconjuntos de ventanillas. Las de último minuto, que no son un servicio, atienden pasajeros de todas, excepto cercanías.

La distribución de las ventanillas, como se ve en este ejemplo, se introduce por tipo de servicio mediante listas de la forma [conjunto_ventanillas, tiempo_atención]. La asistencia a los trenes de UM se trata de forma similar, con una lista [conjunto_ventanillas, alarma_previa (minutos)].

```
> suburban := [{1,3, 5, 6, 7, 8, 9}, 1.0];
> median := [{10, 11, 12, 13}, 1.5];
> long := [{11, 12, 13, 14}, 2.0];
> high := [{13, 14, 15, 16}, 1.5];
> alarm := 10.0;
> LastMinute := [{17, 18, 19, 20}, alarm];
```

[1] El procedimiento `minute` es un conversor de horas a minutos

[2] A partir de estos parámetros `AllSimulationArrivals` simula todas las cargas de los trenes. No es necesario, ver la sección siguiente.

El conjunto de cambios en las asignaciones a las ventanillas se llama `movements`, estos movimientos dan dinamismo a la configuración de las ventanillas. Cada movimiento es de la forma [instante, servicio_afectado, operador('union'/'minus'), ventanillas_implicadas] entendiendo que en ese instante se modifica dicho servicio en las citadas ventanillas. Un ejemplo es el siguiente donde las ventanillas 2 y 4 comienzan a atender a C a las 19:30 y la número 10 deja de atender a MD a las 19:40. A continuación se introduce un parámetro para las antelaciones de los viajeros [1] y se prepara toda la simulación con el procedimiento `initialize`[2].

```
> movements := {[minute(19, 30, 0), "suburban", `union`, {2,4}],
                [minute(19, 40, 0), "middledistance", `minus`, {10}]}
> randomLenght := [30, 60, 60, 90]:# [suburban,MD,LD,HS] en pasos!!
> initialize():
```

D.2 Curvas de afluencia

Los datos son introducidos tren a tren. A cada tipo de tren le asignamos una curva de afluencia. La curva de afluencia de un tren está recreada por una lista de carga de viajeros de tamaño un número de pasos temporales fijado (lo marca el parámetro `randomLenght`). Cada elemento de esta lista indica el número de pasajeros que llegan a la estación en cada instante previo a la salida del tren. Exactamente, la posición k de una lista de tamaño n para un tren que llega en el instante t_s tiene a los pasajeros que llegaron en el instante $t_{s-n+k-1}$. La suma de los elementos de esta lista es el número de pasajeros que cogen el tren. Se introducen estas longitudes fijas para evitar errores y poderlo aleatorizar.

El siguiente código almacena el tren de MD de las 19:30, con 60 instantes previos.[3]

```
> carga := [0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 2, 3, 1, 2, 2, 3, 0, 2, 0, 3,
            2, 1, 0, 0, 2, 2, 3, 3, 1, 2, 1, 2, 2, 0, 1, 0, 0, 0, 0, 0,
            2, 1, 1, 1, 1, 2, 0, 2, 2, 0, 0, 2, 1, 1, 0, 0, 0, 0, 0, 1]:
> addTrain("Train2378", "MDTrain", minute(19, 30, 0), carga):
```

[1] `randomLenght` se explica al comienzo de la siguiente sección

[2] `initialize` ordena parámetros e inicializa variables, se puede ver al final del apéndice

[3] `addTrain` actualiza la lista de tipos de servicio ya vista y almacena la carga de los trenes en `TrainInptut` de la que se habla un poco más adelante, se incluirá al final del apéndice.

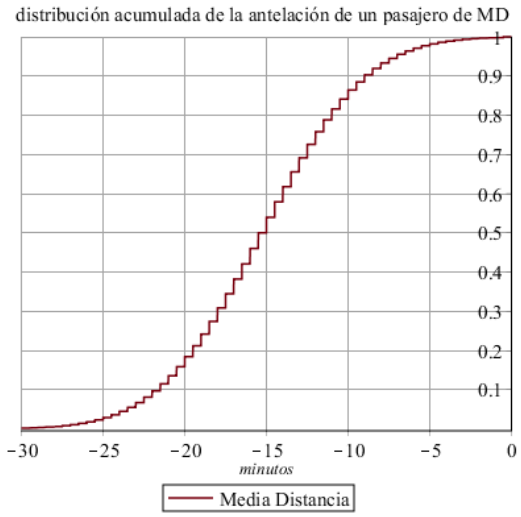

Figura D.1: Función de distribución de la llegada de un pasajero de MD en los instantes previos a la partida del tren, en minutos. El cero corresponde a la salida del tren.

Al no tener disponibles datos experimentales, se construye una distribución para la antelación de un pasajero. Como datos de entrada se han discretizado unas distribuciones ficticias recortando unas normales, por ejemplo en el caso de media distancia:

```
> rango := [seq(0+i*step, i = -randomLenght[2] .. 0)]:
> rangoMD := rango[2 .. -1]: mu := rango[1]/2: sigma := abs(%/2):
> X := RandomVariable(Normal(mu, sigma)):
> c := map(proc (x) options operator, arrow: CDF(X, x) end proc, rango):
> cf := c[2 .. -1]: ci := c[1 .. -2]:
> PMD := cf-ci: PMD := evalf(PMD/add(PMD)):
> XMD := RandomVariable(EmpiricalDistribution(rangoMD, 'probabilities' = PMD)):
```

Se puede ver la distribución discreta de antelación de la llegada de un pasajero de media distancia de la variable aleatoria anterior `XMD` en la Figura D.1.

```
> plot(CDF(XMD, t), t = min(rangoMD) .. 0,
       title = "distribución acumulada de retardo de llegada en minutos",
       legend = "Media Distancia", axis = [gridlines = 10]);
```

Se simula la curva de afluencia de cada tren con una multinomial de la distribución de antelación del pasajero, de tamaño el número de pasajeros de ese tren.

Con esta curva cada pasajero es colocado en el instante en el que llega a las ventanillas. Esto es lo que que hace el procedimiento `cargo`, devolviendo la lista del número de pasajeros que han llegado en los instantes previos a la salida del tren. Los argumentos `X` y `rango` son, respectivamente, la variable aleatoria y el rango explicados en el párrafo anterior.

```
> cargo := proc (X, rango, passangers)
    description "Random generation of delays in the loading of passengers on a train":
    local S, q;
    global horario;
    S := Sample(X, passangers);
    q := map(proc (x) options operator,arrow;numelems([SearchAll(x,S)]) end proc,rango)
  end proc:
```

`AllSimulationArrivals`[1] llamando al procedimiento anterior simula la llegada de todos los pasajeros y de todos los trenes haciendo uso de los parámetros vinculados mencionados. La salida de esta función es la lista `TrainInput` que es una lista de listas [servicio, Tren, Instante]. Por ejemplo en el instante número 80 desde el inicio de la simulación se tendría la siguiente lista de número de pasajeros por tren, agrupados por servicios:

```
> AllSimulationArrivals():
> TrainInput[1 .. -1, 1 .. -1, 80];

       [[0, 0, 0, 0, 0, 0, 1, 2, 0], [0, 2], [2], [0]]
```

El procedimiento `eGrafic`[2] hace sumas parciales de estas llegadas y las grafica. Es el responsable de las curvas que se vieron en 4.4.

```
>  eGrafic(TrainInput);
```

D.3 Fila única

Ahora estas nuevas incorporaciones se tienen que ordenar, en cada instante, en una fila única. Eso lo hace de forma aleatoria el procedimiento `SingleQueue`. Se ayuda del subprocedimiento `shuffle` [3].

[1] `AllSimulationArrivals` se incluye al final del apéndice

[2] El procedimiento `eGrafic` grafica la llegada de pasajeros agrupando por categorías. Se incluye en la última sección del apéndice

[3] `shuffle` es un procedimiento clave para la reordenación en fila única, se incluye al final del apéndice

```
SingleQueue := proc (arrival, t)
  description "the entry of any passenger is ordered in a single chain":
  local c, m, l, partes, reparto, cadena, i, j, a;
  global SuburbanTrain, MDTrain, LDTrain, HSTrain, horario;
  c := numelems(arrival[1]); m := numelems(arrival[2]);
  l := numelems(arrival[3]); a := numelems(arrival[4]);
  partes := map(proc (L) options operator, arrow; op(L) end proc, arrival);
  reparto := shuffle(partes);
  cadena := Array(1 .. add(i, i = partes));
  for i to c do for j in reparto[i] while reparto[i] <> {} do
    cadena[j] := [SuburbanTrain[i][3], t, SuburbanTrain[i][1]]
  end do end do;
  for i from c+1 to c+m do for j in reparto[i] while reparto[i] <> {} do
    cadena[j] := [MDTrain[i-c][3], t, MDTrain[i-c][1]]
  end do end do;
  for i from c+m+1 to c+m+l do for j in reparto[i] while reparto[i] <> {} do
    cadena[j] := [LDTrain[i-c-m][3], t, LDTrain[i-c-m][1]]
  end do end do;
  for i from c+m+l+1 to c+m+l+a do for j in reparto[i] while reparto[i] <> {} do
   cadena[j] := [HSTrain[i-c-m-l][3], t, HSTrain[i-c-m-l][1]]
  end do end do;
  cadena
end proc
```

Su salida ya es la cola de pasajeros, y al ordenarla, también añade unos atributos: el identificador del tren, la hora de llegada del viajero y su hora de partida.[1]

```
> SingleQueue(A[1 .. -1, 1 .. -1, 80], t(80));

[["M0002", 1179.5, 1190.0], ["C0004", 1179.5, 1185.0], ["C0041", 1179.5, 1190.0],
 ["L0001", 1179.5, 1200.0], ["C0041", 1179.5, 1190.0], ["L0001", 1179.5, 1200.0],
 ["M0002", 1179.5, 1190.0]]
```

D.4 Flujo de pasajeros

La propia exposición del método ya es en sí un ejercicio de programación para nuestro problema. Se explicaron las instrucciones y el organigrama de la forma 4.2 para programarlo. En el software utilizado, la codificación del bloque del programa, denominado `run`, se desarrolla a continuación. El parámetro `arrival` del procedimiento será el propio `TrainInput` de la sección anterior y al ejecutarlo se guardan en tres tablas toda la información del proceso durante la franja horaria analizada. Las tablas `q`, `e` y `s` almacenan el estado de las ventanillas, la entrada de pasajeros y la salida de pasajeros, respectivamente:

[1] En este ejemplo aparece el conversor t, $t(80) = t_{80}$ = minuto en el instante 80

```
>  run := proc(arrival)
     description "BLOQUE PRINCIPAL":
     local i:
     global ToExit, windows, q, e, s, HistorialWindows, horario, t, alarm,
            AreSuburban, AreMD, AreLD, AreHS:

     q := table(): e := table(): s := table():
     windows := [seq([], i = 1 .. MaxWindows)]:
     q[t(1)] := windows:

     for i to numelems(horario) do
        movewindows(movements, t(i)):
        e[t(i)] := SingleQueue(arrival[1 .. -1, 1 .. -1, i], t(i)):
        WindowsChoose(%, t(i)): ToExit := []:
        saturation(t(i),SensitiveSaturation,MaxQueue):
        clock(t(i)):
        urgency(t(i + 1), alarm):
        q[t(i + 1)] := windows:
        s[t(i + 1)] := ToExit
     end do
   end proc:
```

Ahora tecleando lo siguiente se genera toda la información (aquí tras una generación automática de `TrainInput` se corre la simulación):

```
>  AllSimulationArrivals(): :
>  run(TrainInput):
```

Sólo queda mostrar en lenguaje Maple las instrucciones pendientes que se vieron en el desarrollo teórico y que aparecen en el bloque anterior, son las instrucciones del bucle que se vio entre cambios de estado:

1. `movewindows`: Modifica las configuraciones de las ventanillas si existe el evento en la lista `movements` ya comentada. Crea la tabla indexada en minutos `HistorialWindows`, y un historial de ventanillas activas para facilitar algunas tareas.

```
movewindows := proc(set, instant)
  description "processes the movements in the windows creating a history":
  local m,i;
  global suburban, middledistance, longdistance, highspeed, lastminute,
         HistorialWindows,windows,step,q;
  HistorialWindows[instant]:=HistorialWindows[instant-step]:
  for m in set do
    if m[1] = t(i) then
      if m[2] = "suburban" then suburban[1]:=m[3](suburban[1],m[4])
```

```
      elif m[2]="middledistance" then middledistance[1]:=m[3](middledistance[1],m[4])
      elif m[2]="longdistance" then longdistance[1] := m[3](longdistance[1], m[4])
      elif m[2]="highspeed" then highspeed[1] := m[3](highspeed[1], m[4])
      elif m[2]="lastminute" then lastminute[1] := m[3](lastminute[1], m[4])
      end if
    end if:
    HistorialWindows[instant]:=[suburban[1],middledistance[1],longdistance[1],
                                highspeed[1],lastminute[1], WindowsSaturation]
  end do
end proc:
```

Podemos ver que a las 19:40 está operativa la ventanilla 3 de C y se ha eliminado la ventanilla 10 para MD:

```
>  HistorialWindows[minute(19,40,0)];
[{1, 2, 3, 4, 5, 6, 7, 8, 9}, {11, 12, 13}, {11, 12, 13, 14, 15},
 {13, 14, 15, 16}, {17, 18, 19, 20}]
```

2. `WindowsChoose`: Es la instrucción de elección de ventanilla, coloca a los usuarios que acaban de llegar en una ventanilla, elige la ventanilla de menor cola que se adapte a las particularidades del viajero. Llama a la subrutina `WhatService`[1], que devuelve al pasajero las ventantanillas disponibles para sus circunstancias, junto al tiempo para despachar. Chequea si un pasajero es de UM.

```
WindowsChoose := proc(passangers, hour)
    description "passengers in the input chain select, sequentially,
                 the shortest queue":
    local m, w, service, placed, i, SizeQueue;
    global windows;
    for i to numelems(passangers) do
      service := WhatService(passangers[i]);
      SizeQueue := numelems~(windows);
      m := min(SizeQueue[[op(service[1])]]);
      w := Search(m, SizeQueue[[op(service[1])]]);
      windows[service[1][w]] := [op(windows[service[1][w]]), [op(passangers[i]),
                                                              service[2]]]
    end do
end proc:
```

3. `clock`: Es la instrucción del reloj, avanza el tiempo, permite cambiar de estado y extrae a los usuarios con su título de transporte ya adquirido.

[1] Se puede encontrar al final del apéndice

```
clock := proc(hour)
  description "the combustion of the machine":
  local w, atentiontime:
  global MaxWindows, windows, ToExit, t;
  for w to MaxWindows do
    atentiontime:=step:
    while windows[w]<> [] and atentiontime>= windows[w][1][4]  do
      atentiontime:=atentiontime-windows[w][1][4]:
      windows[w][1][4] := hour + step-atentiontime:
      ToExit := [op(ToExit), windows[w][1]];
      windows[w] := windows[w][2 .. -1]
    end do:
  if windows[w]<> [] then windows[w][1][4] := windows[w][1][4]-atentiontime fi
  end do
end proc:
```

Veamos los atributos de los pasajeros atendidos a las 19:23:00 (obsérvese que se añade un nuevo atributo, la hora en ser atendido):

```
> s[minute(19, 23, 0)];
[["C0002", 1155.0, 1165.0, 1157.0], ["C0011", 1155.0, 1160.0, 1157.0],
 ["C0011", 1155.5, 1160.0, 1157.0], ["C0011", 1156.0, 1160.0, 1157.0],
 ["C0021", 1156.0, 1170.0, 1157.0], ["M0001", 1154.0, 1170.0, 1157.0]]
```

4. `urgency`: Esta es la instrucción de cambio de cola por parte de usuarios especiales. Cambia a cualquier pasajero, excepto cercanías, que haya entrado en alarma por la salida inminente de su tren, y le compense cambiarse. Los aspirantes siguen un primer criterio por posición en la cola y un segundo criterio por ventanilla más alta, es decir los más cercanos a las ventanillas de último minuto. Usa el subprocedimiento `IfInterchange`[1] para aplicar el primer criterio anterior, busca pasajero en alerta, y sólo se cambia si existen alguna ventanilla con menor tamaño de cola que la posición que esté ocupando.

```
urgency := proc(hour, alarm)
  description "shifts alert passengers to a lower position at the last minute.
               Priority: 1st for position at the head of the queue and 2nd for
               highest window number":
  local w, m, p, wlm, fin;
  global windows, highspeed, middledistance, lastminute;
```

[1] Se puede encontrar al final del apéndice.

```
    fin := false;
    while not fin do
      fin := true;              #excludes commuter windows
      for w in `union`(seq(`union`(seq(HistorialWindows[minuto][i], i in {2,3,4,6})),
                       minuto in horario[1..search(hour-step,horario)])) do
        if 0 < IfInterchange(w, hour, alarm) then
          fin := false;
          m := map(C -> numelems(C), windows[[op(lastminute[1])]]);
          p := IfInterchange(w, hour, alarm);
          wlm := Search(min(m), m);
          windows[lastminute[1][wlm]]:=[op(windows[lastminute[1][wlm]]),windows[w][p]];
          windows[w] := windows[w][[op({seq(1 .. numelems(windows[w]))} minus {p})]]
        fi
      end do
    end do
  end proc:
```

D.5 Gráficos

Se han realizado dos tipos de gráficos, el que reproduce el estado y fase de las colas en un instante y los generales acumulados en toda la franja.

D.5.1 Gráfico de estado

`PlotWindows` muestra el tamaño de las colas para cada ventanilla en un instante o momento del proceso dado. Cada pasajero se representa por una bola de color según el tipo de tren. El eje horizontal representa a la línea de ventanillas y el vertical el tamaño alcanzado por la cola de espera. Se ayuda del subprocedimiento `colorP`[1].

```
PlotWindows := proc(state, minute)
  description "create a ball chart of a state":
  local W, c, Color, w, n, p, i;
  W := {seq(1 .. numelems(state))};
  c := []; Color := [];
  for w in W do
    n := numelems(state[w]);
    if 0 < n then for p to n do
      c := [op(c), [w, p]];
      Color := [op(Color), colorP(state[w][p])]
```

[1]Procedimiento que da color a los pasajeros según el tipo de tren, se puede encontrar al final del apéndice.

```
    end do end if
  end do;
  pointplot(c, view = [0 .. numelems(state), 1 .. 10], color=Color,
            title="windows state"-formath(min2hms(minute), true),
            style=point, symbol=solidcircle, symbolsize=25, scaling=constrained,
            labels=[window*number, queue*size])
end proc:
```

El estado de las ventanilas a las 19:24:30 estará como se ve en la Figura D.2. Se aprecia que en la ventanilla 12 hay pasajeros de dos tipos distintos y que las ventanillas de UM están ocupadas por usuarios AVE. La última instrucción del código nos muestra cómo ve el programa el estado de la ventanilla 12 (como una lista de tres pasajeros):

```
>  instante := minute(19, 24, 30):
>  PlotWindows(q[instante], instante);
>  q[instante][12];

[["L0001",1174.5,1200,1.5],["M0002",1176.5,1190,1.5],["L0001",1177.5,1200,2.0]]
```

D.5.2 Gráficos de evolución

Estos gráficos informan de la mecánica general de las ventanillas con el paso de los minutos. Como una ventanilla puede atender varios tipos de servicio y varias ventanillas ser del mismo tipo, se han considerado los dos siguientes gráficos, se pueden combinar.

- Por tipo de ventanilla: Agrupa a los pasajeros por el tipo de ventanilla en el que estén. Nótese que, al ser compartidas, un pasajero puede estar en varios tipos de ventanilla, a excepción de UM. Por ejemplo, un pasajero en una ventanilla compartida de MD y LD contabilizará tanto en las ventanillas de LD como en las de MD. Presentan o bien el número total de elementos de las colas de ventanilla por hora y tipo de la ventanilla o bien el máximo tamaño en cola (de todas las ventanillas de este tipo) por hora y tipo de ventanilla.

  ```
  >  GeneralGraphicWindowService := proc(ServiceNumber, function)
       description "graph of the evolution of the queues by type of window.
                    function = max / add":
       local totals, points, i;
  ```

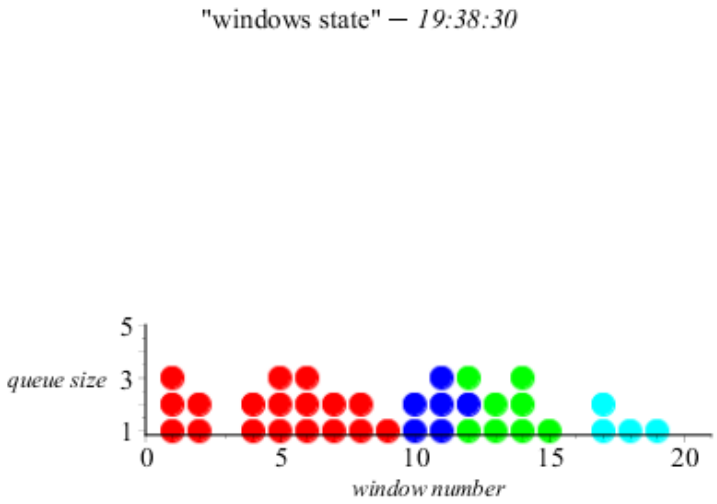

Figura D.2: Gráfico que ilustra el estado de las ventanillas a las 19:24:30. El eje horizontal representa la línea de ventanillas y el vertical el tamaño de la cola. Cada bola representa a un pasajero, en rojo de C, en azul de MD, en verde de LD y en cyan de AVE

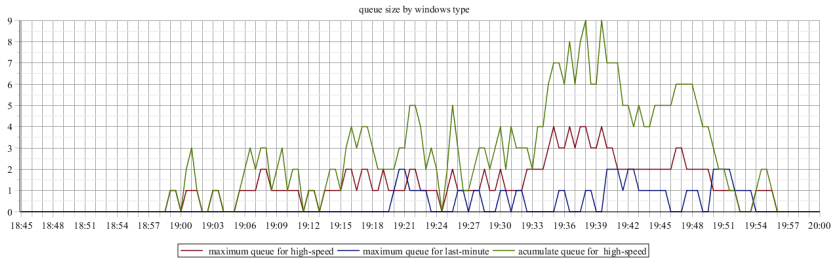

Figura D.3: Gráfico que compara a lo largo de la simulación: la cola total de AVE (en verde), el máximo tamaño de las colas de AVE (en rojo) y el máximo tamaño de las colas de UM (en azul).

```
  global q;
  totals := t -> function([seq(numelems(q[t][w]), w in HistorialWindows[t]
                                                        [ServiceNumber])]);
  points := [seq([t(i), totals(t(i))], i = 1 .. numelems(horario))];
end proc:
```

Para obtener comparativas se pueden unir varios gráficos, como se vio en la Figura 4.10. Las siguientes líneas generan la Figura D.3, donde se comparan la cola total de AVE (tipo 4), el máximo tamaño de las colas de AVE y el máximo tamaño de las colas de UM (tipo 5) :

```
>  plot1 := GeneralGraphicWindowService(4, max):
>  plot2 := GeneralGraphicWindowService(5, max):
>  plot3 := GeneralGraphicWindowService(4, add):
>  plot([plot1, plot2, plot3], gridlines = true,
                                          title = "queue size by windows type",
       tickmarks = [[seq(horario[i] = schedule[i],
                                      i = 1 .. numelems(horario),2)],default],
       legend = ["maximum queue for high-speed","maximum queue for last-minute",
                                           "acumulate queue for  high-speed"]);
```

- Por tipo de tren: Traza el número de elementos de las colas de ventanilla por hora y prestación de los trenes. A diferencia del anterior un pasajero sólo puede estar en un tipo de tren.

```
AccumulatedByTrainType := proc(ServiceNumber)
  description "plot input of the evolution of queues by type of train (service)":
  local f, total, w, c, i, points,TypeTrain;
  global q, t;
```

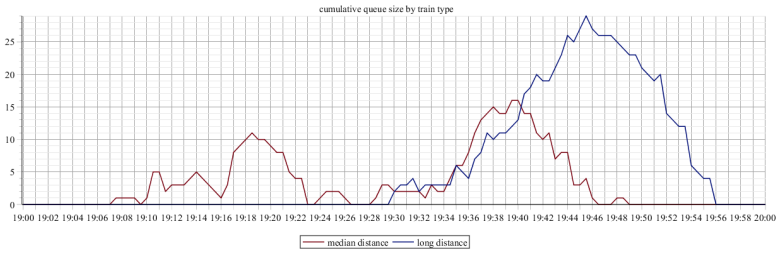

Figura D.4: Gráfico que compara el tamaño de las colas totales de los pasajeros por tipò de tren y hora. En este caso MD y LD.

```
  TypeTrain:=[AreSuburban, AreMD, AreLD, AreHS]:
  f := p -> evalb(p[1] in TypeTrain[ServiceNumber]);
  points := [];
  for i to numelems(horario) do
    total := 0;
    for w in HistorialWindows[t(i)][ServiceNumber] union HistorialWindows[t(i)][5] do
      c := q[t(i)][w];
      map(f, c);
      total := total + numelems([SearchAll(true, %)])
    end do:
    points := [op(points), [t(i), total]]
  end do
end proc:
```

Igualmente se puede obtener comparativas de las colas de viajeros de MD y LD (Figura D.4):

```
> plot1 := AcumulateTrainService(2):
> plot2 := AcumulateTrainService(3):
> plot([plot1, plot2], gridlines = true, title = "cumulative queue size by train type",
       tickmarks = [[seq(horario[i] = schedule[i], i = 1 .. numelems(horario), 2)],
                                  default], legend = ["median distance", "long distance"]);
```

D.6 Resto de los procedimientos

Aquí se incorporan los procedimientos auxiliares.

Este primero es protocolario, define una serie de variables y funciones globales. El procedimiento `simulationset` proporciona todas las variables aleatorias de llegada de pasajeros, como la de media distancia `XMD` vista en 2.2.

```
initialize := proc()
  local totalsteps, i;
  global minute,initial, duration, step, t, horario, schedule, windows, MaxWindows,
         SuburbanTrain, MDTrain, LDTrain,
  HSTrain,TrainInput, HistorialWindows,saturationactived:
  totalsteps := round(duration/step) + 1;
  t := i -> initial*1.0 + step*(i - 1);
  horario := [seq(t(i), i = 1 .. totalsteps)];
  map(h -> formath(min2hms(h), true), horario);
  schedule := map(h -> formath(min2hms(h), false), horario);
  HistorialWindows :=table():
  HistorialWindows[t(0)]:= [suburban[1],middledistance[1],longdistance[1],highspeed[1],
                                                    lastminute[1],WindowsSaturation];
  MaxWindows:=max(%):
  saturationactived:=table():
  saturationactived[t(0)]:=0:
  minute := (h, m, s) -> horario[search(hms2min(h, m, s), horario)]:
  simulationset(step,randomLenght):
end proc:
```

El procedimiento `cargotrain` se utiliza tanto en la introducción manual de datos `addTrain` como en la automática de `ArrivalSimulation`. Traslada la curva de afluencia (de un periodo previo a la salida del tren) a su posición de simulación (horario real).

```
cargotrain:=proc(cargo,hour)
  local i0,train:
  global horario:
  train:= Array(1 .. numelems(horario));
  i0 := search(hour, horario);
  train[[seq(i0+i+1, i = -numelems(cargo) .. -1)]] := Array(cargo);
  convert(train, list)
end proc:
```

Este es otro procedimiento formal, es utilizado en la introducción manual de trenes. Rellena lista de servicios y la carga se almacena en `TrainInput`.

```
addTrain:=proc(trainName,typetrain,minute,charge)
  description "Insert trains":
  global TrainInput,SuburbanTrain, MDTrain, LDTrain, HSTrain, AreSuburban,
                                                    AreMD, AreLD, AreHS;
  if typetrain = "SuburbanTrain" then
    SuburbanTrain:=[op(SuburbanTrain),[minute,add(charge),trainName]]:
    TrainInput[1]:= [op(TrainInput[1]), cargotrain(charge,minute)]:
    AreSuburban := convert(map(x -> x[3], SuburbanTrain), set);
  elif typetrain="MDTrain" then
    MDTrain:=[op(MDTrain),[minute,add(charge),trainName]]:
```

```
    TrainInput[2]:= [op(TrainInput[2]), cargotrain(charge,minute)]:
    AreMD := convert(map(x -> x[3], MDTrain), set);
  elif typetrain="LDTrain" then
    LDTrain:=[op(LDTrain),[minute,add(charge),trainName]]:
    TrainInput[3]:= [op(TrainInput[3]), cargotrain(charge,minute)]:
    AreLD := convert(map(x -> x[3], LDTrain), set);
  elif typetrain="HSTrain" then
    HSTrain:=[op(HSTrain),[minute,add(charge),trainName]]:
    TrainInput[4]:= [op(TrainInput[4]), cargotrain(charge,minute)]:
    AreHS := convert(map(x -> x[3], HSTrain), set);
  else print("nombre de servicio incorrecto")
  fi
end proc:
```

El procedimiento `AllSimulationArrivals` no tiene interés en sí mismo y no se ha incorporado aquí en aras de la brevedad. Hace una llamada por cada tipo de tren al subprocedimiento `ArrivalSimulation`, que sí tiene interés. Éste es el que obtiene la curva de afluencia de un servicio, proporcionando automáticamente los datos de cada tren a `TrainInput`.

```
ArrivalSimulation := proc(service, rank, variable)
  description "Random generation of passenger loading in a train service":
  local i, n, arrivals, types,train,Q;
  global TrainInput;
  types := [SuburbanTrain, MDTrain, LDTrain, HSTrain];
  n := numelems(service);
  arrivals := [];
  for i to n do
    Q:=cargo(variable, rank, service[i][2]):
    train:=cargotrain(Q,service[i][1]):
    arrivals := [op(arrivals), train]
  end do;
  i := Search(service, types);
  TrainInput[i] := arrivals;
end proc:
```

La función `shuffle` es clave para obtener la fila única. Reparte aleatoriamente en r partes de tamaños $n_1, ..., n_r$ un conjunto de $n = \sum_{i=1}^{r} n_i$ elementos. Por ejemplo, para repartir 16 elementos en tres subconjuntos de tamaños 3, 9 y 4, basta teclear:

```
>shuffle([3,9,4]);
 [{4, 11, 13}, {1, 2, 5, 6, 7, 8, 10, 12, 16}, {3, 9, 14, 15}]
```

Esta función admite cardinales nulos (generando subconjuntos vacíos). Si ningún cardinal es nulo, es una partición aleatoria de los tamaños dados. Cada reparto tiene la misma probabilidad D.1 de ser creado (como consecuencia de utilizar el comando `randperm`[1]). El código se muestra a continuación:

$$\frac{\prod_{i=1}^{r}(n_i!)}{(\sum_{i=1}^{r} n_i)!} \tag{D.1}$$

```
shuffle := proc (cardinales)
  local i, n, S, R;
  R := combinat:-randperm(add(cardinales));
  S := [];
  for i to numelems(cardinales) do
    n := cardinales[i];
    S := [op(S), {op(R[1 .. n])}];
    R := R[n+1 .. -1]
  end do;
  S
end proc
```

A continuación se incluye el procedimiento gráfico `eGrafic` que representa las curvas de afluencia instantánea por tipo de servicio. El argumento de entrada tiene que tener la forma de la lista `TrainInput`, para la que está diseñado. Con este procedimiento se realizó, por ejemplo, la Figura 4.4.

```
eGrafic := proc (arrival)
  local PLOTS, colores, puntos, i, m, l, h, q, s;
  global horario, schedule;
  PLOTS := [];
  colores := [red, green, blue, cyan];
  h := numelems(horario);
  for s to numelems(arrival) do
    q := add(arrival[s]);
    puntos := [seq([horario[j], q[j]], j = 1 .. h)];
    PLOTS := [op(PLOTS), puntos]
  end do;
  plot(PLOTS[1 .. numelems(arrival)], color = colores, title = "Curvas de llegada
      de los trenes ", axis = [gridlines], legend = ["Cercanías", "Media Distancia",
      "Larga Distancia", "AVE"], style = pointline, tickmarks = [[seq(horario[i] =
      schedule[i], i = 1 .. numelems(horario), 10)], default])
end proc
```

`WhatService` es un subprocedimiento de `WindowsChoose`, obtiene las ventanillas disponibles para un viajero que se acaba de incorporar, no tiene interés

[1] Comando del paquete combinat de *Maple* que construye una permutación aleatoria.

destacable. Nótese que un recién llegado no puede acceder directamente a cola de UM.

```
WhatService := proc(passanger)
  description "determines the service of a passenger":
  local Wservice, departure;
  global AreSuburban, AreMD, AreLD, AreHS, suburban, middledistance, longdistance,
         highspeed, SuburbanTrain, MDTrain, LDTrain, HSTrain;
  if passanger[1] in AreSuburban then Wservice := suburban
  elif passanger[1] in AreMD then Wservice := middledistance
  elif passanger[1] in AreLD then Wservice := longdistance
  else Wservice := highspeed
  fi:
  Wservice
end proc:
```

La siguiente función es clave para `urgency`. Busca un pasajero en alerta, devolviendo su posición si existen alguna ventanilla con menor tamaño de cola que la posición que esté ocupando.

```
IfInterchange := proc(w, hour, alarm)
  description "Find turncoat passenger. It is only changed if it is on alert and there is
               a window with a queue size smaller than the position it occupies":
  local m, n, p;
  global windows;
  m := map(C -> numelems(C), windows[[op(lastminute[1])]]);
  n := numelems(windows[w]);
  p := 2;
  if 1 < n then
    while p <= n and (alarm < windows[w][p][3] - hour or p <= min(m)) do p := p + 1 end do;
    if n < p or windows[w][p][1] in AreSuburban then 0 else p end if
                                  #evita que se cuelen cercanías de ventanillas compartidas
  else 0
  fi
end proc:
```

Esta última función es muy simple y colorea los pasajeros en los gráficos de estado de `PlotWindows`.

```
colorP := proc (p)
  if p[1] in AreSuburban then red
  elif p[1] in AreMD then blue
  elif p[1] in AreLD then green
  else cyan
  end if
end proc;
```

Bibliografía

[1] Harry Beck. En *Wikipedia*. URL https://wikipedia.org/wiki/Harry_Beck.

[2] Jornadas FuzzyMAD. UCM Facultad de C.C. Matemáticas y el Instiuto Interdisciplinar, Madrid, 2019. URL https://eventos.ucm.es/43474/detail/fuzzymad-2019.html.

[3] Declaración sobre la red. Adif, 2021. URL https://www.adif.es/documents/20124/2269179/20210630_01_DR_Adif_Libro_V0.pdf/0cb3b767-b541-2500-9406-2aa455407187?t=1641320762259.

[4] G. Aguilera Venegas, J. L. Galán García, y E. Mérida Casermeiro. An accelerated-time simulation of baggage traffic in an airport terminal. *Mathematics and Computers in Simulation*, págs. 58–66, 2014. ISSN 0378-4754. URL https://doi.org/10.1016/j.matcom.2013.12.009.

[5] G. Aguilera Venegas, J. L. Galán García, y J. M. García. An accelerated-time simulation of car traffic on a motorway using a CAS. *Mathematics and Computers in Simulation*, págs. 21–30, 2014. URL https://doi.org/10.1016/j.matcom.2012.03.010.

[6] A. Almech y E. Roanes Lozano. A 3D proposal for the visualization of speed in railway networks. *AIMS Mathematics*, 5:7480, 2020. ISSN 2473-6988. URL https://doi.org/10.3934/math.2020479.

[7] A. Almech y E. Roanes Lozano. Automatic generation of diagrammatic subway maps for any date with Maple. *Mathematics in Computer Science*, 14:193–207, 2020. URL https://doi.org/10.1007/s11786-019-00413-8.

[8] A. Almech, E. Roanes Lozano, C. Solano Macías, y A. Hernando. A New Approach to Shortest Route Finding in a Railway Network with Two Track Gauges and Gauge Changeovers. *Mathematical Problems in Engineering*, 2019. URL https://doi.org/10.1155/2019/8146150.

[9] A. Almech y E. Roanes Lozano. An Accelerated-Time Simulation of Queues at Ticket Offices at Railway Stations. *Mathematical Problems in Engineering*, 2021:10, 2021. URL https://doi.org/10.1155/2021/9313174.

[10] R. Aniyeri y R. Nadar. Passengers queue analysis in international airports terminals in Kerala using multiphase queuing system. *International Journal of Mathematics in Operational Research (IJMOR)*, 12(1):1–30, 2018. URL http://doi.org/10.1504/IJMOR.2018.088566.

[11] W. Cartwright. Beck's Representation of London's Underground System: Map or Diagram? En *Geospatial Science Research 2*. RMIT University, Melbourne, 2012. ISBN 978-0-9872527-1-5. URL http://ceur-ws.org/Vol-1328/GSR2_Cartwright.pdf.

[12] W. Cartwright. Going around in circles? A qualitative evaluation of a proposed metro map for Melbourne's underground system. En *Proceedings of the 2014 Geospatial Science Research 3 Symposium*. RMIT University, Melbourne, 2014. ISSN 1613-0073. URL http://ceur-ws.org/Vol-1307/paper2.pdf.

[13] W. Cartwright. Rethinking the definition of the word 'map': an evaluation of Beck's representation of the London Underground through a qualitative expert survey. *International Journal of Digital Earth*, 8(7):522–537, 2015. URL https://doi.org/10.1080/17538947.2014.923942.

[14] E. Castillo, I. Gallego, S. Sánchez Cambronero, J. M. Menéndez, A. Rivas, M. Nogal, y Z. Grande. An Alternate Double–Single Track Proposal for High-Speed Peripheral Railway Lines. *Computer-Aided Civil and Infrastructure Engineering*, 30(3):181–201, 2014. URL https://doi.org/10.1111/mice.12083.

[15] M. Cijsouw y M. A. Westenberg. Embedding Travel Time Cues in Schematic Maps. En *Master's Thesis*. Technische Unversiteit Eindhoven, 2015. URL http://alexandria.tue.nl/extra1/afstversl/wsk-i/Cijsouw_2015.pdf.

[16] J.C. Denain y P. Langlois. Cartographie en anamorphose. *Mappemonde*, 49:16–19, 1998. URL https://doi.org/10.4000/cybergeo.129.

[17] D. Dorling. Anamorphosis: The geography of physicians, and mortality. *International Journal of Epidemiology*, (4):745–750, 2007. ISSN 0300-5771. URL https://doi.org/10.1093/ije/dym017.

[18] F. Dou, K. Yan, Y. Huang, L. Wang, y L. Jia. Optimal path choice in railway passenger travel network based on residual train capacity. *The Scientific World Journal*, 2014:153949, 2014. URL https://doi.org/10.1155/2014/153949.

[19] J. L. Galán García, G. Aguilera Venegas, y P. Rodríguez Cielos. An accelerated-time simulation for traffic flow in a smart city. *Journal of Computational and Applied Mathematics*, 270:557–563, 2014. ISSN 0377-0427. URL https://doi.org/10.1016/j.cam.2013.11.020.

[20] J. L. Galán García, G. Aguilera Venegas, M.A. Galán García, y P. Rodríguez Cielos. A new Probabilistic Extension of Dijkstra's Algorithm to simulate more realistic traffic flow in a smart city. *Applied Mathematics and Computation*, 267:780–789, 2015. URL https://doi.org/10.1016/j.amc.2014.11.076.

[21] J. Gertner. Getting Up to Speed. *New York Times*, Jun 10st 2009. URL http://www.nytimes.com/2009/06/14/magazine/14Train-t.html?pagewanted=all&_r=0.

[22] Gulliver. Will california lead the way on trains? *The Economist*, Jun 21st 2009. URL https://www.economist.com/gulliver/2009/06/20/will-california-lead-the-way-on-trains.

[23] H. Haverkort. Embedding cues about travel time in schematic maps. En *Schematic Mapping Workshop*. 2014. URL http://herman.haverkort.net/lib/exe/fetch.php?media=research:haverkort-cues-about-travel-time.pdf.

[24] A. Hernando, E. Roanes Lozano, A. García Alvarez, L. Mesa, y I. Gonzalez Franco. Optimal route finding and rolling-stock selection for the spanish railways. *Computing in Science and Engineering*, 14(4):82–89, 2012. URL https://doi.org/10.1109/MCSE.2012.80.

[25] A. Hernando, E. Roanes Lozano, y A. García Álvarez. A recommender system for train routing: when concatenating two minimum length paths is not the minimum length path. *Applied Mathematics and Computation*, 319:486–498, 2018. URL https://doi.org/10.1016/j.amc.2017.05.043.

[26] A. Hernando, E. Roanes Lozano, y A. García Álvarez. An accelerated-time microscopic simulation of a dedicated freight double-track railway line. *Mathematical and Computer Modelling*, 51(9-10):1160–1169, 2010. URL https://doi.org/10.1016/j.mcm.2009.12.032.

[27] A. Kierzkowski y T Kisiel. Simulation model of security control system functioning: A case study of the Wroclaw Airport terminal. Journal of Air Transport Management. *Journal of Air Transport Management*, 64(B):173–185, 2017. URL https://doi.org/10.1016/j.jairtraman.2016.09.008.

[28] J. Li. Train station passenger flow study. En *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165)*, tomo 2, págs. 1173–1176. 2000. URL https://doi.org/10.1109/WSC.2000.899082.

[29] J. Li y L. Wang. Microscopic simulation on ticket office of large scale railway passenger station. En *7th Advanced Forum on Transportation of China (AFTC 2011)*, tomo 2, págs. 41 – 47. 2011. URL https://doi.org/10.1049/cp.2011.1374.

[30] J. Lu y Y. Si. Clustering-based force-directed algorithms for 3D graph visualization. *The Journal of Supercomputing*, 76:9654–9715, 2020. URL https://doi.org/10.1007/s11227-020-03226-w.

[31] F. McGee, M. Ghoniem, G. Melançon, y et al. The State of the Art in Multilayer Network Visualization. *Computer Graphics Forum*, 38:125–149, 2019. URL https://doi.org/10.1111/cgf.13610.

[32] S. Pallottino y M. G. Scutellà. Shortest Path Algorithms In Transportation Models: Classical and Innovative Aspects. *Equilibrium and Advanced Transportation Modelling*, págs. 245–281, 1998. URL https://doi.org/10.1007/978-1-4615-5757-9_11.

[33] J. Ponicki, J. Camaj y M. Kendra. Possibilities of Simulation Tools for Describing Queuing Theory and Operations Service Lines in Railway Passenger Transport. En *Proceedings of the 2016 International Conference on Engineering Science and Management*, págs. 191–194. Atlantis Press, 2016/08. URL https://doi.org/10.2991/esm-16.2016.44.

[34] E. Ortega Pérez, S. Mancebo, y I. Otero. Road and railway accessibility atlas of Spain. *Journal of Maps*, 7(1):31–41, 2011. URL http://doi.org/10.4113/jom.2011.1167.

[35] E. Roanes Lozano, A. Almech, C. Solano Macías, y A. Hernando. Determining the Best Routes on Dual Gauge Railway Networks using Graphs. págs. 97–117. Computational and Mathematical Methods in Science and Engineering (MMSE), Cádiz, 2017. ISSN 2312-0177, ISSN-L 2312-0177, ISBN 978-84-617-8694-7.

[36] E. Roanes Lozano, J. L. Galán García, y A. García Álvarez. Estimating radial railway network improvement with a CAS. *Journal of Computational and*

Applied Mathematics, 270:294–307, 2014. URL https://doi.org/10.1016/j.cam.2013.12.051.

[37] E. Roanes Lozano, A. Hernando, A. García Álvarez, L. Mesa, y I. Gonzalez Franco. Calculating the Exploitation Costs of Trains in the Spanish Railways. *Computing in Science and Engineering*, 15(3):89–95, May 2013. URL http://dx.doi.org/10.1109/MCSE.2013.54.

[38] E. Roanes Lozano, L. M. Laita, y E. Roanes Macías. An accelerated-time simulation of departing passengers' flow in airport terminals. *Mathematics and Computers in Simulation*, 67(1-2):163–172, 2004. ISSN 0378-4754. URL https://doi.org/10.1016/j.matcom.2004.05.016.

[39] E. Roanes Lozano, A. García Álvarez, y A. Hernando. A geometric approach to the estimation of radial railway network. *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A. Matematicas (RACSAM)*, 106:35–46, 2012. URL https://doi.org/10.1007/S13398-011-0050-6.

[40] E. Roanes Lozano, A. Martínez Zarzuelo, A. García Álvarez, M. Wester, y E. Roanes Macías. Automatically obtaining railway maps from a set of historical events. *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A. Matematicas (RACSAM)*, 105:149–165, 2011. URL https://doi.org/10.1007/s13398-011-0010-1.

[41] E. Roanes Lozano, A. Martínez Zarzuelo, A. García Álvarez, y E. Roanes Macías. Unas reflexiones sobre el reconocimiento de rutas en mapas ferroviarios y teoría de grafos. *Boletín de la Sociedad Puig Adam de profesores de matemáticas*, 78:79–90, 2008. ISSN 1135-0261. URL https://www.ucm.es/data/cont/media/www/pag-89521//Boletin%2078%20de%20Soc%20PUIG%20ADAM.pdf.

[42] F. Schulz, D. Wagner, y K. Weihe. Dijkstra's algorithm on–line: an empirical case study from public railroad transport. *Algorithm Engineering*, 1668:110–123, 1999. URL http://dx.doi.org/10.1007/3-540-48318-7_11.

[43] H. Small. Florence nightingale's statistical diagrams. En *Stats and Lamps Research Conference organised by the Florence Nightingale Museum*. St. Thomas's Hospital, March 1998. URL https://www.york.ac.uk/depts/maths/histstat/small.htm.

[44] L.H. Soicher. Historical London Underground Maps. URL http://www.geocities.ws/lhsoicher/undergroundmaps.html.

[45] R. Stolletz. Analysis of passenger queues at airport terminals. *Research in Transportation Business and Management*, 1(1):144–149, 2011. URL https://doi.org/10.1016/j.rtbm.2011.06.012.

[46] J. Stott, P. Rodgers, J. C. Martínez-Ovando, y S. G. Walker. Automatic Layout of Metro Maps using Multicriteria Optimisation. *IEEE Transactions on Visualization and Computer Graphics*, 17(1):101–114, 2011. URL http://dx.doi.org/10.1109/TVCG.2010.24.

[47] N. Street. Timecontours: Using isochrone visualisation to describe transport network travel cost. En *Final Report*. Department of Computing, Imperial College London, 2006. URL https://www.researchgate.net/publication/228650015_TimeContours_Using_isochrone_visualisation_to_describe_transport_network_travel_cost.

[48] T. Tang, Y. Shao, y L. Chen. Modeling pedestrian movement at the hall of high-speed railway station during the check-in process. *Physica A: Statistical Mechanics and its Applications*, 467:157–166, 2017. URL https://doi.org/10.1016/j.physa.2016.10.008.

[49] M. Van Campenhout. Travel Time Maps. En *Master's Thesis*. Technische Universiteit Eindhoven, 2018. URL https://research.tue.nl/files/46994894/693219-1.pdf.

[50] C. Ware y P. Mitchell. Visualizing Graphs in Three Dimensions. *ACM Transactions on Applied Perception*, 5:1–15, 2008. URL https://doi.org/10.1145/1279640.1279642.

[51] P. P.-Y. Wu y K. Mengersen. A review of models and model usage scenarios for an airport complex system. *Transportation Research Part A: Policy and Practice*, 47:124–140, 2013. URL https://doi.org/10.1016/j.tra.2012.10.015.