Universidad de Oviedo

Facultad de Ciencias

Grado en Matemáticas

EL PROBLEMA DE DISEÑO DE RUTAS EN EL TRÁNSITO URBANO

Pablo Llorián González

18 de Noviembre de 2025

Dirigido por

Irene Mariñas del Collado y Pelayo Suárez Dosantos

De acuerdo con lo expresado en el artículo 8.3 del Reglamento sobre la asignatura Trabajo Fin de Grado en la Universidad de Oviedo, aprobado por su Consejo de Gobierno el 5 de marzo de 2020 (BOPA de 30 de marzo de 2020), quiero expresar lo siguiente:

Yo, Pablo Llorián González, con DNI 71743725D, en relación con la memoria que presento ante el Tribunal, para su valoración como Trabajo Fin de Grado, quiero DECLARAR que soy el/la autor/a de la misma, habiendo citado debidamente las fuentes utilizadas en su desarrollo.

Para que conste, firmo el presente documento.

En Oviedo, a 7 de noviembre de 2025

Fdo. 

Índice general

Capítulo 1

Introducción

Este capítulo introduce el contexto general del trabajo y las razones que motivan su desarrollo. En él se expone la importancia del problema de diseño de rutas en el transporte urbano y su papel dentro de la planificación moderna de las ciudades. Además, se presentan los objetivos planteados y la estructura general del documento, que sirven de guía para los capítulos posteriores.

1.1. Motivación

El diseño de redes de transporte urbano es un problema de especial interés en la organización de las ciudades modernas, fundamentalmente en un contexto de creciente urbanización, preocupación por el cambio climático y necesidad de sistemas de movilidad más sostenibles. Frente a estas dificultades, la teoría de grafos proporciona un contexto

matemático adecuado para modelar redes de transporte. Las ciudades pueden representarse como grafos cuyos nodos describen paradas o estaciones, y en los que las aristas representan conexiones físicas como pueden ser calles o carreteras.

Este enfoque no solo permite aplicar conceptos matemáticos a un problema con gran impacto práctico, sino que también pone en valor el papel de las matemáticas en la mejora de la calidad de vida urbana. Desde una perspectiva aplicada, resulta de especial interés el estudio de problemas que combinan la teoría de grafos, la optimización y el uso de algoritmos, debido a su utilidad en entornos reales. En este contexto, se plantea el estudio y la implementación de métodos orientados a la optimización de rutas de transporte urbano.

El problema de diseño de la red de transporte urbano (UTNDP, por sus siglas en inglés *Urban Transit Network Design Problem*) busca determinar un conjunto óptimo de rutas y horarios para un sistema de transporte urbano. Se trata de un problema ampliamente reconocido por su complejidad, ya que abarca múltiples etapas interrelacionadas: diseño de red, frecuencia de servicio, elaboración de horarios, asignación de vehículos y programación de conductores (Ceder and Wilson, 1986). El objetivo fundamental del UTNDP es optimizar la eficiencia del transporte público, mejorando aspectos clave como el coste operativo, la cobertura de la red y la calidad del servicio ofrecido. Aunque lograr un incremento relevante del uso del transporte público sea una tarea muy compleja, unos servicios rentables, frecuentes y fiables son atributos esenciales para fomentar un cambio notable en los hábitos de movilidad, reduciendo así la dependencia del vehículo privado.

Lo ideal es optimizar las etapas mencionadas simultáneamente, pero esto no suele ser posible en la práctica dado que cada una de ellas presenta una complejidad computacional muy elevada por sí misma. Según Shih and Mahmassani (1994), entre las distintas fases

que conforman el UTNDP, una de las más estudiadas es el problema de rutas de transporte urbano (URTP, por sus siglas en inglés *Urban Transit Routing Problem*), que se centra en la determinación de las rutas asumiendo como conocidas las localizaciones de las paradas. Trata de optimizar al mismo tiempo dos aspectos clave: el coste para el pasajero y el coste para el operador. Este enfoque multiobjetivo requiere métodos capaces de manejar varias soluciones y explorar eficazmente un gran número de combinaciones posibles.

Tal y como se recoge en Bagloee and Ceder (2011), numerosas redes de transporte público no han sufrido ninguna revisión estructural durante las últimas décadas a pesar de los profundos cambios que se han producido en la configuración de las ciudades. La reubicación de las zonas comerciales de los centros urbanos hacia áreas suburbanas no siempre ha venido acompañada de una respuesta eficaz por parte de los sistemas de transporte público, lo que refuerza la necesidad de nuevos enfoques computacionales para remodelar las redes existentes.

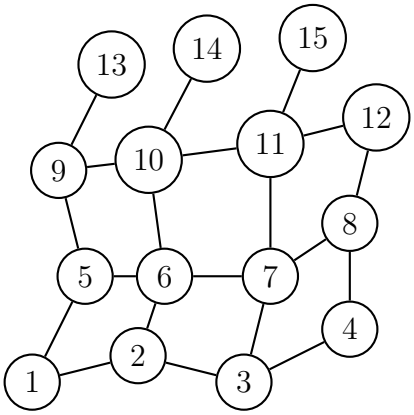


Figura 1.1: Representación de la red de transporte de Mandl (1980).

Como ejemplo representativo, una referencia clásica en el estudio del URTP es el trabajo de Mandl (1980), a partir del cual se define una red de transporte público con 15 nodos (ver figura 1.1). A pesar de su pequeño tamaño, este caso es un modelo recurren-

temente utilizado para evaluar algoritmos, ya que combina una estructura sencilla con suficiente complejidad para poner a prueba métodos heurísticos y evolutivos. Su papel como escenario común en la experimentación permite comparar distintas soluciones bajo condiciones similares.

Los algoritmos evolutivos han ganado un protagonismo creciente en las últimas décadas como herramientas eficaces para la resolución de problemas complejos de optimización. Su fortaleza radica en la exploración de amplios espacios de búsqueda sin requerir información analítica sobre el problema, lo que permite afrontar con éxito entornos complejos o con múltiples restricciones (Pérez and Rückauer, 2004). Gracias a esta versatilidad, se aplican en campos muy diversos, como por ejemplo en la medicina, donde se usan en tareas de planificación y gestión de recursos hospitalarios. En particular, durante la pandemia del COVID-19 se emplearon para optimizar modelos de simulación y mejorar la asignación de recursos en contextos de alta demanda y disponibilidad limitada (Torres Díaz et al., 2022).

En el ámbito académico, la aplicación de algoritmos evolutivos al diseño de rutas de transporte urbano ha evolucionado desde planteamientos generales hacia propuestas cada vez más adaptadas a las características del problema. Inicialmente, estos algoritmos se integraron en esquemas híbridos junto con técnicas como la programación matemática, con el propósito de explorar el espacio de soluciones de una manera más eficiente. A medida que el interés por el enfoque multiobjetivo fue creciendo, se crearon modelos capaces de equilibrar de manera simultánea los costes operativos del sistema y la calidad de servicio al usuario. No obstante, la naturaleza contradictoria de las metas y la complejidad de representar correctamente el conjunto de soluciones eficientes (conocido como frente de Pareto) forzaron la introducción de nuevas estrategias que garantizan tanto la variedad como la viabilidad de las soluciones generadas (Von Lücken et al., 2004).

En este contexto, destacan los algoritmos genéticos, que pueden entenderse como métodos de búsqueda inspirados en la teoría de la evolución biológica de Darwin. Su funcionamiento se basa en una población de soluciones que evoluciona generación tras generación mediante procesos de selección, cruce y mutación, de modo que las nuevas soluciones tienden a mejorar la calidad de las anteriores (Arranz de la Peña and Parra Truyol, 2007). En el ámbito del URTP, además de variantes genéticas clásicas, se han desarrollado algoritmos evolutivos multiobjetivo que incorporan operadores específicos y heurísticas diseñadas para preservar la conectividad de la red y garantizar la validez de las soluciones generadas. Entre estas propuestas, destaca el algoritmo SEAMO2 utilizado en Mumford (2013), que combina dichas estrategias en un marco evolutivo multiobjetivo orientado a optimizar tanto la eficiencia operativa del sistema como la calidad del servicio. En la investigación sobre el diseño de redes de transporte también se han aplicado otros algoritmos, como NSGA-II (Deb et al., 2002), SPEA2 (Zitzler et al., 2001) o SEAMO (Valenzuela, 2002). Sin embargo, SEAMO2 incorpora componentes especializados, como operadores genéticos y un procedimiento de reparación interno, que permiten mantener la factibilidad estructural de las rutas y respetar las restricciones del problema.

El presente trabajo toma como punto de partida el artículo de Mumford (2013), que presenta una versión adaptada del algoritmo evolutivo SEAMO2 para resolver el URTP. Sobre esta base, se busca combinar el estudio teórico del problema con su implementación práctica, analizando el funcionamiento del algoritmo y los resultados obtenidos.

1.2. Objetivos

El objetivo general de este trabajo es abordar el URTP desde un enfoque formal basado en la teoría de grafos y la optimización multiobjetivo. Para ello, se establecen

los fundamentos teóricos necesarios que permiten describir rigurosamente la estructura y funcionamiento de las redes de transporte, sirviendo como punto de partida para el desarrollo del modelo matemático propuesto.

Partiendo de esta idea, se implementa un algoritmo evolutivo adaptado a las particularidades del problema, detallando su estructura, los operadores genéticos empleados y los mecanismos de evaluación y reparación incorporados para garantizar la factibilidad y la calidad de las soluciones generadas. Esta parte del trabajo conecta los conceptos teóricos con el desarrollo del algoritmo, mostrando cómo se aplican en la construcción del método y en el tratamiento de los objetivos que definen el URTP.

Por último, se aplica el algoritmo sobre distintos conjuntos de datos con el objetivo de evaluar su funcionamiento en escenarios de diferente escala y complejidad, verificando la estabilidad de los resultados y la adecuación del enfoque propuesto al problema planteado.

1.3. Estructura del trabajo

El trabajo se organiza en cinco capítulos. En el Capítulo 2 se presentan los fundamentos teóricos necesarios, donde se introducen los principales conceptos de la teoría de grafos y principios básicos sobre algoritmos y optimización multiobjetivo que sirven de base para el desarrollo posterior. De seguido, en el Capítulo 3, se lleva a cabo la formulación del URTP en el marco adoptado, definiendo sus objetivos y restricciones, y se describe el procedimiento evolutivo implementado. En el Capítulo 4 se aplica dicho procedimiento sobre distintos conjuntos de datos y, finalmente, en el Capítulo 5, se recogen las conclusiones y se plantean posibles líneas de mejora y de trabajo futuro.

Capítulo 2

Marco Teórico

Como se ha expuesto en el capítulo previo, el diseño de redes de transporte urbano plantea un problema que presenta una gran dificultad, tanto por la variedad de factores que intervienen como por el elevado número de alternativas que han de evaluarse. En este capítulo se introducen los conceptos teóricos necesarios para el estudio del problema.

2.1. Conceptos básicos de la teoría de grafos

Por su carácter estructurado y relacional, la teoría de grafos resulta especialmente adecuada para modelar redes de transporte urbano. Su formalismo facilita la descripción exacta de la topología del sistema y el análisis de características significativas como la conectividad o la presencia de rutas entre nodos. Esta representación facilitará, en los

próximos capítulos, tanto la formulación matemática del problema como la creación y evaluación de soluciones óptimas.

En esta sección se introducen los elementos fundamentales de la teoría de grafos, comenzando por la definición de grafo y avanzando hacia propiedades y estructuras más específicas que permiten modelar redes de transporte urbano. Se incluyen aspectos como grafos dirigidos, subgrafos o caminos, junto con consideraciones sobre la conectividad y el cálculo de distancias y costes entre nodos.

La primera aparición documentada del concepto de grafo se remonta al año 1736, cuando el matemático suizo Leonhard Euler resolvió el famoso problema de los siete puentes de Königsberg (la actual Kaliningrado, enclave ruso entre Polonia y Lituania). La ciudad, atravesada por el río Pregel, contaba con dos islas unidas entre sí y con las orillas conectadas mediante siete puentes. La cuestión que se planteaba es la siguiente: "¿Se puede atravesar con una ruta continua todos los puentes de modo que se recorran todas las zonas de la ciudad por tierra, pero no se cruce cada puente más que una sola vez?"

Euler propuso una representación abstracta del mapa de la ciudad, en la cual las distintas zonas urbanas se modelaban como nodos y los puentes como aristas que conectaban dichos nodos. Utilizando este enfoque, razonó que para recorrer todos los puentes sin repetir ninguno, cada vez que se entra a una zona (nodo), debe existir un camino de salida distinto (otra arista). Por ello, es necesario que todos los nodos, a excepción del inicial y el final, tengan un número par de conexiones, lo que se denomina grado par. Como bien se puede observar en la figura 2.1, todos los nodos del grafo de la ciudad de Königsberg tienen grado impar; por tanto, tal recorrido era imposible de realizar. Esta sencilla pero profunda reflexión sentó las bases de una nueva rama de las matemáticas (Ortigosa, 2022).

Figura 2.1: Representación esquemática del problema de los siete puentes de Königsberg[1].

A partir de este ejemplo histórico, es posible introducir formalmente los conceptos fundamentales de la teoría de grafos, los cuales resultan imprescindibles para la comprensión del modelado de las redes de transporte urbano. Las definiciones que se presentan a continuación han sido elaboradas a partir de una revisión de distintas fuentes introductorias (ver, por ejemplo, Barrero et al. (2010); Diestel (2025); Gross et al. (2018) y Kocay and Kreher (2016)), unificando la notación y los criterios formales con el fin de mantener la coherencia a lo largo del trabajo.

2.1.1. Estructura general de un grafo

La noción básica a partir de la cual se construyen los modelos de las redes de transporte es la de grafo. Aunque ya se ha mencionado de manera intuitiva, a continuación se presenta su definición formal.

---

[1]Imágen extraída de: https://www.microsiervos.com/archivo/matematicas/siete-puentes-konigsberg-problema-topologico-fotografias.html.

Definición 2.1. *Un* ***grafo*** *es un par* $G = (V, A)$ *donde* V *es un conjunto no vacío de elementos llamados* ***vértices*** *o* ***nodos*** *y* A *es un multiconjunto de pares de elementos de* V*, llamados genéricamente* ***aristas*** *o* ***arcos****.*

Atendiendo a si los pares de elementos que forman A siguen un orden o no, se distinguen dos tipos de grafos.

Definición 2.2. $G = (V, A)$ *se dice* ***grafo orientado*** *o* ***dirigido*** *si* A *es un conjunto de pares ordenados de elementos de* V*, es decir, dados* $e_1 = (i, j)$ *y* $e_2 = (j, i)$*, se tiene que* $e_1 \neq e_2$*. Por consiguiente,* $A \subseteq V \times V$*.*

Los pares $e = (i, j) \in A$ se llaman **arcos**, y se representan gráficamente por $i \to j$, donde i se llama vértice o nodo inicial y j vértice o nodo final de e.

Un ejemplo sencillo se muestra en la figura 2.2, donde se tiene que $V = \{1, 2, 3, 4, 5, 6\}$ y $A = \{(1, 2), (1, 4), (2, 3), (2, 6), (3, 5), (4, 1), (4, 5), (5, 4)\}$.

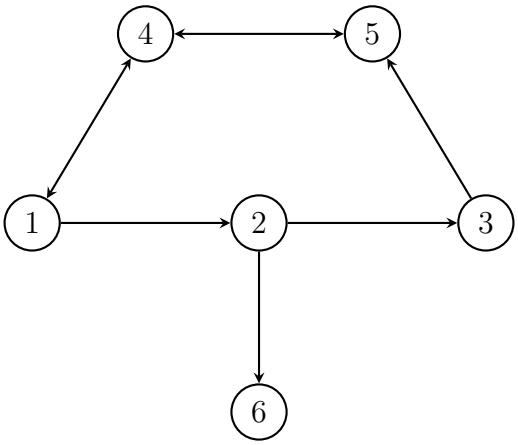


Figura 2.2: Ejemplo de grafo dirigido.

Definición 2.3. $G = (V, A)$ *se dice* ***grafo no orientado*** *o* ***no dirigido*** *si* A *es un conjunto de pares no ordenados de elementos de* V*, es decir,* $\forall i, j \in V$ *se tiene que*

$(i, j) = (j, i)$. *Con lo cual,* $A \subseteq \mathcal{P}_2(V)$, *donde* $\mathcal{P}_2(V) = \{S \in \mathcal{P}(V) : |S| = 2\}$ *denota el conjunto de elementos de* $\mathcal{P}(V)$ *con cardinalidad* 2.

En este caso, los pares no ordenados de A se denotan por $\{i, j\}$, se llaman **aristas** o **ejes** y se representan gráficamente por $i - j$, donde i, j reciben el nombre de vértices o nodos de la arista.

La figura 2.3 ilustra este concepto. En este caso, se tiene que $V = \{1, 2, 3, 4, 5\}$ y $A = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 1\}\}$.

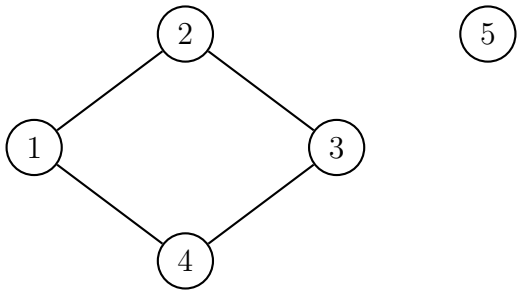


Figura 2.3: Ejemplo de grafo no dirigido.

Aunque en la formulación específica del problema del capítulo posterior se emplearán grafos no dirigidos de acuerdo con el enfoque simplificado de Mumford (2013), en esta sección se ha optado por introducir los conceptos fundamentales desde la teoría de grafos dirigidos. Esta decisión responde tanto a razones de generalidad y coherencia con el desarrollo teórico, como a una motivación práctica, pues en contextos reales de transporte urbano, las rutas no siempre son bidireccionales y resulta necesario representar explícitamente la dirección del movimiento entre pares de nodos. Además, la mayoría de las definiciones que se desarrollan en el marco de grafos dirigidos admiten una versión análoga en el caso no dirigido, la cual permite aplicar directamente los conceptos introducidos en esta sección al modelo particular del URTP.

Además de la estructura de grafo, resulta útil caracterizar su complejidad mediante parámetros numéricos que permitan describir su dimensión global. Dos de los más importantes son el orden y el tamaño del grafo.

Definición 2.4. *Dado un grafo cualquiera $G = (V, A)$, se definen:*

- *El **orden** de G como el número de nodos que contiene, es decir, el cardinal de V. Se denota por $o(G) = |V|$.*
- *El **tamaño** de G como el número de arcos o aristas que contiene, es decir, el cardinal de A. Se denota por $t(G) = |A|$.*

Con el fin de precisar las propiedades estructurales de los grafos que se usan en este trabajo, se introducen a continuación algunas restricciones habituales sobre sus elementos. En particular, se excluyen ciertas configuraciones como la infinitud o la presencia de bucles, aunque se permite la existencia de aristas múltiples, las cuales conducen a la noción de p-grafo.

Definición 2.5. *Si V es un conjunto finito, se dice que G es un **grafo finito**.*

Definición 2.6. *Un **bucle** es una conexión que une un vértice consigo mismo, es decir, un arco de la forma (i, i).*

Este caso puede verse representado en la figura 2.4, donde se ilustra un bucle en el nodo 2 de un grafo dirigido.

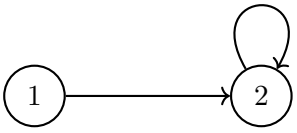
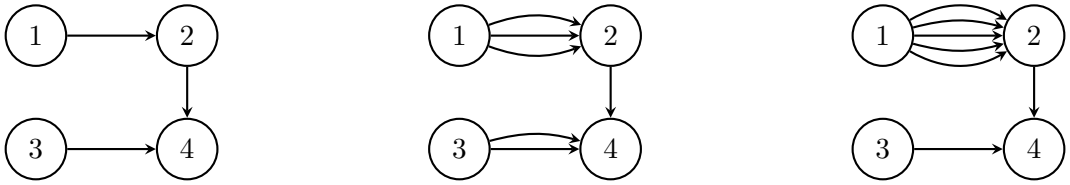


Figura 2.4: Ejemplo de bucle.

Definición 2.7. *Un* p***-grafo*** *es un grafo finito que admite a lo sumo* p *aristas entre cada par de vértices distintos. No admiten bucles.*

Véase un ejemplo en la figura 2.5, donde se muestran diferentes configuraciones de p-grafos dirigidos según el número máximo de arcos permitidos entre cada par de vértices.



1-grafo. 3-grafo. 5-grafo.

Figura 2.5: Ejemplos de p-grafos.

Definición 2.8. *Un* ***grafo simple*** *es un caso particular de un* p*-grafo con* $p = 1$ *y sin bucles.*

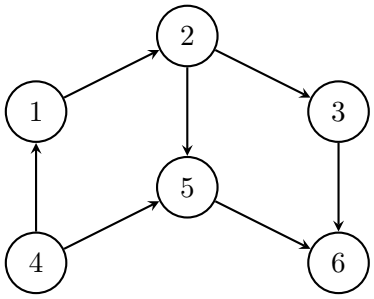
Por tanto, en el URTP aparecen p-grafos no dirigidos, finitos y sin bucles, en los que las aristas representan conexiones entre nodos y pueden existir múltiples enlaces entre un mismo par de vértices.

2.1.2. Subestructuras y relaciones locales

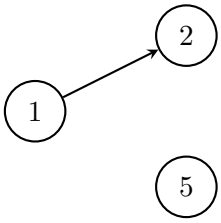
A partir de esta estructura básica, es habitual considerar subconjuntos o restricciones parciales del grafo original, lo que conduce a los conceptos de subgrafo y subgrafo generado. Dado $G = (V, A)$ un grafo orientado cualquiera, se tiene que:

Definición 2.9. *Un* ***subgrafo*** *de* G *es un grafo* $G' = (V', A')$ *con* $V' \subseteq V$ *y* $A' \subseteq A$.

Nótese que, como se exige la condición de que G' sea un grafo, se tiene que $A' \subseteq A \cap (V' \times V')$. Se proporciona un ejemplo de subgrafo en la figura 2.6.



$G = (V, A)$.

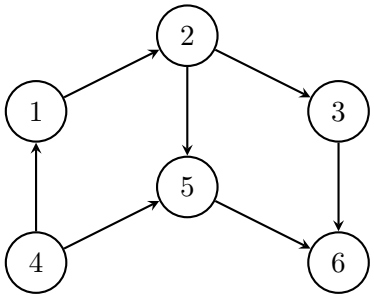


Subgrafo de G.

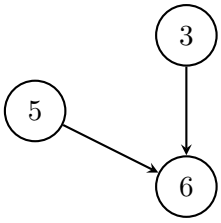
Figura 2.6: Ejemplo de subgrafo.

Definición 2.10. *El* ***subgrafo*** *de* G ***generado por un conjunto de vértices*** $V' \subseteq V$ *es* $G' = (V', A')$*, donde* $A' = A \cap (V' \times V')$.

Se muestra un ejemplo de subgrafo generado por un conjunto de vértices en la figura 2.7.



$G = (V, A)$.



Subgrafo generado por $V' = \{3, 5, 6\}$.

Figura 2.7: Ejemplo de subgrafo generado por un conjunto de vértices.

Además de considerar subestructuras de un grafo, es fundamental conocer las relaciones locales que se establecen entre sus elementos. Estas relaciones permiten caracterizar el

comportamiento de los nodos en términos de conectividad, y son de gran utilidad para entender el patrón de enlaces que une las distintas paradas dentro de una red de transporte urbano. A continuación se introduce el concepto de adyacencia entre vértices y arcos, que tendrá un rol clave tanto en la modelización de la red como en la generación y evaluación de las soluciones obtenidas mediante el algoritmo.

Definición 2.11. *Dos arcos se denominan* ***arcos adyacentes*** *si tienen un vértice en común. Por otro lado, dos* ***vértices*** *se dicen* ***adyacentes*** *si existe un arco que los conecta.*

Véase en la figura 2.8 un ejemplo de ambos conceptos de adyacencia, donde se ilustran las relaciones posibles entre arcos que comparten un mismo extremo y entre vértices conectados.

Arcos adyacentes: $(1, 2)$ y $(2, 3)$ Vértices adyacentes: 4 y 5

Figura 2.8: Ejemplos de adyacencia.

A partir de las nociones anteriores, resulta útil identificar la relación directa entre los extremos de cada arco, lo que permite describir con mayor precisión el sentido de las conexiones en un grafo dirigido. En este contexto, se introduce la siguiente terminología.

Definición 2.12. *Sea $G = (V, A)$ un grafo dirigido cualquiera. Si $(i, j) \in A$, se dice que j es el* ***sucesor*** *de i y que i es el* ***antecesor*** *o* ***predecesor*** *de j.*

2.1.3. Caminos y conectividad

A partir de la estructura básica de un grafo y de las relaciones locales entre sus elementos, resulta esencial comprender cómo estos pueden ser recorridos, es decir, qué nodos pueden alcanzarse desde otros mediante secuencias de arcos consecutivos. Este estudio posibilita la introducción de conceptos como camino, ciclo o conexión, todos ellos esenciales para caracterizar el comportamiento global de una red de transporte.

Definición 2.13. *Sea $G = (V, A)$ un grafo dirigido cualquiera. Un* ***camino*** *o* ***trayectoria*** *es una sucesión ordenada de vértices $\pi_{ij} = (i_1, i_2, \ldots, i_n)$ con $i = i_1, i_2, \ldots, j = i_n$ e $i_k \in V \quad \forall k \in \{1, \ldots, n\}$, de modo que el par $(i_k, i_{k+1}) \in A \quad \forall k \in \{1, \ldots, n-1\}$.*

Definición 2.14. *La* ***longitud*** *de un camino $\pi_{ij} = (i_1, i_2, \ldots, i_n)$ es el número de arcos que lo componen, es decir, $n - 1$.*

En la figura 2.9 se muestra un ejemplo de un camino de longitud 5, compuesto por la secuencia de nodos $(1, 2, 3, 4, 5, 6)$.

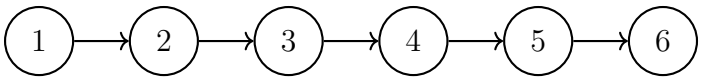


Figura 2.9: Ejemplo de camino.

Definición 2.15. *Un camino se llama* ***camino simple*** *si no pasa dos veces por el mismo arco. Un camino que no pasa dos veces por el mismo vértice se llama* ***camino elemental****.*

Teorema 2.1. *Todo camino elemental es simple.*

Sobre la base de los caminos definidos, cabe considerar aquellos que regresan al punto de partida, generando un recorrido cerrado. Este tipo de estructura recibe el nombre de

ciclo, y es particularmente importante en el marco de este proyecto, ya que la presencia de ciclos internos innecesarios dentro de una ruta puede comprometer su eficacia operativa, al introducir redundancias que elevan el coste del servicio sin mejorar la cobertura o la conectividad.

Definición 2.16. *Un **ciclo** es un camino donde coinciden el vértice inicial y final. Al igual que para los caminos, la **longitud** del ciclo es el número de arcos que lo componen.*

Definición 2.17. *Un **ciclo simple** es aquel que no pasa dos veces por el mismo arco. De manera análoga, un **ciclo elemental** es aquel que no pasa dos veces por el mismo nodo, a excepcón del nodo inicial y final.*

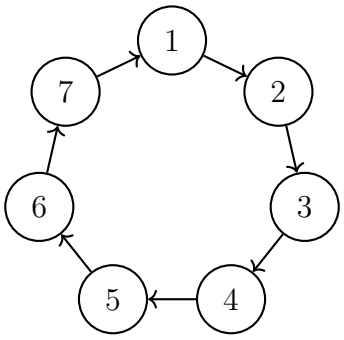


Figura 2.10: Ejemplo de ciclo.

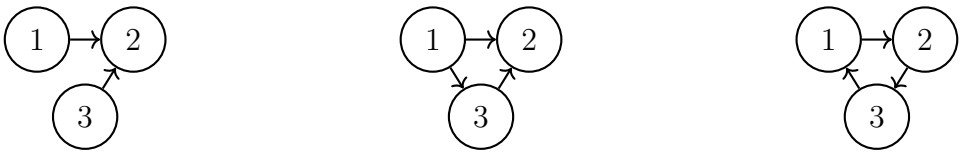
En la figura 2.10 se muestra un ejemplo de ciclo de longitud 7 compuesto por la siguiente secuencia de vértices: $(1, 2, 3, 4, 5, 6, 7, 1)$.

Los conceptos previos permiten abordar el estudio de la conectividad entre vértices, aspecto clave en la regulación de las redes de transporte público. En grafos no dirigidos, un grafo se dice conexo si existe al menos un camino entre cualquier par de nodos. Esta noción es única y no distingue grados de conectividad. En el caso de los grafos orientados, esta conectividad admite distintos grados, cuya distinción es bastante significativa. De seguido se introducen los conceptos de conexión débil, cuasi-fuerte y fuerte.

Definición 2.18. *Dado un grafo orientado cualquiera $G = (V, A)$, se dice que es* ***conexo*** *si existe al menos un vértice $i \in V$ tal que $\forall j \in V$ con $j \neq i$ existe un camino dirigido de i a j. Atendiendo al grado de conectividad que presentan sus vértices, se dice que G es:*

- ***Débilmente conexo*** *si, al ignorar la orientación de los arcos, cualquier par de vértices está conectado por una secuencia de arcos, es decir, existe un camino entre ellos en alguna dirección.*
- ***Cuasi-fuertemente conexo*** *si para todo par de vértices $i, j \in V$, existe al menos un camino dirigido que conecta i con j o bien j con i (no necesariamente ambos).*
- ***Fuertemente conexo*** *si para todo par de vértices $i, j \in V$, existe un camino dirigido de i a j y también de j a i.*

La figura 2.11 ilustra estos tres conceptos de conectividad que se pueden dar en grafos dirigidos.



Débilmente conexo. Cuasi-fuertemente conexo. Fuertemente conexo.

Figura 2.11: Ejemplos de tipos de conectividad en grafos dirigidos.

2.1.4. Pesos, caminos de menor valor, representación matricial y algoritmo de Floyd-Warshall

Hasta ahora se han introducido las nociones fundamentales que permiten representar topológicamente una red de transporte. Sin embargo, en muchos contextos prácticos no es suficiente con conocer la estructura de conexión entre los nodos y es necesario incorporar información cuantitativa sobre el coste de los desplazamientos entre pares de vértices. Este tipo de información se incorpora mediante la asignación de un valor llamado peso a cada arco del grafo, que representa magnitudes relevantes como pueden ser el tiempo de recorrido, la distancia o el coste económico. El concepto formal que captura esta idea es el de grafo ponderado, que se presenta a continuación.

Definición 2.19. *Un **grafo ponderado** es una terna $G = (V, A, p)$, donde:*

- *V es un conjunto finito de nodos o vértices.*
- *$A \subseteq V \times V$ es un conjunto de arcos dirigidos.*
- *$p : A \to \mathbb{R}^+$ es una función de peso que asigna a cada arco un valor numérico positivo, el cual suele representar una magnitud como tiempo, coste o distancia.*

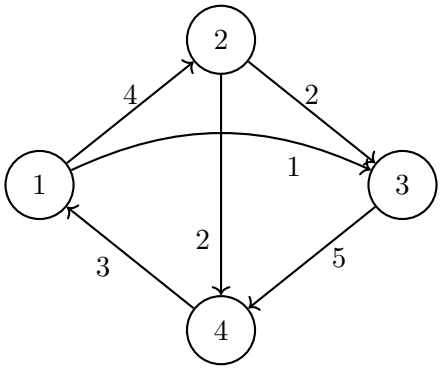


Figura 2.12: Ejemplo de grafo ponderado dirigido.

Este caso particular puede verse representado en la figura 2.12, que ilustra la asignación de pesos a los arcos de un grafo dirigido.

Es importante destacar que los pesos asignados a los arcos en un grafo dirigido dependen de la orientación, por lo que el coste de ida y vuelta entre dos vértices puede diferir. Aunque no se dé en el URTP, esta asimetría es común en redes de transporte urbano, donde pueden llegar a influir factores como el tráfico, la pendiente o las paradas.

La presencia de pesos en los arcos permite comparar distintas rutas entre vértices. De ahí surge el concepto de camino de menor valor, entendido como aquel que minimiza el coste total del recorrido.

Definición 2.20. *Sea $G = (V, A, p)$ un grafo ponderado dirigido. Dado un par de vértices $i, j \in V$, se denota por Π_{ij} al conjunto de todos los caminos que van desde i hasta j.*

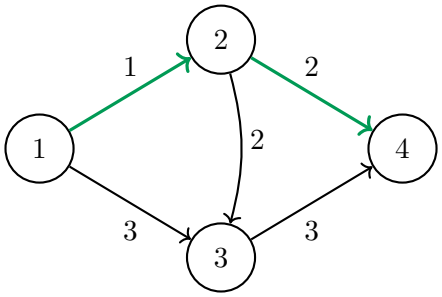
*Para un camino $\pi_{ij} \in \Pi_{ij}$, definido por la secuencia de vértices $(i_1, i_2, \ldots, i_n)$ con $i = i_1, i_2, \ldots, j = i_n$, el **valor del camino**, denotado por $w(\pi_{ij})$, es la suma de los pesos de los arcos que lo componen:*

$$w(\pi_{ij}) = \sum_{k=1}^{n-1} p(i_k, i_{k+1}).$$

*Se dice que $\pi_{ij}^* \in \Pi_{ij}$ es un **camino de menor valor (CMV)** de i a j si minimiza el valor total del recorrido:*

$$w(\pi_{ij}^*) = \min_{\pi_{ij} \in \Pi_{ij}} w(\pi_{ij}).$$

Se facilita un ejemplo gráfico en la figura 2.13, donde se muestran los tres caminos posibles entre los vértices 1 y 4 y se identifica el CMV.



Camino	Valor total
$\pi_{14} = (1, 2, 4)$	$1 + 2 = 3$
$\pi'_{14} = (1, 3, 4)$	$3 + 3 = 6$
$\pi''_{14} = (1, 2, 3, 4)$	$1 + 2 + 3 = 6$

Figura 2.13: Ejemplo: $\pi_{14} = (1, 2, 4)$ es el camino de menor valor del nodo 1 al 4.

Para abordar de forma operativa la información estructural de un red, resulta habitual recurrir a la representación en forma de matriz del grafo. Este enfoque permite expresar tanto la presencia como la ausencia de arcos entre nodos, así como el valor de sus pesos asociados si se dispone de una estructura ponderada.

Definición 2.21. *Sea $G = (V, A)$ un grafo dirigido de orden n. La **matriz de adyacencia** de G, denotada por $A(G) = (a_{ij})_{i,j=1}^{n}$, es la matriz de orden $n \times n$ definida por:*

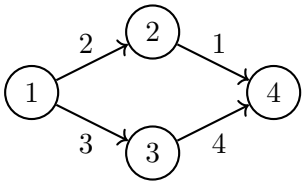
$$a_{ij} = \begin{cases} 1 & si\ (i, j) \in A, \\ 0 & si\ (i, j) \notin A. \end{cases}$$

Esta matriz refleja únicamente la existencia o ausencia de arcos entre los vértices i y j, indicando con un valor binario si ambos se encuentran directamente conectados en el grafo.

Definición 2.22. *Sea $G = (V, A, p)$ un grafo ponderado dirigido de orden n. La* ***matriz de pesos o costes*** *de G, denotada por $W(G) = (w_{ij})_{i,j=1}^{n}$, es la matriz de orden $n \times n$ definida por:*

$$w_{ij} = \begin{cases} 0 & si\ i = j \\ p(i,j) & si\ (i,j) \in A, \\ \infty & si\ (i,j) \notin A. \end{cases}$$

La figura 2.14 muestra un ejemplo de una matriz de adyacencia, acompañada de su matriz de pesos, que permite visualizar simultáneamente la estructura del grafo y los costes asociados a cada arco.



$$A(G) = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad W(G) = \begin{pmatrix} 0 & 2 & 3 & \infty \\ \infty & 0 & \infty & 1 \\ \infty & \infty & 0 & 4 \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

Figura 2.14: Ejemplo de matriz de adyacencia y de costes asociadas a un grafo ponderado dirigido.

En el caso de los grafos no dirigidos, la representación matricial se construye de la misma manera. Sin embargo, en este tipo de grafos tanto la matriz de adyacencia como la de pesos presentan la particularidad de ser simétricas, ya que las conexiones son bidireccionales y los valores asociados a cada par de vértices son idénticos en ambos sentidos.

En el ámbito de los grafos ponderados, el problema del camino de menor valor puede plantearse de diversas maneras: o bien entre un par de vértices concretos o bien de manera global, considerando todos los pares de vértices del grafo. Para el primer caso, existen

algoritmos como el de Dijkstra (1959) o el de Bellman-Ford, que fue descubierto de manera independiente por Ford Jr (1956) y Bellman (1958), eficientes cuando se realizan consultas puntuales. No obstante, esta perspectiva resulta poco adecuada cuando el análisis requiere considerar simultáneamente múltiples trayectos dentro de una red de transporte.

Este es precisamente el escenario que se plantea en este trabajo, donde resulta necesario evaluar soluciones de forma global en función del tiempo mínimo de trayecto entre zonas de la red de tránsito. Tanto el coste asociado al operador como el coste correspondiente con el pasajero dependen directamente de los tiempos de trayecto entre los distintos pares de nodos de un grafo. Para poder evaluar adecuadamente cualquier solución candidata, es necesario conocer de antemano los tiempos mínimos entre todos los vértices del grafo.

El algoritmo de Floyd-Warshall, formulado de manera independiente por Floyd (1962) y Warshall (1962), permite construir esa matriz de tiempos de forma eficaz a partir de la representación ponderada del grafo, y constituye por tanto una herramienta fundamental en el proceso de evaluación integrado en el modelo de optimización multiobjetivo que se implementa en este trabajo.

A la vista de las necesidades descritas, es fundamental disponer de una herramienta que, a partir de un grafo ponderado dirigido $G = (V, A, p)$, permita calcular de manera eficiente los tiempos mínimos de trayecto entre todos los pares de vértices. Dicha información se organiza en una matriz de distancias mínimas (que en general puede representar no solo distancias, sino también otros tipos de costes, como los tiempos de recorrido considerados en el URTP), cuya construcción constituye un paso clave dentro del proceso de evaluación de soluciones del modelo. El algoritmo de Floyd-Warshall ofrece una solución sistemática a este problema, actualizando iterativamente las estimaciones de menor coste entre nodos mediante una estrategia basada en programación dinámica. Aunque en

su formulación general el algoritmo admite la presencia de pesos negativos siempre que no existan ciclos de coste negativo, en el contexto de este trabajo todos los pesos son positivos, pues representan tiempos de recorrido en una red de transporte.

Teorema 2.2 (**Algoritmo de Floyd-Warshall**). *Sea $G = (V, A, p)$ un grafo ponderado, dirigido, fuertemente conexo, de orden n y con pesos positivos, es decir, $p : A \to \mathbb{R}^+$. Se definen la matriz de distancias inicial $D^{(0)} = (D_{ij}^{(0)})_{i,j=1}^{n}$*

$$D_{ij}^{(0)} = \begin{cases} 0 & si\ i = j \\ p(i,j) = p_{ij} & si\ (i,j) \in A \\ +\infty & en\ otro\ caso \end{cases}$$

y la matriz de índices inicial $I^{(0)} = (I_{ij}^{(0)})_{i,j=1}^{n}$

$$I_{ij}^{(0)} = \begin{cases} i & si\ i \neq j \\ - & en\ otro\ caso \end{cases}$$

Si para cada $k \in \{1, 2, ..., n\}$, se definen recursivamente las matrices $D^{(k)}$ e $I^{(k)}$

$$D_{ij}^{(k)} = \text{mín}\left\{ D_{ij}^{(k-1)},\ D_{ik}^{(k-1)} + D_{kj}^{(k-1)} \right\}$$

$$I_{ij}^{(k)} = \begin{cases} I_{ij}^{(k-1)} & si\ D_{ij}^{(k)} = D_{ij}^{(k-1)} \\ I_{kj}^{(k-1)} & si\ D_{ij}^{(k)} \neq D_{ij}^{(k-1)} \end{cases}$$

y $D_{ij}^{(k)} \neq \infty$, entonces se verifica que $D_{ij}^{(k)}$ representa el valor de un camino de menor valor del nodo i a j tal que sus vértices intermedios son elementos del conjunto $\{1, 2, ..., k\}$ e $I_{ij}^{(k)}$ representa el predecesor de j en dicho camino para $i, j \in V$ con $i \neq j$.

Una vez formalizado el fundamento teórico del algoritmo, resulta de gran interés presentar una descripción de su funcionamiento a través de un pseudocódigo (ver algoritmo 1).

Algorithm 1 Pseudocódigo del algoritmo de Floyd–Warshall

Planteamiento: Se considera un grafo $G = (V, A, p)$ ponderado, dirigido, fuertemente conexo, de orden n y con pesos positivos $p(i, j) = p_{ij} \in \mathbb{R}^+$.

Objetivo: Encontrar un CMV entre cualquier par de vértices.

Algoritmo:

1. Inicializar las matrices:

$$D_{ij}^{(0)} = \begin{cases} 0 & \text{si } i = j \\ p_{ij} & \text{si } (i, j) \in A \\ \infty & \text{en otro caso} \end{cases}$$

$$I_{ij}^{(0)} = \begin{cases} i & \text{si } i \neq j \\ - & \text{en otro caso} \end{cases}$$

y considerar $k = 1$.

2. Definir los elementos de $D^{(k)}$ del siguiente modo:

$$D_{ij}^{(k)} = \text{mín}\{D_{ij}^{(k-1)}, D_{ik}^{(k-1)} + D_{kj}^{(k-1)}\}$$

3. Obtener los elementos de $I^{(k)}$ del siguiente modo:

$$I_{ij}^{(k)} = \begin{cases} I_{ij}^{(k-1)} & \text{si } D_{ij}^{(k)} = D_{ij}^{(k-1)} \\ I_{kj}^{(k-1)} & \text{si } D_{ij}^{(k)} \neq D_{ij}^{(k-1)} \end{cases}$$

4. Si $k < n$, hacer $k = k + 1$ e ir al paso 2. Si $k = n$, **PARAR**, ya se tienen los CMV entre todos los pares de vértices. En este punto $D^{(n)}$ contiene las distancias mínimas entre cada par (i, j), mientras que $I^{(n)}$ almacena los predecesores correspondientes que permiten reconstruir cada camino mínimo mediante un recorrido inverso desde el destino hasta el origen.

Esta formulación permite visualizar con claridad el proceso de actualización iterativa de las matrices de distancias e índices. A partir de la figura 2.15 se desarrolla el ejemplo 2.3, que ilustra su aplicación sobre un grafo sencillo, el cual facilita la comprensión de su lógica interna y su utilidad práctica en el contexto de las redes de transporte.

Ejemplo 2.3. *Ejemplo del algoritmo de Floyd–Warshall.*

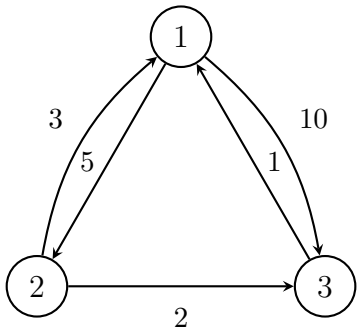


Figura 2.15: Grafo dirigido ponderado con pesos positivos.

- *Paso 1:*

$$D^{(0)} = \begin{pmatrix} 0 & 5 & 10 \\ 3 & 0 & 2 \\ 1 & \infty & 0 \end{pmatrix} \quad I^{(0)} = \begin{pmatrix} - & 1 & 1 \\ 2 & - & 2 \\ 3 & 3 & - \end{pmatrix} \quad k = 1$$

- *Pasos 2 y 3:*

$$D^{(1)} = \begin{pmatrix} 0 & 5 & 10 \\ 3 & 0 & 2 \\ 1 & 6 & 0 \end{pmatrix} \quad I^{(1)} = \begin{pmatrix} - & 1 & 1 \\ 2 & - & 2 \\ 3 & 1 & - \end{pmatrix}$$

- *Paso 4:* $k = 1 < n = 3 \rightarrow k = 2$ *y volver al paso 2.*

- *Pasos 2 y 3:*

$$D^{(2)} = \begin{pmatrix} 0 & 5 & 7 \\ 3 & 0 & 2 \\ 1 & 6 & 0 \end{pmatrix} \quad I^{(2)} = \begin{pmatrix} - & 1 & 2 \\ 2 & - & 2 \\ 3 & 1 & - \end{pmatrix}$$

- *Paso 4:* $k = 2 < n = 3 \rightarrow k = 3$ *y volver al paso 2.*
- *Pasos 2 y 3:*

$$D^{(3)} = \begin{pmatrix} 0 & 5 & 7 \\ 3 & 0 & 2 \\ 1 & 6 & 0 \end{pmatrix} \quad I^{(3)} = \begin{pmatrix} - & 1 & 2 \\ 2 & - & 2 \\ 3 & 1 & - \end{pmatrix}$$

- *Paso 4:* $k = 3 = n \rightarrow$ *PARAR.*

	CMV	**Valor**
1 a 2	$1 \rightarrow 2$	5
1 a 3	$1 \rightarrow 2 \rightarrow 3$	7
2 a 1	$2 \rightarrow 1$	3
2 a 3	$2 \rightarrow 3$	2
3 a 1	$3 \rightarrow 1$	1
3 a 2	$3 \rightarrow 1 \rightarrow 2$	6

2.2. Tipos de algoritmos para optimización

La resolución de problemas de optimización combinatoria en grafos, como el diseño de rutas de transporte urbano, requiere la utilización de procedimientos computacionales capaces de explorar de forma eficiente espacios de soluciones de gran dimensión y con restricciones complejas. Un algoritmo puede entenderse, en términos generales, como un conjunto finito de instrucciones bien definidas que permiten resolver una tarea o problema determinado mediante una secuencia de pasos. Tradicionalmente, estos procedimientos se clasifican en algoritmos exactos, heurísticos y metaheurísticos (Antón, 2025).

2.2.1. Algoritmos exactos

Los algoritmos exactos son métodos deterministas que aseguran la obtención de una solución óptima en un número finito de pasos. Engloban técnicas como la programación lineal entera, los algoritmos de ramificación y poda (*branch and bound*) o los métodos de programación dinámica. A pesar de su rigor matemático, su uso se ve severamente limitado por el crecimiento exponencial del espacio de búsqueda en los problemas computacionalmente complejos, como es el caso del URTP.

2.2.2. Algoritmos heurísticos

El término "heurístico" proviene del griego *heuriskein*, que significa encontrar o descubrir. En el ámbito de la optimización, los algoritmos heurísticos surgen como respuesta a las limitaciones de los métodos exactos, ofreciendo procedimientos prácticos capaces de obtener soluciones satisfactorias sin necesidad de explorar exhaustivamente todo el espacio de búsqueda. Son estrategias diseñadas específicamente para aprovechar la estructura del problema y priorizan la eficiencia computacional frente a la garantía de optimalidad.

Los métodos heurísticos pueden clasificarse en distintas familias, como son los métodos constructivos, que generan soluciones paso a paso, o los de búsqueda local, que mejoran soluciones ya existentes mediante pequeñas modificaciones. También se incluyen enfoques inductivos, de descomposición y de reducción, cada uno con su propia lógica de simplificación del problema.

En el ámbito del diseño de redes de transporte, las heurísticas han demostrado su utilidad en la generación de soluciones iniciales viables, en la resolución de subproblemas

específicos y como componentes dentro de esquemas híbridos. Su flexibilidad y rapidez las hacen atractivas para problemas con restricciones complejas, aunque su dependencia de la estructura del problema y la ausencia de garantías de optimalidad han impulsado el desarrollo de metaheurísticas más sistemáticas (Cunquero, 2003).

2.2.3. Algoritmos metaheurísticos

Por encima de estos últimos se sitúan los algoritmos metaheurísticos, entendidos como marcos de optimización de alto nivel que guían la búsqueda de soluciones en espacios de gran complejidad. A diferencia de las heurísticas, que aplican reglas específicas sobre un problema concreto, las metaheurísticas emplean mecanismos más generales que permiten explorar múltiples regiones del espacio de búsqueda, favoreciendo un equilibrio entre exploración y explotación. Este enfoque se ha consolidado como una herramienta fundamental en la resolución de problemas de optimización complejos, al combinar simplicidad, flexibilidad y eficacia en contextos donde otros métodos resultan ineficientes o inaplicables (Tomar et al., 2024).

Este tipo de técnicas son especialmente eficaces en problemas donde no se pueden aplicar los métodos exactos debido a su coste computacional, y donde las heurísticas tradicionales tienden a quedarse atrapadas en óptimos locales. Su versatilidad radica en que pueden combinarse con heurísticas particulares, adaptarse a distintas estructuras de problema y operar bajo condiciones de incertidumbre o múltiples restricciones. A menudo se inspiran en fenómenos naturales o sociales, como la evolución biológica, el comportamiento de los enjambres o el proceso de enfriamiento de los metales.

Entre las familias más destacadas de metaheurísticas se encuentran (Blum and Roli, 2003):

- Los algoritmos basados en trayectorias, como *Simulated Annealing* y *Tabu Search*, que exploran soluciones vecinas mediante estrategias de mejora controladas.
- Los algoritmos por colonias, como *Ant Colony Optimization* y *Particle Swarm Optimization*, inspirados en el comportamiento colectivo de insectos o aves, ampliamente utilizados en problemas de diseño y planificación.
- Los algoritmos evolutivos, que trabajan sobre poblaciones de soluciones y emplean operadores como la selección, el cruce o la mutación para evolucionar soluciones a lo largo de generaciones. Estos últimos son particularmente adecuados para problemas multiobjetivo gracias a su capacidad para mantener diversidad y aproximar el frente de Pareto (Deb, 2001).

Dentro del enfoque evolutivo, destacan los algoritmos evolutivos multiobjetivo (MOEAs, por sus siglas en inglés *Multi-Objective Evolutionary Algorithms*), cuya finalidad es obtener un conjunto de soluciones eficientes respetando varios criterios en conflicto. Estos algoritmos evolucionan una población de soluciones, priorizando aquellas no dominadas según criterios de dominancia de Pareto y diversidad. En la optimización multiobjetivo no suele existir una única solución que optimice simultáneamente todos los criterios, sino un conjunto de alternativas que representan distintos compromisos entre ellos (Coello, 2010).

Definición 2.23. *Una* ***solución*** *se considera* ***no dominada*** *cuando no existe otra que mejore alguno de los objetivos sin empeorar al menos otro.*

Definición 2.24. *El conjunto formado por todas estas soluciones no dominadas se conoce como* ***frente de Pareto****.*

Este conjunto constituye la referencia frente a la cual se evalúa la calidad de los algoritmos multiobjetivo, ya que representa el equilibrio entre los distintos criterios en conflicto.

Una subcategoría relevante es la de los MOEAs de estado estacionario (*steady-state* MOEAs), en los que la población se actualiza de forma gradual. En este esquema, cada individuo generado se evalúa de inmediato y puede reemplazar a otro si lo mejora en términos de dominancia. Esta estrategia permite una incorporación más rápida de mejoras, reduce la necesidad de evaluaciones globales y facilita el mantenimiento de diversidad sin mecanismos auxiliares complejos (Mumford, 2004).

Uno de los algoritmos más representativos de este grupo es el SEAMO2 (por sus siglas en inglés *Simple Evolutionary Algorithm for Multiobjective-Optimization*, versión 2), el cual emplea un mecanismo de reemplazo directo basado en la dominancia paretiana. Su arquitectura sencilla facilita la incorporación de operadores específicos adaptados al problema, como generadores iniciales de soluciones, mutaciones estructurales o procedimientos de reparación. En el caso del URTP, su implementación incluye operadores diseñados expresamente para mantener la conectividad de la red y asegurar la factibilidad de las soluciones (Mumford, 2013).

En este trabajo, se adopta SEAMO2 como algoritmo de referencia para abordar el problema de diseño de rutas de transporte urbano. La justificación detallada de esta elección ya ha sido expuesta en la sección introductoria, por lo que a continuación se procede a definir formalmente el problema y describir cómo se adapta el algoritmo a sus restricciones y objetivos específicos.

Capítulo 3

Planteamiento del problema de rutas de transporte urbano

En este capítulo se plantea formalmente el URTP a partir de los fundamentos teóricos expuestos en el capítulo anterior. Se establecen las bases para su modelización mediante grafos, definiendo las restricciones y los criterios de optimización que lo describen, así como el enfoque evolutivo adoptado para su resolución.

3.1. Introducción general y conceptos básicos

Aunque en el marco teórico del capítulo anterior se han introducido los conceptos fundamentales desde la teoría de grafos dirigidos, en adelante se adopta el marco de los grafos no dirigidos. Esta elección responde a una simplificación habitual en el modelado del

problema, en la que se asume que los tiempos de viaje, las distancias y la demanda entre dos nodos son simétricos, de modo que no dependen de la dirección del desplazamiento. De esta forma, el planteamiento del URTP se formula sobre grafos no dirigidos, lo que permite definir de manera precisa las restricciones estructurales y los objetivos de optimización que caracterizan al problema.

El URTP constituye uno de los principales desafíos en la planificación de redes de transporte público urbano. Su objetivo es diseñar y optimizar la configuración de rutas que conectan las diferentes paradas de una ciudad, considerando de manera conjunta factores como el tiempo de viaje, la demanda de pasajeros, las conexiones por transbordos, la capacidad de los vehículos, los costes operativos y, en enfoques más recientes, el impacto ambiental. El problema se formula habitualmente como un modelo de optimización en el que se busca minimizar los costes globales del sistema manteniendo unos niveles de servicio aceptables para los usuarios. La dificultad de este planteamiento radica en el número exponencial de posibles rutas y en la diversidad de restricciones que deben satisfacerse, lo que convierte al URTP en un problema de gran complejidad combinatoria (Kourepinis et al., 2023).

A lo largo de las últimas décadas se han propuesto numerosos enfoques para abordar el URTP. Desde métodos heurísticos hasta algoritmos metaheurísticos más sofisticados que tratan de buscar un equilibrio entre la eficiencia computacional y la calidad de las soluciones. En este sentido, el URTP se ha consolidado como un caso paradigmático de problema de optimización multiobjetivo, ya que busca al mismo tiempo reducir el tiempo medio de viaje de los pasajeros y limitar los costes de operación del sistema. Esta doble visión justifica el uso de algoritmos evolutivos como SEAMO2, que permiten manejar de forma eficaz estos objetivos contrapuestos.

A partir de esta caracterización general, resulta necesario introducir algunos conceptos para poder modelar el problema. El punto de partida es la red de transporte, que se define como un grafo no dirigido $G = (V, A)$ de orden n en el que los nodos representan las paradas o estaciones y las aristas los enlaces de transporte directos que existen entre dichas paradas.

Sobre esta base se introduce la red de rutas asociada a un conjunto de rutas determinado, entendida como el subgrafo de la red de transporte que contiene únicamente las aristas que aparecen en al menos una ruta del conjunto, de modo que refleja la selección concreta de trayectos que configuran un plan de servicio. Cada ruta se interpreta como un camino no dirigido, dado que los tiempos de viaje y los niveles de demanda son simétricos en ambos sentidos.

El objetivo es encontrar el conjunto de rutas óptimo que satisfaga los criterios de evaluación. Para evaluar un conjunto de rutas candidato, se construye la red de tránsito. En esta red, los enlaces de transporte corresponden a las conexiones entre dos nodos que pertenecen a una misma ruta, mientras que los enlaces de transbordo representan los posibles cambios de una ruta a otra en un mismo nodo. De este modo, la red de tránsito refleja con mayor precisión el funcionamiento real del sistema, al incorporar tanto los desplazamientos directos como los transbordos que deben realizar los pasajeros.

Para ilustrar los conceptos definidos, se proporciona a continuación un ejemplo práctico. Para ello, se toma como referencia el mapa de las conexiones de tren de alta velocidad en España en 2025 (líneas Renfe AVE, Avlo, Ouigo e Iryo). A partir de esta información se ha programado, con el uso de los paquetes *networkx* y *matplotlib* de Python, una serie de grafos que muestran, respectivamente, un ejemplo de red de transporte, una red de rutas y una red de tránsito.

A partir del mapa de la figura 3.1, se ha construido un grafo no dirigido que recoge las ciudades como nodos y las conexiones de alta velocidad como aristas, obteniendo así la red de transporte de referencia, la cual se puede apreciar en la figura 3.2.

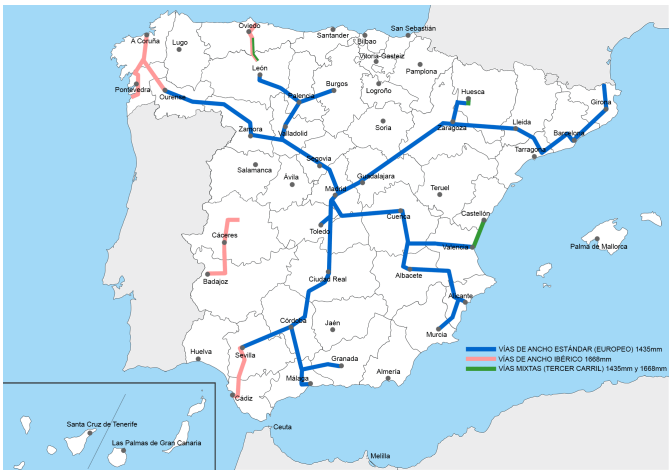


Figura 3.1: Mapa de trenes de alta velocidad en España 2025[1].

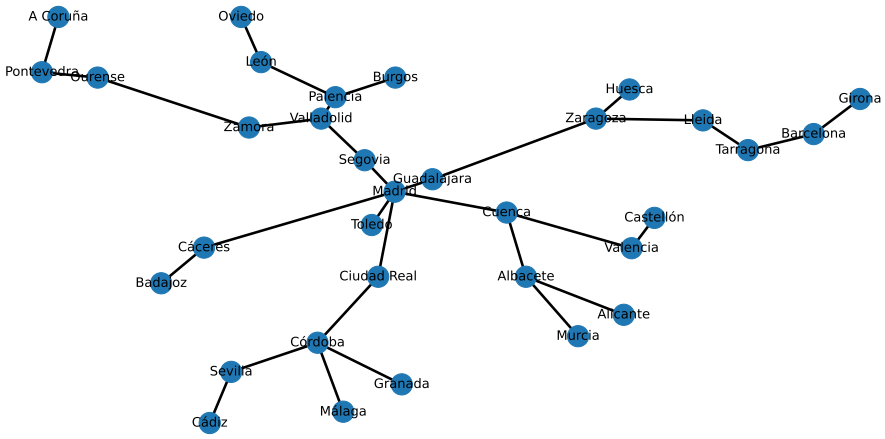


Figura 3.2: Red de transporte: conexiones de tren de alta velocidad en España.

[1]Mapa extraído de: https://www.enterat.com/servicios/mapa-ave-espana.php.

En este caso se ha considerado un subgrafo de la red de transporte y un conjunto formado por tres rutas de ejemplo. La figura 3.3 muestra, de forma conjunta, la red de rutas asociada a dicho conjunto y la red de tránsito derivada, en la que se distinguen los enlaces de transporte dentro de cada ruta y los enlaces de transbordo en los nodos comunes. Ambas representaciones comparten leyenda y se han construido manualmente a modo de ejemplo.

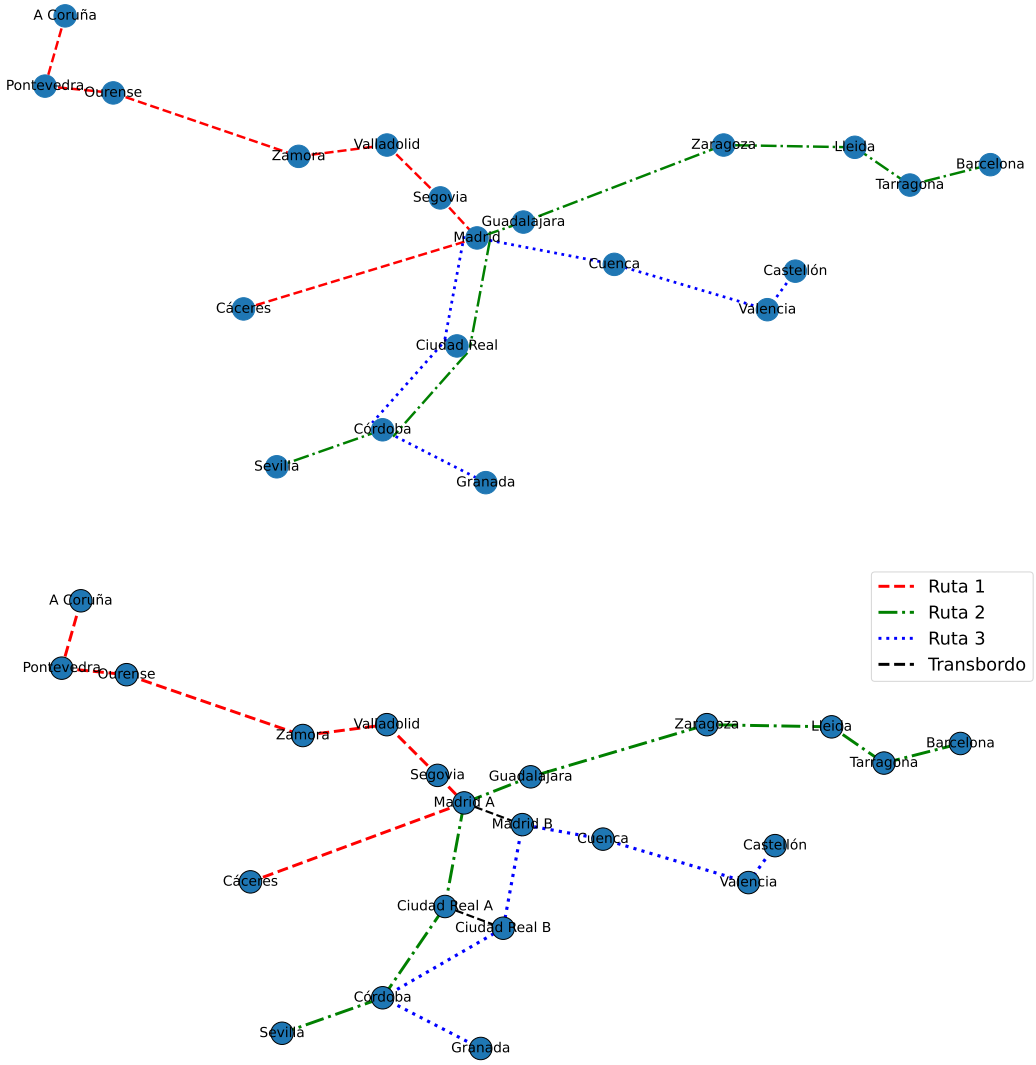


Figura 3.3: Red de rutas y de tránsito: conexiones de tren de alta velocidad en España.

3.2. Restricciones del problema

En el modelado del URTP, las restricciones desempeñan un papel determinante. Su función es doble: por un lado, acotar el espacio de búsqueda de soluciones evitando configuraciones inviables, y por otro, garantizar que las rutas obtenidas reflejen características realistas desde el punto de vista operativo y del servicio al pasajero.

En consecuencia, resulta necesario fijar una serie de condiciones básicas que definan qué configuraciones de rutas pueden aceptarse como válidas. Estas restricciones recogen tanto limitaciones estructurales como consideraciones prácticas del transporte urbano, entre las que se incluyen la conectividad de la red, la exigencia de longitudes mínimas y máximas de recorrido, la prohibición de ciclos o bucles innecesarios, o la disponibilidad de suficientes vehículos para satisfacer la demanda. A continuación, se enumeran las más relevantes en el marco del problema.

1. La red de rutas debe ser conexa, de forma que exista al menos un camino entre cualquier par de nodos de la red.
2. Cada ruta debe respetar una longitud máxima prefijada, asociada a la dificultad de mantener horarios estables y a la fatiga de los conductores.
3. Cada ruta debe respetar una longitud mínima prefijada, con el fin de evitar recorridos demasiado cortos que reduzcan la eficiencia del sistema.
4. Ninguna ruta puede contener ni bucles ni ciclos.
5. El conjunto de rutas debe contener exactamente r rutas, siendo r un parámetro fijado de antemano.

6. La demanda, los tiempos de viaje y las distancias se consideran simétricos. Cada vehículo recorre la ruta en ambos sentidos, invirtiendo la dirección al llegar a un nodo terminal.

7. Se supone que el nivel de demanda permanece constante durante el periodo de estudio.

8. Se establece una penalización fija de 5 minutos por cada transbordo como representación del inconveniente que supone cambiar de un vehículo a otro.

9. No se tiene en cuenta la frecuencia de los vehículos, se supone que siempre hay suficientes vehículos y capacidad. De este modo, el tiempo total de viaje es la suma del tiempo de tránsito en el vehículo y las penalizaciones de 5 minutos por cada transbordo. Esta simplificación, aunque no es totalmente realista, es necesaria para reducir notablemente la complejidad computacional del problema, permitiendo centrar la optimización en el diseño de rutas.

10. La elección de las rutas por parte del pasajero se basa en el tiempo de viaje más corto.

3.3. Funciones objetivo y evaluación de las soluciones

Una vez establecidas las restricciones del problema, es necesario definir los criterios mediante los cuales se valora la calidad de un conjunto de rutas candidato. Se consideran dos funciones objetivo de carácter contrapuesto: el coste del pasajero y el coste del operador. Estas medidas permiten evaluar la eficiencia del sistema desde dos perspectivas complementarias, la del usuario y la del gestor de la red.

3.3.1. Coste del pasajero

Por norma general, los pasajeros tienden a preferir trayectos que minimicen el tiempo total del viaje y eviten, en la medida de lo posible, un número elevado de transbordos. Para modelar este comportamiento, se supone que cada viajero escoge siempre el camino de menor duración entre el origen y el destino. Dicho tiempo mínimo de viaje, denotado por $\alpha_{od}(R)$, para cualquier par de nodos o = origen y d = destino en un conjunto de rutas R, está compuesto por dos elementos: el tiempo de recorrido en el vehículo y las penalizaciones fijas de 5 minutos por cada transbordo.

Sea d_{ij} la demanda de viajes entre los nodos i y j, es decir, el número de pasajeros que desean viajar desde i hasta j. Se define el coste del pasajero para un conjunto de rutas R, denotado como $C_P(R)$, como el tiempo medio de viaje de todos los pasajeros, es decir:

$$C_P(R) \;=\; \frac{\displaystyle\sum_{i,j=1}^{n} d_{ij}\,\alpha_{ij}(R)}{\displaystyle\sum_{i,j=1}^{n} d_{ij}}.$$

donde $\alpha_{ij}(R)$ es el tiempo de viaje más corto desde i hasta j utilizando el conjunto de rutas R y n es el número de nodos.

3.3.2. Coste del operador

Desde la perspectiva del gestor del sistema, el coste está vinculado a factores como el número de vehículos necesarios, el kilometraje total recorrido y el esfuerzo operativo asociado al servicio. En este trabajo se emplea como medida aproximada el coste del

operador, denotado por C_O, definido como la suma de los tiempos necesarios para recorrer todas las rutas que conforman el plan de servicio:

$$C_O = \sum_{a \in R} \sum_{\{i,j\} \in a} t_{ij}(a),$$

siendo a una ruta cualquiera de R y $t_{ij}(a)$ el tiempo asociado al enlace de transporte $\{i, j\} \in a$.

En consecuencia, el URTP se plantea como un modelo de optimización con dos objetivos en conflicto. Por un lado, reducir el tiempo medio de viaje de los pasajeros, y por otro, limitar la longitud total (en tiempo) de las rutas que conforman el sistema. Mejorar el primero suele implicar rutas más extensas y directas, mientras que el segundo favorece recorridos más cortos y económicos. Esta tensión entre calidad del servicio y costes operativos motiva el empleo de métodos de optimización capaces de equilibrar ambos criterios de manera conjunta.

3.4. Otros parámetros y consideraciones adicionales

Además de los costes C_P y C_O, resulta útil emplear indicadores que midan con mayor detalle la calidad de un conjunto de rutas. Se asume que en ningún caso se permiten más de dos transbordos por viaje. Esta condición no se impone como restricción del modelo, sino como un criterio de evaluación: los desplazamientos que requieren más de dos transbordos se contabilizan como demanda insatisfecha. Se consideran, por tanto, los siguientes parámetros:

- $d_0 \equiv$ porcentaje de demanda satisfecha sin transbordos.

- $d_1 \equiv$ porcentaje de demanda satisfecha con un transbordo.
- $d_2 \equiv$ porcentaje de demanda satisfecha con dos transbordos.
- $d_{un} \equiv$ porcentaje de demanda insatisfecha.

3.5. Desarrollo y explicación del algoritmo

El algoritmo seleccionado para resolver el URTP ha sido SEAMO2, una variante de los algoritmos evolutivos multiobjetivo de estado estacionario. Su elección se justifica por dos motivos fundamentales. Por un lado, la simplicidad de su estructura y la facilidad de implementación. Y por otro lado, la eficacia mostrada en la resolución de problemas de gran tamaño, ofreciendo resultados de calidad en tiempos razonables, como ya se comentó en capítulos anteriores.

A diferencia de otros enfoques más complejos, SEAMO2 utiliza una estrategia de reemplazo muy simple: cada nueva solución puede sustituir a otra existente si la mejora en alguno de los objetivos. Gracias a este funcionamiento se reducen los tiempos de ejecución y se preserva la variedad en las soluciones sin necesidad de apoyarse en mecanismos adicionales.

En este trabajo, el algoritmo se ha adaptado a las particularidades del URTP respetando las restricciones previamente establecidas y se ha programado en Python siguiendo las explicaciones y el pseudocódigo propuesto en Mumford (2013). A continuación se describe su funcionamiento, desde la representación y evaluación de las soluciones hasta la generación de rutas, los operadores genéticos, el reemplazo y los mecanismos de reparación que garantizan la viabilidad de las rutas.

Cada solución del problema se representa como un individuo, entendido como un conjunto de rutas que cubre la red de transporte. Dichas rutas se almacenan como secuencias de nodos que indican las paradas recorridas, de manera que un individuo representa una propuesta completa de diseño de red.

A partir de cada conjunto de rutas se construye una red de tránsito asociada, en la que se distinguen los enlaces de transporte, que corresponden a los tramos consecutivos dentro de una misma ruta, frente a los de transbordo, que conectan ocurrencias del mismo nodo en distintas rutas e incorporan la penalización fija por cambio de vehículo. Esta construcción permite modelar de manera más realista el desplazamiento de los pasajeros en el sistema.

La evaluación de cada individuo se lleva a cabo mediante los dos criterios ya presentados: el coste del pasajero (C_P), que refleja el tiempo medio de viaje, y el coste del operador (C_O), que refleja el coste global asociado a la operación de la red. Ambos indicadores ofrecen visiones complementarias y, al estar en conflicto, justifican el enfoque multiobjetivo adoptado.

El cálculo del pasajero, C_P, constituye la parte más exigente desde el punto de vista computacional, ya que requiere conocer los tiempos mínimos de trayecto entre todos los pares de nodos en la red de tránsito. Para resolver este problema se emplea el algoritmo de Floyd–Warshall, el cual permite obtener de manera eficiente la matriz de distancias mínimas. En esta implementación, se ejecuta una sola vez por cada individuo, lo que constituye una mejora importante respecto a otros enfoques y permite reducir el tiempo de cálculo.

La primera fase del algoritmo es la generación de la población inicial. El procedimiento empleado consiste en crear rutas de manera progresiva, nodo a nodo, hasta completar el

número de rutas r establecido en las restricciones del problema. Tanto el número de rutas como sus longitudes se encuentran regulados por dos parámetros, MIN y MAX, que son fijados por el usuario. Para cada ruta se elige al azar una longitud comprendida entre estos valores, de forma que la población inicial presente diversidad en sus configuraciones.

La construcción de rutas comienza con la selección de un nodo semilla. En la primera ruta, este nodo se elige de forma aleatoria entre aquellos con conexiones disponibles en la red de transporte. En las rutas siguientes, en cambio, se favorece la elección de nodos que ya han aparecido en rutas anteriores, con el fin de garantizar la conectividad global del conjunto de rutas desde el inicio.

Una vez elegido el nodo de partida, la ruta se amplía añadiendo nodos adyacentes que todavía no forman parte de ella, evitando de este modo la aparición de ciclos. Este proceso se repite hasta alcanzar la longitud asignada a la ruta o hasta que no sea posible añadir más nodos válidos. En este último caso, el algoritmo incorpora la estrategia denominada *reverse*. Ésta consiste en que si la ruta queda bloqueada antes de completarse, se invierte el sentido y se intenta continuar la construcción desde el otro extremo. Si aun así no se logra alcanzar la longitud deseada, la ruta se mantiene con la extensión obtenida.

El mismo procedimiento se repite para las siguientes rutas, de modo que la segunda se inicializa a partir de un nodo presente en la primera, la tercera desde un nodo que aparezca en cualquiera de las dos anteriores, y así sucesivamente. De esta forma se asegura que todas las rutas tengan al menos un punto de conexión con alguna de las ya creadas, manteniendo la cohesión de la red.

Una vez construidas las r rutas, se obtiene un conjunto inicial que suele cubrir la mayor parte de la red, aunque en ocasiones pueden quedar nodos sin asignar. Para solventar este

problema se aplica un procedimiento de reparación, en el que se recorre la lista de nodos no incluidos y se insertan, siempre que sea posible, en los extremos de alguna de las rutas existentes. Esta operación solo se lleva a cabo cuando existe adyacencia en la red de transporte y sin superar la longitud máxima permitida. En caso de que no sea posible ubicar algún nodo, el conjunto de rutas se descarta y se genera otro nuevo. Se repite este proceso hasta completar la población inicial, formada por un número N de conjuntos de rutas. Este tamaño de población constituye un parámetro de entrada definido por el usuario.

Con esta combinación de generación y reparación se garantiza que los individuos iniciales cumplen las restricciones estructurales definidas en la Sección 3.2: conectividad de la red de rutas, número fijo de rutas r, longitudes mínima y máxima y ausencia de ciclos.

Una vez generada la población inicial, el algoritmo aplica sobre ella una serie de operadores genéticos diseñados de forma específica para el problema. Estos operadores permiten crear nuevas soluciones a partir de las ya existentes, generando diversidad en la población y favoreciendo la exploración de distintas configuraciones de rutas.

El primero de estos operadores es el cruce. Su objetivo es combinar dos individuos (padres) para generar un nuevo conjunto de rutas (hijo). El procedimiento se inspira en un proceso de alternancia, de modo que se toma una primera ruta semilla de uno de los padres, y a partir de ella, se van seleccionando rutas de manera alterna del otro progenitor. Para que una ruta sea considerada, debe compartir al menos un nodo con las que ya forman parte del hijo, garantizando así la conectividad del conjunto de rutas. Entre las rutas elegibles, se priorizan aquellas que incorporan un mayor número de nodos nuevos, de forma que el hijo aporte cobertura adicional a la red. El proceso se repite hasta alcanzar el número de rutas requerido, completando finalmente con trayectos simples (entendidos

como rutas cortas y directas que garantizan la cobertura completa de la red) en caso de ser necesario. Con este mecanismo se consigue preservar la estructura de los padres al tiempo que se introducen elementos novedosos que enriquecen la diversidad.

El segundo operador es la mutación, que introduce cambios locales en un individuo para explorar nuevas configuraciones sin alterar en exceso su estructura. Las modificaciones se producen en los extremos de las rutas, evitando así que se generen ciclos y reduciendo el riesgo de generar soluciones inviables. Se contemplan dos tipos de mutación: adición y eliminación. En el caso de la adición, se incorpora un nodo adyacente al extremo de una ruta siempre que no genere un ciclo y que la longitud resultante no supere el máximo permitido. En el caso de la eliminación, se elimina un nodo terminal de la ruta, pero solo si ya aparece en otra de las rutas del conjunto, de forma que no se pierda cobertura, y respetando siempre la longitud mínima prefijada. El número de mutaciones aplicadas a cada individuo se elige de forma aleatoria dentro de un intervalo proporcional a su tamaño, concretamente $I \in \left[1, \frac{r \cdot MAX}{2}\right]$ con $I \in \mathbb{Z}$, donde r es el número de rutas del conjunto y MAX la longitud máxima permitida para cada ruta. En cada operación, la elección entre añadir o eliminar un nodo se realiza con la misma probabilidad, equilibrando la exploración de rutas más largas con la simplificación de otras ya existentes. Este planteamiento proporciona variabilidad de manera controlada y evita el fenómeno del *bloating*, que consiste en la generación de rutas innecesariamente extensas y poco útiles.

Tras la aplicación de estos operadores evolutivos, es posible que queden nodos de la red sin cubrir, por lo que se aplica nuevamente el procedimiento de reparación descrito anteriormente. Con ello, se garantiza que las soluciones resultantes conserven la viabilidad y cumplan las restricciones de conectividad, cobertura y longitud establecidas.

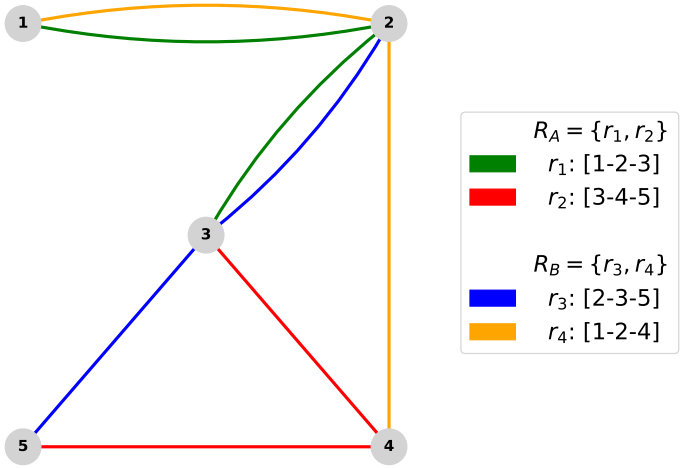


Figura 3.4: Red de transporte para ilustrar el funcionamiento de los operadores genéticos.

El funcionamiento de estos operadores puede ilustrarse con un ejemplo sencillo. Supongamos una red de transporte formada por cinco nodos numerados del 1 al 5. Un primer individuo R_A contiene las rutas $r_1 = [1-2-3]$ y $r_2 = [3-4-5]$, mientras que otro individuo R_B incluye $r_3 = [2-3-5]$ y $r_4 = [1-2-4]$. Es decir, $R_A = \{r_1, r_2\}$ y $R_B = \{r_3, r_4\}$, como se puede apreciar en la figura 3.4.

Para realizar el cruce entre los progenitores R_A y R_B, se considera, por ejemplo, r_1 como ruta inicial. En el segundo progenitor, pueden elegirse dos candidatas: r_3 y r_4, ya que ambas comparten nodos con la semilla. A su vez, ambas aportan nodos nuevos (r_3 introduce el nodo 5 y r_4 el nodo 4), por lo que, en caso de empate en el número de nodos nuevos aportados, la selección se realiza de forma aleatoria. En este ejemplo, se elige r_3. Así, el hijo resultante contendría r_1 y r_3, es decir, $R_{hijo} = \{r_1, r_3\}$ (ver figura 3.5).

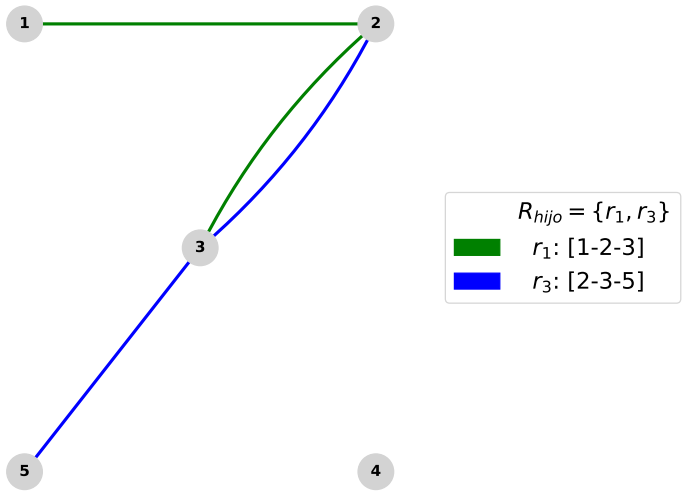


Figura 3.5: Hijo resultante del cruce explicado entre R_A y R_B.

A continuación, podría aplicarse una mutación de tipo adición sobre r_1, extendiéndola con el nodo 4 hasta $[1-2-3-4]$, siempre que no se exceda la longitud máxima. En cambio, una mutación de tipo eliminación sobre el nodo 1 no sería posible, ya que ese nodo no aparece en ninguna otra ruta del conjunto y suprimirlo supondría perder cobertura. Finalmente, si tras estas operaciones quedara algún nodo sin cubrir, el procedimiento de reparación trataría de añadirlo al extremo de una ruta adyacente, completando así la cobertura.

En definitiva, estos operadores permiten que la población explore nuevas configuraciones manteniendo siempre la viabilidad de las soluciones, lo que asegura que el proceso evolutivo avance hacia propuestas de calidad para el URTP.

Una vez explicados los procedimientos de generación de la población inicial, evaluación y los operadores evolutivos, ya es posible describir cómo funciona el algoritmo SEAMO2

aplicado al URTP. Se trata de un MOEA de estado estacionario, en el que en cada iteración se genera un único individuo y se decide inmediatamente si se incorpora a la población.

El algoritmo comienza con la generación de una población inicial de soluciones factibles, obtenida mediante el crecimiento nodo a nodo y el procedimiento de reparación. Cada individuo es evaluado de acuerdo con las dos funciones objetivo consideradas: C_P y C_O. A partir de esta evaluación inicial se almacenan los mejores valores obtenidos para cada uno de los objetivos, que sirven como referencia para el seguimiento de la evolución.

A continuación, se repite un ciclo evolutivo durante el número de generaciones establecido. En cada paso, se seleccionan dos progenitores de la población actual y se aplica el operador de cruce. El hijo resultante se somete de inmediato al proceso de reparación, con el fin de garantizar cobertura y viabilidad antes de continuar. Seguidamente, se aplica la mutación. que introduce pequeñas modificaciones en los extremos de las rutas mediante operaciones de adición o eliminación elegidas con igual probabilidad, en un número aleatorio de iteraciones dentro del intervalo definido. Una vez completado, el nuevo individuo se evalúa nuevamente en los dos objetivos y se procede a la fase de reemplazo.

Con el hijo ya evaluado, el algoritmo decide si debe incorporarse a la población aplicando un sencillo esquema de comprobaciones. En primer lugar, se compara con los dos progenitores: si el hijo resulta mejor que alguno de ellos en uno de los objetivos sin empeorar en el otro, lo sustituye directamente. Si no supera a ninguno de los padres, se comprueba si mejora el mejor valor alcanzado hasta ese momento en alguno de los dos costes. En ese caso, se incorpora y reemplaza al progenitor con peor suma de costes, actualizando el registro correspondiente. Esta suma se emplea únicamente como criterio sencillo de desempate. Aunque C_P y C_O miden aspectos distintos, su combinación ofrece una referencia práctica del rendimiento global. Se ha empleado esta versión simplificada

del reemplazo de original de SEAMO2, pues mantiene su idea básica de actualización inmediata y eficiente. En cualquier otro caso, el hijo se descarta y la población permanece sin cambios.

El ciclo evolutivo se repite hasta alcanzar el número máximo de generaciones fijado por el usuario. Una vez cumplida esta condición de parada, el algoritmo concluye la producción de descendientes y se obtienen los resultados finales.

También se calculan los indicadores d_0, d_1, d_2 y d_{un}, que permiten evaluar la calidad de las soluciones desde la perspectiva del usuario. Para cada par, origen (o) - destino (d) con demanda d_{od}, se obtiene, sobre la red de tránsito, el camino de menor tiempo y el número de transbordos asociados. Si el trayecto se realiza sin cambios de ruta, la demanda correspondiente contribuye a d_0; si implica un único transbordo, se suma a d_1; y si requiere dos, a d_2. Toda la demanda que no pueda servirse con dos transbordos o menos, ya sea porque el par no queda conectado en la red de tránsito o porque requeriría más enlaces de transbordo, se contabiliza en d_{un}.

Al finalizar la ejecución, el algoritmo imprime el conjunto de soluciones no dominadas obtenidas. Cada una se presenta con el valor de los dos costes, C_P y C_O, seguido de la lista de rutas que conforman el individuo. En la línea siguiente se muestran los indicadores d_0, d_1, d_2 y d_{un}, expresados como porcentajes de la demanda total. Gracias a este formato, los resultados no se limitan a cifras globales, sino que también ofrecen una representación detallada de cada propuesta de red, lo que facilita tanto la comparación de soluciones en términos de costes como la interpretación práctica de las configuraciones obtenidas.

El funcionamiento global del algoritmo SEAMO2 puede resumirse de manera esquemática en el siguiente pseudocódigo (ver algoritmo 2):

Algorithm 2 Pseudocódigo del algoritmo SEAMO2 para el URTP

Planteamiento: Se parte de una serie de conjuntos de datos que describen la red de transporte y se adapta al URTP para su resolución mediante el algoritmo SEAMO2.

Objetivo: Obtener un conjunto de soluciones no dominadas que equilibren el coste del pasajero y el coste del operador.

Algoritmo:

Generaración de la población inicial

Cálculo de C_P y C_O para cada individuo

Registro del mejor individuo para cada objetivo

repeat

  Seleccionar dos progenitores p_1 y p_2

  $hijo \leftarrow$ Cruce(p_1,p_2)

  $hijo \leftarrow$ Reparación($hijo$)

  $hijo \leftarrow$ Mutación($hijo$) *(Adición/eliminación con probabilidad 1/2)*

  Cálculo de C_P y C_O del $hijo$

  if $hijo$ domina a p_1 **then**

    Reemplazar p_1 por $hijo$

  else if $hijo$ domina a p_2 **then**

    Reemplazar p_2 por $hijo$

  else if $hijo$ mejora alguno de los mejores costes registrados **then**

    if $(C_P(p_1) + C_O(p_1)) > (C_P(p_2) + C_O(p_2))$ **then**

      Reemplazar p_1 por $hijo$

    else

      Reemplazar p_2 por $hijo$

    end if

  else

    Descartar $hijo$

  end if

  Actualización de mejores costes si procede

until se alcance el número máximo de generaciones

Impresión de las soluciones no dominadas de la población

Cálculo de los indicadores d_0, d_1, d_2 y d_{un} para cada solución

Capítulo 4

Casos prácticos

En este capítulo se ejecuta el algoritmo SEAMO2 implementado y descrito en la Sección 3.5, con el objetivo de analizar su comportamiento y el compromiso entre los costes del pasajero y del operador.

Para fijar un marco de trabajo uniforme y separar los datos de la lógica del algoritmo, todas las ejecuciones se realizan a partir de una estructura común de bases de datos que recoge la información de nodos, enlaces y demanda de pasajeros. Este formato permite organizar los datos de manera clara y facilita su lectura y procesamiento desde distintos entornos de programación.

La primera base de datos recoge el inventario de paradas de la red de transporte. Cada fila corresponde a un nodo e incluye cuatro campos:

- *id*: número entero que identifica de forma única la parada.

- *name*: nombre o etiqueta de la parada.
- *lat*: coordenada de latitud en grados decimales (de -90 a 90): positiva cuando está hacia el norte y negativa cuando está hacia el sur.
- *lon*: coordenada de longitud en grados decimales (de -180 a 180): positiva cuando está hacia el este y negativa cuando está hacia el oeste.

En la ejecución del algoritmo se emplea exclusivamente el *id* de los nodos. Mientras que *name*, *lat* y *lon* se incluyen para facilitar la identificación y representación gráfica, pero no intervienen en la optimización.

La segunda base de datos recoge las conexiones directas posibles entre paradas. Cada fila aporta información procedente de tres columnas:

- *from*: identificador de la parada de origen.
- *to*: identificador de la parada de destino.
- *travel_time*: tiempo de viaje asociado a dicha conexión, expresado en minutos.

Así, cada fila representa una arista de la red de transporte con su par de nodos y el peso asociado al tiempo de recorrido que se utilizará como medida de coste para esa relación directa.

La tercera base de datos contiene la demanda entre dos nodos origen-destino, de modo que cada fila presenta tres columnas:

- *from*: identificador de la parada de origen.

- *to*: identificador de la parada de destino.
- *demand*: número de viajeros que desean viajar desde ese origen hasta ese destino.

Por tanto, especifica los volúmenes de desplazamiento entre pares de nodos y permite identificar qué relaciones tienen mayor o menor intensidad de uso.

Es importante señalar que tanto los tiempos de viaje como los niveles de demanda deben ser simétricos, ya que el modelo trabaja con grafos no dirigidos. En consecuencia, el tiempo y la demanda entre dos nodos i y j cualesquiera son iguales en ambos sentidos, cumpliendo así las restricciones establecidas en el planteamiento del problema.

A partir de estas tres bases de datos se carga en memoria el grafo de transporte, los tiempos de viaje y la matriz de demanda. Se fija la penalización por transbordo de 5 minutos y se establecen las restricciones del diseño: el número de rutas r y los límites MIN y MAX de longitud. Con este material, se ejecuta SEAMO2, que evalúa cada individuo en las dos funciones objetivo en conflicto (C_P y C_O) y calcula además los indicadores de demanda d_0, d_1, d_2 y d_{un}. Con estos resultados se aproxima el frente de Pareto y se comparan distintas configuraciones de red dentro de un mismo marco de evaluación.

En las siguientes secciones se aplica este procedimiento sobre distintos conjuntos de datos, presentando en cada caso las salidas del algoritmo, los resultados y figuras correspondientes.

4.1. Conexiones de tren de alta velocidad en España

En esta sección se utiliza un conjunto de datos sintético elaborado específicamente para este trabajo a partir de la red de conexiones de tren de alta velocidad en España en 2025, usada como ejemplo en el capítulo anterior en las figuras 3.1 y 3.2. Los datos no representan una red de transporte real, ya que el objetivo para este primer caso es disponer de un conjunto de datos claro y controlado para ejecutar el algoritmo con el esquema de entrada explicado.

En primer lugar, se asigna un identificador para cada una de las 33 ciudades siguiendo un orden alfabético, como se muestra en la tabla 4.1. Las coordenadas geográficas de latitud y longitud fueron extraídas de manera aproximada de Google Maps[1] para poder realizar la representación gráfica.

La siguiente base de datos recoge las conexiones directas habituales entre las ciudades identificadas, las cuales ya se mostraron anteriormente en la figura 3.2. La designación de los tiempos de viaje fue realizada consultando en la página web de Renfe[2] los trayectos más frecuentes entre semana y asignando manual y aproximadamente un tiempo razonable por conexión. La red se trata como no dirigida y simétrica, de modo que estos tiempos funcionan como pesos de las aristas. A continuación se muestra en la figura 4.1 la representación gráfica de la red ponderada con dichos tiempos de viaje.

[1] https://www.google.com/maps

[2] https://www.renfe.com

1	A Coruña	12	Córdoba	23	Oviedo
2	Albacete	13	Girona	24	Palencia
3	Alicante	14	Granada	25	Pontevedra
4	Badajoz	15	Guadalajara	26	Segovia
5	Barcelona	16	Huesca	27	Sevilla
6	Burgos	17	León	28	Tarragona
7	Castellón	18	Lleida	29	Toledo
8	Ciudad Real	19	Madrid	30	Valencia
9	Cuenca	20	Murcia	31	Valladolid
10	Cáceres	21	Málaga	32	Zamora
11	Cádiz	22	Ourense	33	Zaragoza

Tabla 4.1: Identificadores de las ciudades del conjunto de datos.

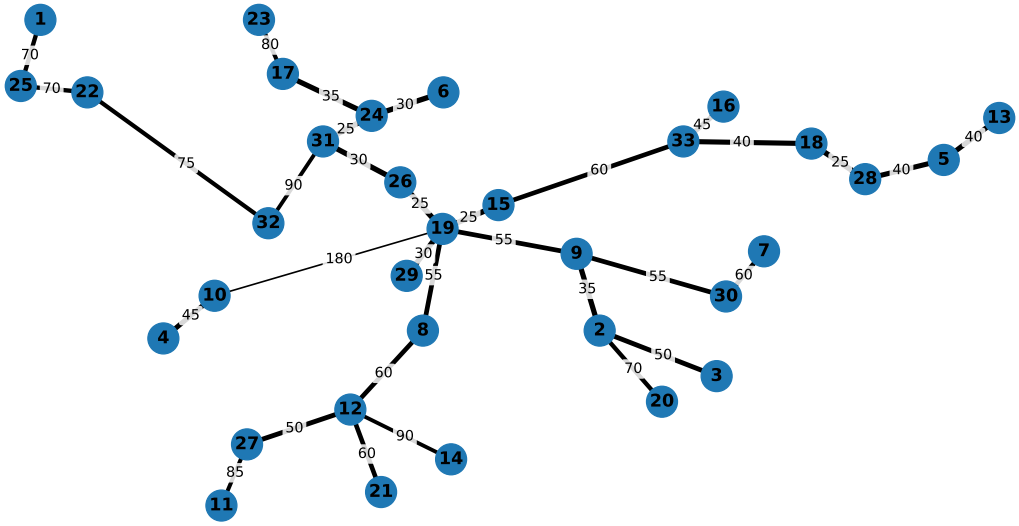


Figura 4.1: Representación gráfica de la red de transporte ponderada.

Por último, los valores de demanda se estimaron creando un escenario razonable basado en una aproximación que tuvo en cuenta el tamaño y la población de las ciudades, así como su conectividad dentro de la red. Se obtuvo así un escenario ilustrativo diseñado para mostrar el comportamiento del algoritmo en esta instancia.

Con este conjunto de datos expuesto, se procede a continuación con la ejecución del algoritmo SEAMO2. La tabla 4.2 muestra los parámetros empleados, que recogen tanto las restricciones impuestas al problema como los valores asociados al funcionamiento del algoritmo. Se fijaron $r = 12$ rutas por individuo, con longitudes comprendidas entre $MIN = 1$ y $MAX = 14$. Los parámetros evolutivos se configuraron con una población de 200 individuos y 1000 generaciones, valores que proporcionan una exploración suficiente del espacio de soluciones sin comprometer la eficiencia computacional. En conjunto, estos ajustes resultan adecuados para una instancia de tamaño reducido, como la considerada, permitiendo alcanzar soluciones estables y representativas del comportamiento del algoritmo.

Parámetro	Valor
Número de rutas (r)	12
Longitud mínima de ruta (MIN)	1
Longitud máxima de ruta (MAX)	14
Penalización por transbordo	5
Tamaño de población (N)	200
Generaciones	1000

Tabla 4.2: Parámetros de la ejecución sobre el conjunto de datos de las conexiones de tren de alta velocidad en España.

Con los parámetros ya establecidos, la ejecución del algoritmo genera tres soluciones no dominadas que muestran un equilibrio adecuado entre los costes del pasajero y del operador, las cuales se denotan por R_A, R_B y R_C. La tabla 4.3 presenta sus correspondientes costes e indicadores de demanda, lo que permite comparar el comportamiento de cada solución bajo las mismas condiciones de ejecución. Como se puede observar, los valores de d_2 y d_{un} son nulos tanto para las soluciones R_B como R_C. Esto indica que toda la demanda se satisface mediante trayectos que requieren, como máximo, un único transbordo, lo que refleja un alto grado de conectividad y eficiencia en la cobertura de la red propuesta por estas soluciones.

	C_P	C_O	d_0	d_1	d_2	d_{un}
R_A	185.70	1980	39.62	39.35	19.59	1.45
R_B	183.43	3535	62.55	37.45	0.00	0.00
R_C	183.38	4135	63.42	36.58	0.00	0.00

Tabla 4.3: Soluciones no dominadas obtenidas tras la ejecución sobre el conjunto de datos de las conexiones de tren de alta velocidad en España.

A partir de las soluciones no dominadas obtenidas, se seleccionaron R_A y R_C por situarse en los extremos del equilibrio entre los dos objetivos considerados. El conjunto de rutas R_C alcanza el menor coste para los pasajeros, priorizando la calidad del servicio a costa de un mayor esfuerzo operativo. En cambio, R_A presenta el menor coste del operador, favoreciendo una red más eficiente y económica aunque con trayectos algo menos directos para los pasajeros. La tabla 4.4 muestra dichas soluciones.

Mejor C_P : Conjunto de rutas $R_C = \{r_{C_1}, r_{C_2}, \ldots r_{C_{12}}\}$.
C_P= 183.38 C_O = 4135 $d_0 = 63.42\,\%$ $d_1 = 36.58\,\%$ $d_2 = 0.00\,\%$ $d_{\text{un}} = 0.00\,\%$ *Rutas*: [7-30-9-19-15-33-16], [1-25-22-32-31-26-19-8-12-27-11], [20-2-9-19-29], [13-5-28-18-33-15-19-26-31-24-17-23], [3-2-20], [1-25-22-32-31-26-19-10-4], [11-27-12-21], [6-24-31-26-19-9-30-7], [3-2-9-19-8-12-14], [13-5-28-18-33-15-19-8-12-21], [16-33-15-19-26-31-24-17-23], [4-10-19-15-33-16].
Mejor C_O : Conjunto de rutas $R_A = \{r_{A_1}, r_{A_2}, \ldots r_{A_{12}}\}$.
$C_P = 185.70$ $C_O = 1980$ $d_0 = 39.62\,\%$ $d_1 = 39.35\,\%$ $d_2 = 19.59\,\%$ $d_{\text{un}} = 1.45\,\%$ *Rutas*: [16-33], [23-17-24-31-26-19-15-33], [3-2-9-19-8-12-14], [12-21], [1-25-22-32-31-26-19-10-4], [2-20], [13-5-28-18-33-15-19], [19-29], [11-27-12], [6-24], [9-30], [7-30-9].

Tabla 4.4: Mejores soluciones para los pasajeros y para el operador.

Con el fin de visualizar mejor los resultados y comprobar la estabilidad del algoritmo, se realizaron cinco ejecuciones independientes utilizando semillas distintas, manteniendo fijos el resto de parámetros. Esto permite obtener un conjunto más amplio de soluciones no dominadas y representar de forma conjunta la relación entre C_P y C_O. Entre ellas, se tomó como primera ejecución la realizada anteriormente para facilitar el análisis detallado, al presentar un comportamiento estable y representativo. La figura 4.2 muestra los resultados de todas las ejecuciones, donde se aprecia una tendencia coherente y una distribución estable de los valores alcanzados en ambos objetivos.

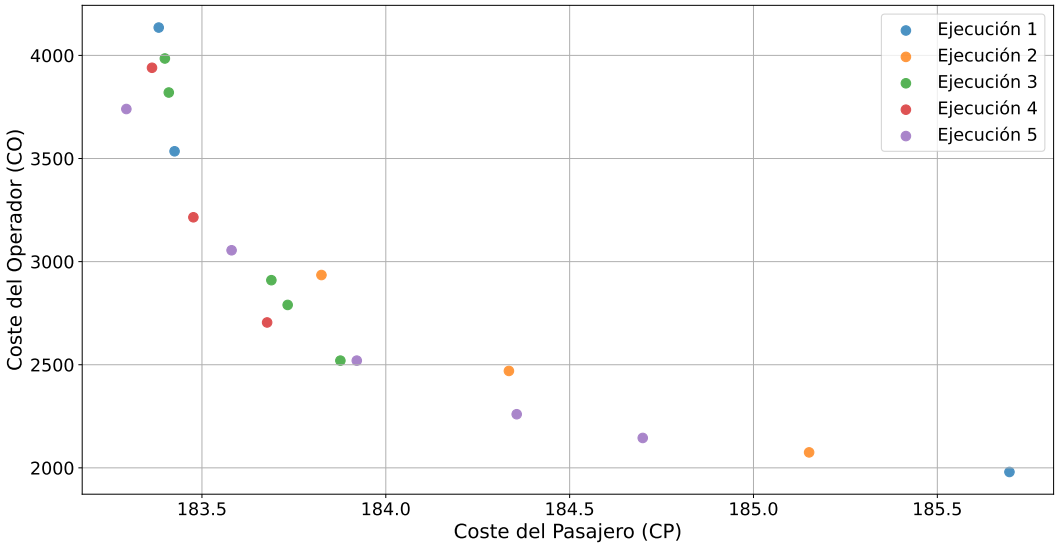


Figura 4.2: Frente de Pareto obtenido con 5 ejecuciones independientes sobre el conjunto de datos de conexiones de tren de alta velocidad en España.

Se puede apreciar como debido a la naturaleza heurística del algoritmo, las soluciones obtenidas en cada ejecución difieren entre sí, variando incluso el número de puntos que conforman el frente de Pareto. Este comportamiento es característico de los métodos evolutivos, que no garantizan la optimalidad global de las soluciones, pudiendo ocurrir que algunas de ellas resulten dominadas por otras obtenidas en ejecuciones distintas. Por este motivo, y siempre que el coste computacional lo permita, puede resultar conveniente realizar varias ejecuciones independientes del algoritmo con el fin de mejorar la exploración del frente de Pareto y aumentar la probabilidad de alcanzar soluciones cercanas al óptimo.

4.2. Red de transporte de Brighton

El segundo conjunto de datos empleado en este trabajo se denomina *Mumford2*, una de las instancias propuestas por Mumford (2013). Se trata de una red de transporte urbano de dimensiones intermedias, basada en un mapa real de la ciudad de Brighton (Reino Unido) en la que los tiempos de viaje y la demanda de pasajeros se consideran simétricos. No obstante, la disposición de sus nodos y enlaces no reproduce la geografía real de la ciudad, sino que fue generada para conservar únicamente la estructura topológica de una red realista.

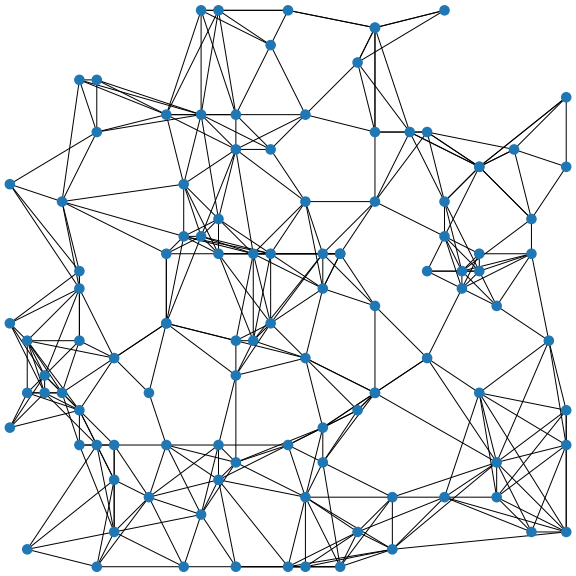


Figura 4.3: Grafo Mumford2.

El conjunto de datos utilizado en esta implementación se ha obtenido de un repositorio de GitHub[3], y se compone de tres conjuntos de datos como los ya explicados, a partir de los cuales se ha construido el grafo representado en la figura 4.3.

Para la ejecución de este conjunto de datos se ajustaron los valores de algunos parámetros debido al mayor tamaño y complejidad de la red. Esta instancia incluye un número considerablemente superior de nodos y aristas, lo que incrementa de forma notable el tiempo de evaluación de cada individuo, especialmente en el cálculo del C_P, que requiere obtener las distancias mínimas entre todos los pares de nodos de la red de tránsito. Para mantener tiempos de ejecución razonables y evitar un coste computacional excesivo, se redujeron el tamaño de la población y el número de generaciones. En cambio, los parámetros estructurales del problema (r, MIN y MAX) se mantuvieron iguales a los utilizados por Mumford (2013). Los parámetros empleados se recogen en la tabla 4.5.

Parámetro	Valor
Número de rutas (r)	56
Longitud mínima de ruta (MIN)	9
Longitud máxima de ruta (MAX)	21
Penalización por transbordo	5
Tamaño de población (N)	4
Generaciones	12

Tabla 4.5: Parámetros de la ejecución sobre el conjunto de datos Mumford2.

[3] https://github.com/RenatoArbex/TransitNetworkDesign

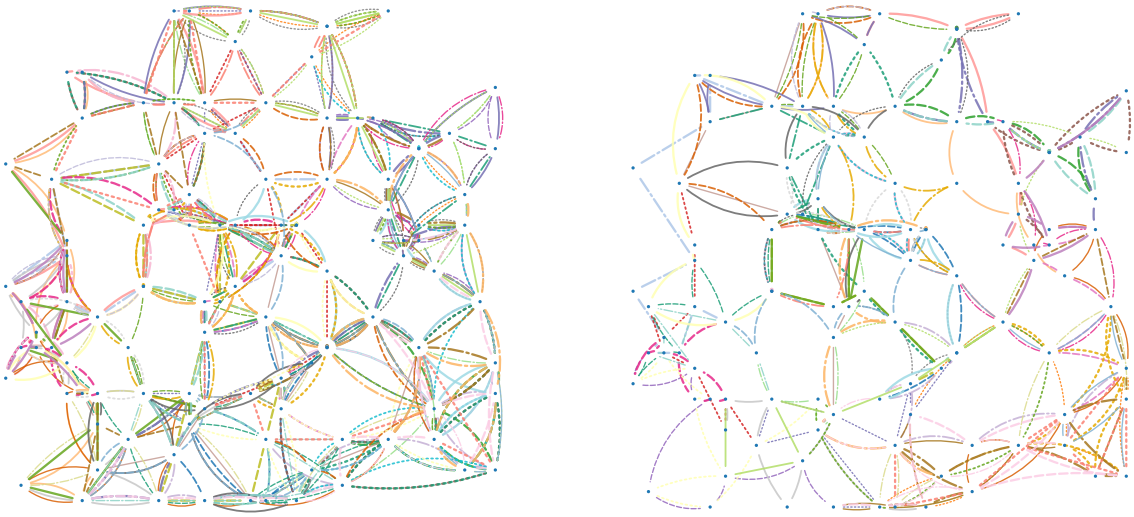
Los resultados obtenidos se muestran en la tabla 4.6, donde cada fila corresponde a un conjunto de rutas generado por el algoritmo, con sus costes e índices de demanda correspondientes.

	C_P	C_O	d_0	d_1	d_2	d_{un}
R_A	29.53	5134	66.22	33.78	0.00	0.00
R_B	32.73	2535	26.77	59.29	13.77	0.17
R_C	33.31	2454	24.71	60.67	14.46	0.16
R_D	29.68	4999	65.79	34.21	0.00	0.00

Tabla 4.6: Soluciones no dominadas obtenidas tras la ejecución sobre el conjunto de datos Mumford2.

De entre las cuatro soluciones no dominadas obtenidas, se toman como referencia R_A y R_C, que presentan los valores más bajos en C_P y en C_O respectivamente. Aunque la representación gráfica de estas soluciones no resulta especialmente útil desde un punto de vista analítico debido al gran tamaño de la red y al elevado número de rutas implicadas, se ha considerado interesante realizarla con Python.

En la figura 4.4 se muestran los conjuntos de rutas R_A y R_C, compartiendo la leyenda de la figura 4.5, donde $X = \{A, C\}$. A pesar de su complejidad visual, estos grafos permiten apreciar de forma intuitiva la cobertura de la red y la estructura general de ambas soluciones, ofreciendo una perspectiva complementaria a los resultados numéricos presentados en la tabla anterior.



R_A ($C_P = 29.53$; $C_O = 5134$) R_C ($C_P = 33.31$; $C_O = 2454$)

Figura 4.4: Representación gráfica de las soluciones no dominadas R_A y R_C.

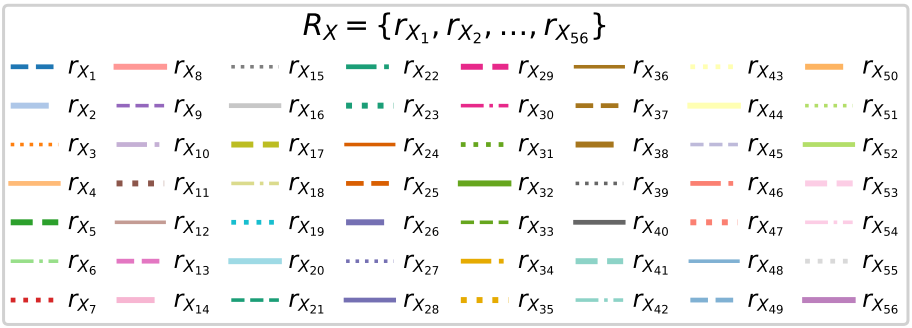


Figura 4.5: Leyenda para las rutas de las soluciones no dominadas R_A y R_C.

Se puede observar que la densidad de las aristas disminuye en la solución R_C, asociada al menor coste para el operador, mientras que la solución R_A, que minimiza el coste para el pasajero, presenta una mayor cobertura y una red más densa. Esta diferencia refleja el

compromiso entre ambos objetivos: reducir el esfuerzo operativo simplifica la red, mientras que mejorar la calidad del servicio implica un mayor coste para el operador.

Para examinar la consistencia del algoritmo en este conjunto de datos, se realizaron varias ejecuciones independientes, manteniendo fijos los demás parámetros de la tabla 4.5, siendo las soluciones obtenidas en la primera de ellas las que se presentan en la tabla 4.6. De este modo se obtuvo un conjunto más amplio de soluciones no dominadas, cuya distribución conjunta se representa en la figura 4.6.

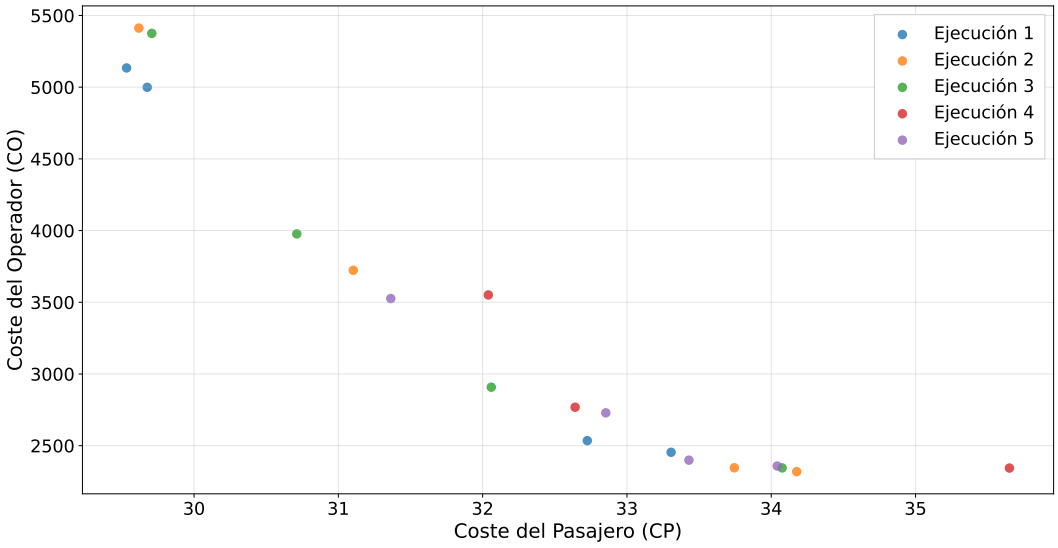


Figura 4.6: Frente de Pareto obtenido con 5 ejecuciones independientes sobre el conjunto de datos Mumford2.

Se puede apreciar una tendencia coherente y una distribución estable de las soluciones no dominadas, que reflejan un equilibrio entre C_P y C_O. El frente de Pareto presenta una forma suavemente decreciente, evidenciando el compromiso característico entre los objetivos. En conjunto, los resultados confirman la estabilidad del algoritmo.

Capítulo 5

Conclusiones

El presente trabajo aborda el problema de rutas de transporte urbano desde un enfoque formal basado en la teoría de grafos y en la optimización multiobjetivo. A partir de la formulación del modelo y de la implementación de una versión adaptada del algoritmo evolutivo SEAMO2, se estudia el comportamiento de la metodología propuesta sobre diferentes conjuntos de datos. El desarrollo integra el planteamiento teórico, la programación del algoritmo y el análisis de los resultados alcanzados, con el objetivo de evaluar su utilidad en el contexto del URTP.

Los resultados obtenidos muestran que el enfoque evolutivo propuesto permite alcanzar soluciones viables y coherentes con la estructura del problema, equilibrando los dos objetivos considerados: el coste del pasajero y el coste del operador.

En el caso de la red de conexiones de tren de alta velocidad en España en 2025, el frente de Pareto muestra una distribución amplia y una cobertura pronunciada, que re-

fleja la existencia de múltiples configuraciones eficientes y una relación no lineal entre los dos objetivos. Se aprecia que, a medida que disminuye el coste del pasajero, el coste del operador aumenta de manera más notable, lo que refleja un margen de compensación relativamente amplio entre la calidad del servicio y el esfuerzo operativo necesario para mantenerla. Además, las distintas ejecuciones independientes producen frentes coherentes entre sí, lo que confirma la estabilidad del algoritmo y su capacidad para mantener diversidad en la población de soluciones no dominadas.

En la instancia de mayor tamaño correspondiente a la red de Brighton, el frente de Pareto presenta una estructura un poco más alineada y compacta. Este comportamiento indica una correlación más fuerte entre los objetivos, de modo que las mejoras en el coste del pasajero implican incrementos prácticamente proporcionales en el coste del operador. Esta relación puede atribuirse tanto a la naturaleza del problema, pues es más complejo y presenta un número elevado de nodos y aristas, como a las condiciones experimentales utilizadas. Debido al alto coste computacional de la instancia, se empleó un tamaño de población y un número de generaciones reducidos, lo que limita la exploración del espacio de búsqueda y la diversidad de soluciones obtenidas. Aun así, el resultado no indica un peor rendimiento del algoritmo, sino una convergencia más concentrada hacia una zona estable del frente de Pareto, propia de redes de gran tamaño con objetivos estrechamente relacionados.

Desde el punto de vista computacional, el procedimiento de evaluación constituye el componente más costoso del algoritmo, al requerir el cálculo de las distancias mínimas entre todos los pares de nodos de la red de tránsito. La implementación usada optimiza este proceso mediante una única ejecución del algoritmo de Floyd–Warshall por individuo, lo que reduce significativamente el tiempo de cómputo sin afectar a la calidad de las

soluciones. Aun así, se considera que la eficiencia del algoritmo puede mejorarse mediante un uso más adecuado de los recursos computacionales en problemas de gran escala.

El trabajo pone de manifiesto que los algoritmos evolutivos constituyen una herramienta eficaz para abordar el diseño de redes de transporte urbano, especialmente en problemas de naturaleza combinatoria donde los métodos exactos se quedan cortos. Su capacidad para mantener diversidad poblacional y equilibrar objetivos en conflicto permite aproximar de forma eficiente el frente de Pareto. No obstante, la aplicación práctica de estos algoritmos está condicionada por la disponibilidad de conjuntos de datos adecuados. La construcción de matrices de demanda y tiempos de viaje fiables continúa siendo un desafío, al igual que la adaptación de los modelos a contextos reales con variaciones temporales o direccionales en la demanda.

Entre futuras líneas de investigación, se considera relevante incorporar información temporal sobre la demanda, integrar la frecuencia de los servicios y extender el modelo a redes dirigidas con tiempos asimétricos. Asimismo, la combinación de estrategias evolutivas con métodos de búsqueda local o técnicas de aprendizaje automático podría aumentar la capacidad de convergencia hacia soluciones más eficientes. Por último, la exploración de configuraciones algorítmicas con mayor eficiencia computacional permitiría aplicar el modelo a redes urbanas de gran escala, manteniendo la precisión en la evaluación de los costes.

En conjunto, el trabajo demuestra que el enfoque evolutivo aplicado al diseño de redes de transporte urbano resulta eficaz y versátil. El algoritmo implementado permite obtener soluciones equilibradas entre los objetivos del operador y del pasajero, incluso en problemas más complejos. Además, el método no solo genera un conjunto de soluciones eficientes, sino que proporciona al planificador un frente de Pareto que facilita una toma

de decisiones informada y estratégica, al mostrar de forma explícita el compromiso entre la eficiencia operativa y la calidad del servicio. A pesar de las limitaciones de los conjuntos de datos, los resultados confirman que los algoritmos evolutivos constituyen una herramienta válida para la planificación y mejora de sistemas de transporte urbano.

Bibliografía

Antón, D. (2025). Optimización de rutas de transporte urbano. https://www.routingmaps.com/optimizacion-rutas-transporte/. Publicado en RoutingMaps. Último acceso: 7 de agosto de 2025.

Arranz de la Peña, J. and Parra Truyol, A. (2007). Algoritmos genéticos. *Universidad Carlos III*, pages 1–8.

Bagloee, S. A. and Ceder, A. A. (2011). Transit-network design methodology for actual-size road networks. *Transportation Research Part B: Methodological*, 45(10):1787–1804.

Barrero, A. C., de García, G. W., and Parra, R. M. M. (2010). *Introducción a la Teoría de Grafos*. Elizcom SAS.

Bellman, R. (1958). On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308.

Ceder, A. and Wilson, N. H. (1986). Bus network design. *Transportation Research Part B: Methodological*, 20(4):331–344.

Coello, C. A. C. (2010). A tutorial on multi-objective optimization using metaheuristics. *Monografías Matemáticas "García de galdeano"*, 36:19–38.

Cunquero, R. M. (2003). Algoritmos heurísticos en optimización combinatoria. *Valencia: Universidad de Valencia. Retrieved*, 11(01):2012.

Deb, K. (2001). *Multi-objective Optimization Using Evolutionary Algorithms*. Wiley.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197.

Diestel, R. (2025). *Graph theory*, volume 173. Springer Nature.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 269:271.

Floyd, R. W. (1962). Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345–345.

Ford Jr, L. R. (1956). Network flow theory. Technical report.

Gross, J. L., Yellen, J., and Anderson, M. (2018). *Graph theory and its applications*. Chapman and Hall/CRC.

Kocay, W. and Kreher, D. L. (2016). *Graphs, algorithms, and optimization*. Chapman and Hall/CRC.

Kourepinis, V., Iliopoulou, C., Tassopoulos, I. X., Aroniadi, C., and Beligiannis, G. N. (2023). An improved particle swarm optimization algorithm for the urban transit routing problem. *Electronics*, 12(15):3358.

Mandl, C. E. (1980). Evaluation and optimization of urban public transportation networks. *European Journal of Operational Research*, 5(6):396–404.

Mumford, C. L. (2004). Simple population replacement strategies for a steady-state multi-objective evolutionary algorithm. In *Genetic and Evolutionary Computation Conference*, pages 1389–1400. Springer.

Mumford, C. L. (2013). New heuristic and evolutionary operators for the multi-objective urban transit routing problem. In *2013 IEEE congress on evolutionary computation*, pages 939–946. IEEE.

Ortigosa, P. P. (2022). Teoría de grafos. In *Desarrollo de destrezas en resolución de problemas de olimpiadas matemáticas*, pages 65–84. Universidad de La Rioja.

Pérez, J. H. and Rückauer, C. C. (2004). Una revisión de los algoritmos evolutivos y sus aplicaciones. *Enlaces: revista del CES Felipe II*.

Shih, M.-C. and Mahmassani, H. S. (1994). A design methodology for bus transit networks with coordinated operations. *Center for Transportation Research, The University of Texas at Austin*. Technical Report SWUTC/94/60016-1.

Tomar, V., Bansal, M., and Singh, P. (2024). Metaheuristic algorithms for optimization: A brief review. *Engineering Proceedings*, 59(1):238.

Torres Díaz, N. et al. (2022). Desarrollo de algoritmos dirigidos por retos: Optimización de un modelo de simulación para la planificación de la capacidad y recursos de hospitales durante la pandemia de COVID-19.

Valenzuela, C. L. (2002). A simple evolutionary algorithm for multi-objective optimization (SEAMO). In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 1, pages 717–722. IEEE.

Von Lücken, C., Hermosilla, A., and Barán, B. (2004). Algoritmos evolutivos para optimización multiobjetivo: Un estudio comparativo en un ambiente paralelo asíncrono. In *X Congreso Argentino de Ciencias de la Computación*.

Warshall, S. (1962). A theorem on boolean matrices. *Journal of the ACM (JACM)*, 9(1):11–12.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the strength pareto evolutionary algorithm. *TIK report*, 103.