FACULTAD DE MATEMÁTICAS

DEPARTAMENTO: ESTADÍSTICA E INVESTIGACIÓN OPERATIVA

Trabajo Fin de Grado:

Problemas de rutas de vehículos por arcos

Autor:
María Calvo González

Dirigido por:
Justo Puerto Albandoz

2017 - 2018

Abstract

At the beginning of this work, we are going to give a historical introduction of arc routing. We will take a look at the complexity of this type of problems, and then, we will focus on the Chinese Postman Problem. It is arguably the most central problem in this area. Given a graph, it basically tries to find a minimun cost tour traversing at least once each edge. We will study the undirected, directed, mixed and windy version in detail, and their respective ways to deal with them. Finally, we will see some applications about real instances for each version.

Propósito general

Comenzaremos nuestro trabajo hablando de los inicios del estudio de las rutas por arcos con el problema de los puentes de Königsberg. Comentaremos otros problemas históricos relacionados, como el trazado de figuras sin levantar el lápiz del papel o la resolución de laberintos. Esto nos conducirá a la necesidad del nacimiento de la optimización, y haremos un planteamiento previo del Problema del Cartero Chino, en el que se pretende encontrar una ruta de coste mínimo que recorra todas las aristas o arcos del grafo al menos una vez. Se plantean además ciertos resultados teóricos que serán necesarios posteriormente. Concluiremos nuestra introducción comentando la complejidad de los problemas que vamos a estudiar, así como la de otras variantes no tratadas en este trabajo.

En el segundo capítulo plantearemos el Problema del Cartero Chino no dirigido, caso en el que tenemos un grafo no dirigido. La resolución de este se basa en buscar un grafo de aumento que sea Euleriano, y una vez hallado, encontrar un circuito Euleriano en él. Veremos como calcular este aumento tanto desde el punto de vista de la programación dinámica como de la programación matemática.

En el tercer capítulo daremos el planteamiento del PCC para cuando tenemos grafos dirigidos, mixtos y windy. Veremos así en primer lugar el Problema del Cartero Chino dirigido, y su correspondiente formulación. A continuación, estudiaremos el Problema del Cartero Chino mixto, así como varias formas de abordarlo, tanto cuando existe solución exacta como cuando no, y dos formulaciones de este (además de una comparación de ambas). Finalmente, nos centraremos en el Pro-

blema del Cartero windy, es decir, aquel en el que el coste de cada arista depende de la dirección en la que la recorramos. Al igual que en el caso mixto, plantearemos varios algoritmos para su resolución exacta (si es posible) o aproximada (si no), así como su formulación correspondiente.

Concluiremos nuestro trabajo con un capítulo de aplicaciones de las cuatro variantes estudiadas para casos reales. Para ello utilizaremos un programa en Xpress que nos proporcionará la solución óptima para cada caso.

Índice general

1 Introducción

A modo introductorio, haremos un breve recorrido histórico desde el inicio del estudio de las rutas por arcos hasta la aparición de la optimización. Además, proporcionaremos ciertos resultados teóricos que serán necesarios posteriormente. En la tercera sección, presentaremos esquemáticamente los problemas en los que se va a profundizar en este trabajo, así como otras variantes. Estudiaremos la complejidad para cada uno de ellos y concluiremos comentando la situación de los problemas de rutas por arcos en la actualidad.

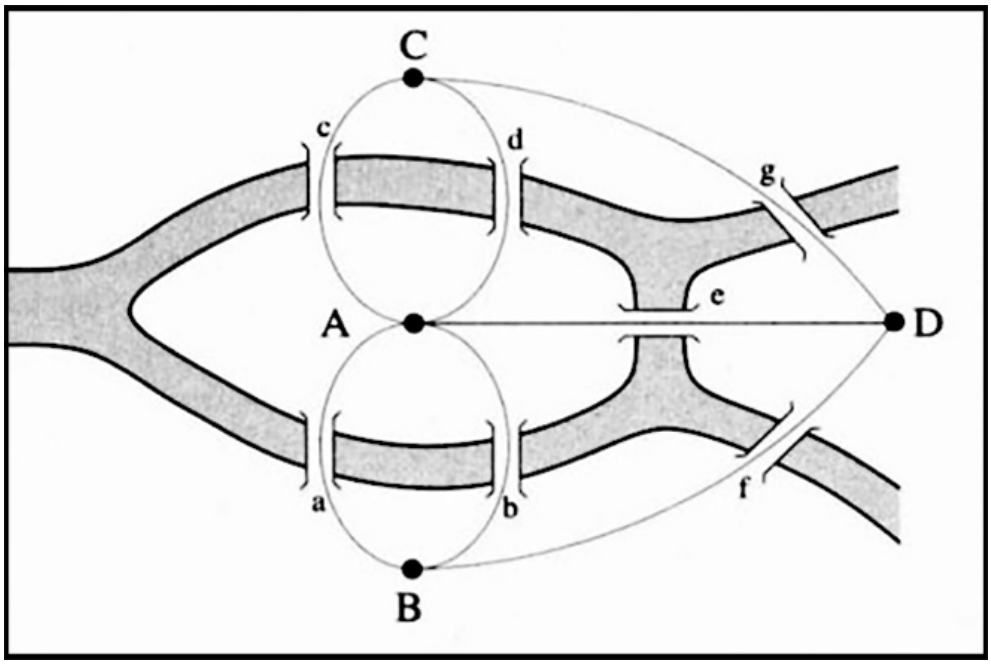
1.1. Notas históricas

El estudio de las rutas por arcos se remonta a tiempos en que el matemático Leonhard Euler fue llamado a estudiar el famoso problema de los puentes de Könisberg, allá por el siglo XVIII. Sin embargo, aunque algunos matemáticos pusieron interés sobre estos problemas y aportaron comentarios interesantes, no fue hasta 1960 con la primera publicación del Problema del Cartero Chino cuando nació este área realmente.

El problema de los puentes de Königsberg

Königsberg es una ciudad en Prusia, atravesada por un río que se bifurca para rodear con sus brazos a la isla Kneiphof. El terreno se divide así en cuatro regiones distintas, las que entonces estaban unidas mediante siete puentes. El problema

consistía en encontrar un recorrido para cruzar a pie toda la ciudad pasando sólo una vez por cada uno de los puentes.



A partir de esta situación, Euler [19] formuló una versión generalizada del problema: "Sea cual sea la disposición, la división del río y el número de puentes que hay, ¿puede alguien determinar si es posible o no cruzar todos los puentes exactamente una vez?". Sin embargo, no estaba claro si la ruta debía ser abierta o cerrada.

La solución que propuso a su problema fue: "Si hay más de dos zonas en las que desemboca un número impar de puentes, no hay solución. Si esto ocurre exactamente en dos zonas, hay solución si empezamos en alguna de estas dos zonas. Si no hay zonas con esta característica, hay solución empezando desde cualquier sitio.

Como la versión de la ruta cerrada está muy relacionada con los grafos eulerianos, bastaría preguntarse si el grafo que representa el acertijo es euleriano. Sabiendo que un vértice de un grafo se dice que tiene grado o valencia impar si el número de aristas y arcos incidentes en él es impar, si los cuatro vértices del grafo representan las áreas que tienen valencia impar, la respuesta es no: no hay ruta abierta o cerrada que cruce exactamente una vez cada puente de Königsberg.

Otros estudios históricos relacionados

Un problema relacionado es el dibujo de figuras dadas con el mínimo número de trazos. En 1809, Poinsot [42] usó argumentos similares a los de Euler para probar que los grafos que son completos con n vértices pueden dibujarse enteros sin levantar el lapiz del papel si n es impar, pero no si n es par. Esto es debido a que los grafos completos son eulerianos si tienen un número impar de vértices. Unos años despúes, Listing [37] aportó la idea clave para resolver lo que posteriormente se conocería como el Problema del Cartero Chino: unir los vértices del grafo con valencia impar.

Otro problema relacionado es el de escapar de un laberinto, en el cual se tuvo un especial interés al final del siglo XIX. Uno de los que hicieron aportaciones importantes fue König [34]: "Un laberinto puede ser identificado con un grafo; los vértices del grafo corresponden con los puntos del laberinto en los que hay bifurcaciones en el camino, y los nodos finales con los callejones sin salida. Se requiere de un método para llegar a un punto específico del laberinto, el cual se considera usualmente como el centro de este". Hubo tres matemáticos que indagaron al respecto para encontrar el método: Wiener, Tremáux y Tarry, cada uno de ellos con hipótesis y soluciones distintas. Se concluyó finalmente con que un laberinto puede ser recorrido pasando como máximo dos veces por cada una de sus calles, una vez en cada sentido.

El nacimiento de la optimización

Fue Meigu Guan, un matemático chino, quien en 1960 introdujo lo que hoy conocemos como el Problema del Cartero Chino (PCC), mediante la siguiente formulación [30]: "Un cartero tiene que cubrir una zona que se le ha asignado, antes de volver a la oficina. Se busca encontrar la distancia más corta para el cartero". El primero en darle nombre fue Jack Edmonds [17], aunque quizá hubiera sido previamente mencionado por su compañero de investigación Alan Goldman.

El PCC puede ser interpretado como el problema de encontrar un subconjunto de aristas con la mínima longitud que, en el grafo original, produce un grafo euleriano. Guan [30] señaló que un grafo siempre tiene un número par de vértices de grado

impar, y que un grafo euleriano puede obtenerse replicando algunas aristas con el objetivo de conectar los vértices de grado impar. A partir de esto, Guan propuso un algoritmo para el PCC, pero era no polinomial.

El problema generalizado de los puentes de Königsberg también lo es [35], en el que se preguntan por los caminos que cruzan cada arista al menos una vez y para el cual el número de repeticiones de aristas es mínimo.

La aportación de Edmonds [17] al PCC fue un algoritmo que combinaba otros dos: el algoritmo del camino más corto, y el del máximo emparejamiento [31]. Consistía en construir un grafo completo G' cuyos vértices son los de grado impar del grafo inicial G, y cuyos costes de las artistas son los de los caminos más cortos que los unen en G, computando la asignación de mínimo coste en G'. La versión dirigida del PCC es polinomial, mientras que la definida en un grafo mixto (PCCM) es NP-dura. Estudiaremos la complejidad de cada versión con más detenimiento posteriormente.

1.2. Resultados teóricos previos

Será importante fijar de forma previa algunos resultados teóricos acerca de la caracterización de grafos eulerianos, necesarios para que el trabajo sea autocontenido. Supondremos que los grafos que consideraremos son conexos si son no dirigidos, y fuertemente conexos si son dirigidos o mixtos.

- **Grafos eulerianos no dirigidos**

  Euler afirmó (y fue posteriormente probado por Hierholzer [32]):

  Teorema 1. *Un grafo conexo no dirigido G es euleriano si y solo si cada vértice de G tiene grado par.*

  Una segunda caracterización de grafos eulerianos fue la dada por Veblen [46] en 1912:

  Teorema 2. *Un grafo conexo no dirigido G es euleriano si y solo si es la unión disjunta de ciclos.*

- **Grafos eulerianos dirigidos**

  La paridad de los vértices de un grafo dirigido G=(V,A) es de nuevo una condición necesaria para que el grafo sea euleriano, pero no suficiente. La condición suficiente fue dada por König [34] :

  Teorema 3. *Un grafo fuertemente conexo y dirigido G es euleriano si y solo si cada vértice de G es simétrico,*

  donde simétrico significa que el número de arcos entrantes y salientes en cada vértice es igual.

- **Grafos eulerianos mixtos**

  La paridad de los vértices también es una condición necesaria pero no suficiente para que un grafo mixto G=(V,E,A) sea euleriano. Además, esta característica junto con la simetría son condiciones suficientes para la "unicursalidad". Sin embargo, para el caso de grafos mixtos, existe una condición más que debe cumplirse. Esta fue planteada por Ford y Fulkerson [24]:

  Teorema 4. *Un grafo mixto y fuertemente conexo G es euleriano si y solo si G es par y equilibrado,*

  donde equilibrado significa que para cualquier subconjunto de vértices S $\subseteq$ V no vacío, el valor absoluto de la diferencia entre el número de arcos de S hasta V\S y el número de arcos de V\S hasta S debe ser menor o igual al número de aristas entre V\S y S. No obstante, es evidente la complejidad de comprobar si un grafo es o no equilibrado.

1.3. La complejidad de los problemas de rutas por arcos

En esta sección vamos a tratar la complejidad de dichos problemas en el ámbito de las rutas por arcos. Veremos así si su solución requiere de una cantidad significativa de recursos computacionales, sin importar el algoritmo utilizado para ello.

Este tema es de gran importancia en la algorítmica, ya que nos permite saber si podemos trasladar a la práctica los diversos algoritmos que se proponen para la resolución de problemas.

Comentaremos la complejidad de nuestro Problema del Cartero Chino, de las variantes de éste que estudiaremos posteriormente y de otras extensiones, el Problema del Cartero Rural y el Problema de Rutas por Arcos con Capacidades (no estudiados en profundidad en este trabajo), para los que compararemos su complejidad con nuestro problema original.

El PCC busca la ruta de coste mínimo atravesando todas las aristas del grafo al menos una vez. Sin embargo, existen dos extensiones a las que haremos referencia en esta sección brevemente:

- *El Cartero Rural:* Es una generalización del PCC pues solo un subconjunto de aristas tienen que ser visitadas.
- *Rutas por Arcos con Capacidades:* Es el problema de rutas por arcos más general, pues permite usar más de un vehículo para moverse por las aristas.

A excepción de la versión más básica del PCC, todos son NP-duros, es decir, son computacionalmente intratables. La forma de abordar este inconveniente es:

- Aproximaciones mediante algoritmos de tiempo polinomial [1, 45, 47], donde se cambia optimalidad en la solución por eficiencia, y la ejecución en tiempo exponencial por polinomial.

- Algoritmos parametrizados y análisis de complejidad [13, 23, 40], donde se busca identificar unos pocos parámetros específicos del problema que influyen en el comportamiento exponencial.

Para estudiar los problemas de complejidad, consideraremos dos tipos de problemas:

* **Problemas de optimización** Q. Consisten en una función que, dadas algunas peticiones, devuelve una solución factible que minimiza (o maximiza) una medida m_Q. Q puede resolverse en un tiempo T(n) si hay un algoritmo que, dado un caso x, computa una solución factible Y para x en T(n) pasos computacionales, siendo $m_Q(Y)$ mínimo o máximo.

* **Problemas de decisión** Q. Consisten en un conjunto de casos que cumplen alguna propiedad. Decimos que Q puede resolverse en un tiempo T(n) si hay algún algoritmo que, dado un caso x de longitud $|x| = n$ bits, determina si $x \in Q$ usando al menos T(n) pasos computacionales. Las dos clases

más importantes de problemas de decisión son P (donde todos los problemas pueden resolverse por una máquina de Turing determinista en un tiempo n^c para alguna constante $c \in \mathbb{N}$) y NP (donde la Máquina de Turing sería no determinista).

- **Algoritmos de tiempo polinomial**

  Consideramos que un problema de decisión A puede reducirse (en un tiempo polinomial) a un problema de decisión B si hay algún algoritmo ejecutable en tiempo polinomial y, que dado un caso x de A, produce un caso y de B tal que $x \in A \longleftrightarrow y \in B$. De esta manera, decimos que un problema es NP-duro si todos los problemas en NP pueden reducirse a él. Si Q está también contenido en NP, entonces se llama NP-completo.

  Muchos problemas de la vida real han resultado ser NP-duros y por tanto, no admiten algoritmos en tiempo polinomial. Como es posible usar algoritmos que resuelvan problemas de optimización para solventar problemas de decisión, esto también implica que hay algoritmos en tiempo no polinomial para problemas de minimización o maximización correspondientes a problemas de decisión NP-duros. Una posibilidad sería utilizar algoritmos de aproximación, es decir, aceptar una solución factible razonablemente cerca de la óptima, aunque no necesariamente óptima. Otra incluso sería dejando algo a un lado los problemas que conlleva considerar qué ocurre en el peor de los casos y centrándose más en los tiempos esperados de ejecución [1, 45, 47] .

- **Algoritmos parametrizados**

  Los algoritmos parametrizados [13, 23, 40] son una manera de abordar el asunto de encontrar soluciones óptimas para problemas NP-duros en un tiempo razonable. La idea es acotar la explosión combinatoria con un aspecto del problema: el parámetro.

  Sea Q un problema de decisión. Si hay un algoritmo A tal que, $\forall p \in \mathbb{N}$ y para cada x de Q cuyo parámetro sea p, con A decidiendo si $x \in Q$ en un tiempo polinomial, entonces decimos que Q es resoluble en un tiempo polinomial para valores constantes del parámetro. La clase de estos problemas se denomina XP, para los cuales los algoritmos se ejecutan en tiempo $n^{f(p)}$, siendo p un parámetro y f una función computable. Además, decimos que un problema

de decisión Q se denomina Tratable mediante Parámetro Fijado (TPF) con respecto a un parámetro p si hay algún algoritmo que lo resuelva para cada x de Q en tiempo $f(k)\cdot|x|^c$ para alguna función computable f y una constante c. La diferencia entre XP y TPF es que, aunque ambos son resolubles en tiempo polinomial para valores de parámetros constantes, en XP el grado del polinomio correspondiente depende del parámetro, mientras que en TPF no.

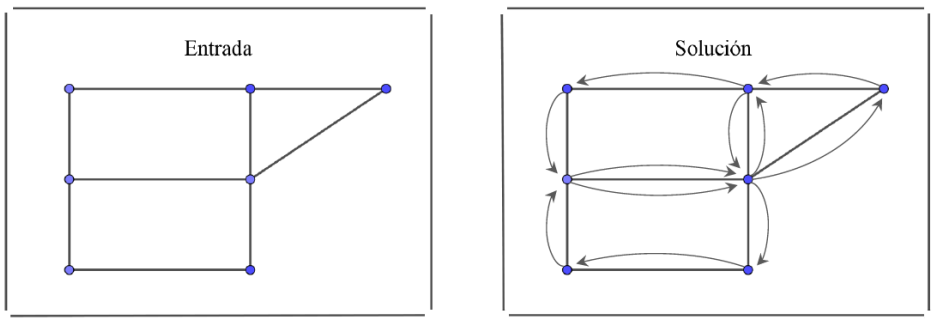
Una propiedad muy importante en la parametrización es encontrar la manera de medir la efectividad del preprocesamiento. La kernelización (o problema Kernel) de un problema de decisión Q es un algoritmo en tiempo polinomial que, para cada x con parámetro p de Q, computa un x' con parámetro p' de Q tal que, para algunas $f, g : \mathbb{N} \longrightarrow \mathbb{N}$ se tiene que:

- $x \in Q \Longleftrightarrow x' \in Q$
- $|x'| \leq g(p)$
- $p' \leq f(p)$

La función g se llama tamaño del problema Kernel, y constituye una medida de la efectividad del algoritmo de kernelización.

1.3.1. Complejidad del Problema del Cartero Chino

Nos centraremos ahora en el Problema del Cartero Chino, donde su estructura combinatoria muestra su carácter NP-duro. Veamos una representación gráfica de un ejemplo de PCC, donde el coste de las aristas es proporcional a su longitud:



En la siguiente tabla [35] podemos observar un breve resumen de la complejidad del PCC:

Variante del PCC	Complejidad Clásica
No dirigido	Algoritmo en tiempo $O(\|V\|^3)$
Dirigido	Algoritmo en tiempo $O(\|V\|^3)$
	Algoritmo en tiempo $O((\|E\| - \|V\|)\|V\|^2)$
Mixto	NP-completo
	Resoluble en tiempo $O(\|V\|^3)$ si cada vértice tiene grado par
Windy	NP-completo
	en P para algunos casos especiales
k-Jerarquico	NP-completo
	Resoluble en tiempo $O(k\|V\|^4)$ si hay precedencia de relación linear
Min-Sum-k-PCC	Resoluble en tiempo $O(\|V\|^3)$ con vértice "oficina de correos"
	En otro caso es NP-completo
Min-Max-k-PCC	NP-completo
Variante del PCC	**Aproximación**
Mixto	Factor 3/2 en tiempo $O(max\{\|V\|^3, \|A\|(max\{\|A\|, \|E\|\})^2\})$
Windy	Factor 3/2
Min-Max-k-PCC	Factor $(2 - 1/k)$ en tiempo $O(\|V\|^3)$
Variante del PCC	**Complejidad parametrizada**
Mixto	Algoritmo en tiempo $O(2^{\|E} \cdot \|V\|^3)$
	en TPF con respecto a $\|A\|$
	en XP con respecto al treewidth
Min-Sum-k-PCC	en TPF con respecto a k sin vértice "oficina de correos"

- **El Problema del Cartero Chino clásico**

  Como podemos observar en el gráfico anterior, el PCC puede presentarse a partir de:

  - **Entrada:** un grafo no dirigido y conexo $G = (V, E)$ con aristas con coste $c(e) \geq 0$ para cada $e \in E$ y con un coste máximo de c_{max}.
  - **Objetivo:** Encontrar una ruta que pase por cada arista en E al menos una vez y con un coste de, a lo sumo, c_{max}.

  Acerca del carácter tratable del problema, Edmonds y Johnson [18] probaron que el PCC es resoluble en tiempo polinomial. Todo se reduce a encontrar un supergrafo Euleriano G' tal que el coste de las aristas en G' no exceda c_{max}. Por hipótesis, G es conexo, por lo que solo hay que añadir aristas para que

cada vértice tenga grado par. Esto puede lograrse resolviendo un problema equivalente de emparejamiento perfecto de mínimo coste en un grafo auxiliar G^*, con aristas con costes.

Es posible transformar un emparejamiento en G^* en un supergrafo Euleriano de G, y viceversa. Como para construir G^* podemos usar un algoritmo ejecutable en tiempo $O(|V|^3)$, el PCC es resoluble en dicho tiempo, y está en P.

Aunque no estamos encontrando una ruta del coste deseado, con los resultados anteriores tenemos asegurada su existencia en un supergrafo Euleriano, decidiendo así el PPC. Para el caso de los grafos dirigidos sería análogo.

- **Grafos mixtos**

  El PCC mixto puede presentarse de la siguiente forma:

  - **Entrada:** un grafo mixto y fuertemente conexo $G = (V, E, A)$ con aristas con coste $c(e) \geq 0$ para cada $e \in E \cup A$ y con un coste máximo de c_{max}.
  - **Objetivo:** Encontrar una ruta (dirigida) que pase por cada arista en E y cada arco en A al menos una vez y con un coste de, a lo sumo, c_{max}.

  Si asumimos que $P \neq NP$, entonces el PCC mixto no es resoluble en tiempo polinomial. Sobre su dureza, diremos que su principal dificultad es que, para resolverlo, tenemos que orientar las aristas y añadir algunos arcos más para obtener un grafo Euleriano. Es decir, para hacer que todos los vértices sean equilibrados. Además:

  Teorema 5. *El Problema del Cartero Chino mixto es NP-completo, incluso si el grafo de entrada es planar, cada vértice tiene a lo sumo grado tres, y cada arista y arco tiene coste uno.*

  Acerca del carácter tratable del problema decir que, si cada vértice tiene un número par de aristas y arcos incidentes, el PCC mixto es resoluble en tiempo polinomial por reducción a un problema de flujo en redes. Por otro

lado, cuando el número de aristas (no dirigidas) en el grafo de entrada es pequeño, el PCC mixto puede resolverse eficientemente. Fernandes et al. [21] demostró posteriormente que el PCC mixto está en XP con respecto al parámetro "treewidth (mínima anchura entre todas las posibles descomposiciones) del grafo de entrada", es decir, que es resoluble en tiempo polinomial si la treewidth es constante. Existen algunos casos tratables más en los que se obtiene la resolución en tiempo polinomial transformando el PCC mixto en un PCC windy.

Finalmente, nos planteamos cómo podemos obtener una solución aproximada usando solo tiempo polinomial. La primera aproximación que surgió fue la de factor 2 [18]. Su origen está basado en que, siendo $\tilde{G}$ el multigrafo resultante de duplicar cada arista y arco del grafo de entrada G, el algoritmo usado para la aproximación me proporciona una solución de coste como máximo el de $\tilde{G}$, que es al menos dos veces el óptimo para G.

Posteriormente se propuso una estrategia mixta de factor 5/3 [25], que se mejoró con una de factor 3/2, sugerida por Raghavachari y Veerasamy [43]. Zaragoza Martínez [52, 51] probó además que no es posible aproximar el coste del *deadheading* con algún factor constante, donde deadheading significa atravesar los arcos y aristas más de una vez. Este resultado se obtuvo al demostrar la NP-dureza de decidir si un grafo mixto dado tiene una ruta que pase exactamente una vez por cada arista.

- **Costes windy**

  Otra variante del PCC consiste en considerar costes asimétricos, es decir, distintos costes de una arista dependiendo de en que sentido se recorra. Así, el PCC windy se puede formular de la siguiente manera:

  - **Entrada:** un grafo conexo y no dirigido $G = (V, E)$ con dos valores de coste $c(u, v)$, $c(u, v) \geq 0$ para cada arista $u, v \in E$ y con un coste máximo de c_{max}.
  - **Objetivo:** Encontrar una ruta (dirigida) que pase por cada arista en E al menos una vez y con un coste de, a lo sumo, c_{max}.

Desafortunadamente, es NP-duro [35], pues está demostrado que todo PCC mixto puede transformarse en un PCC windy en tiempo polinomial. Sin embargo es resoluble en tiempo polinomial si el coste de pasar por cada ciclo en el grafo de entrada es el mismo independientemente de la dirección, y si el grafo de entrada es Euleriano. Aprovechando esto último, Z.Win [49] obtuvo una aproximación de factor 2, aunque posteriormente Raghavachari y Veeresamy [44] dieron una de factor 3/2 combinando una estrategia mixta similar a la usada en el PCC mixto.

- **Jerarquías de aristas y relaciones de precedencia**

  En aplicaciones prácticas, es posible encontrarnos una variante del PCC donde algunas aristas deban ser recorridas antes que otras. Sea así $G = (V, E)$ un grafo no dirigido, y sea $P = \{E_1, ..., E_k\}$ una partición de las aristas, tal que $\cup_{i=1}^{k} E_i = E$ y $E_i \cup E_j = \emptyset$ para todo $1 \leq i, j \leq k$. Denominaremos a los conjuntos E_i como *clases de prioridad*. Sea además, $R \subseteq P \times P$ una ordenación parcial. Diremos que un camino en G *respeta* R si para todo $(E_i, E_j) \in R$ cada arista en E_i es recorrida antes que cualquier otra en E_j.

  El PCC Jerárquico se puede formular de la siguiente manera:

  - **Entrada:** un grafo no dirigido y conexo $G = (V, E)$ con aristas con coste $c(e) \geq 0$ para cada $e \in E$ y con un coste máximo de c_{max}. Adicionalmente, una partición $P = \{E_1, ..., E_k\}$ y una relación de ordenación parcial $R \subseteq P \times P$.
  - **Objetivo:** Encontrar una ruta que pase por cada arista en E al menos una vez, con un coste de, a lo sumo, c_{max} y que respete R.

  Este problema es NP-duro en general. Sin embargo, hay tres factores que parecen influir en la complejidad, que son [35]:

  * Si las clases están conectadas: Hay que resolver el Problema del Cartero Rural como subproblema si una de las clases de prioridad no está conectada. Sin embargo, está demostrado que el PCC Jerárquico es NP-duro incluso si las clases si lo están.

- Número de clases de prioridad: El PCC Jerárquico es NP-duro incluso cuando sólo hay dos clases de prioridad. Esto se obtiene de la reducción del Problema del Cartero Rural (NP-duro también). Si las clases están conectadas, entonces es necesario tener muchas clases de prioridad en la redución de dureza estándar. Esta se hace para el Problema del Cartero Rural desde el Problema del Ciclo Hamiltoniano (el cual se pregunta si dado un grafo, este admite un camino cerrado pasando una única vez por cada arista y es NP-duro) creando una componente conectada en las aristas obligatorias para cada vértice del caso original.
- Si la relación de precedencia es una ordenación linear en lugar de parcial, caso en el que el PCC Jerárquico con clases de prioridad conectadas es resoluble en tiempo polinomial. Recientemente se han descubiertos algoritmos ejecutables en tiempo $O(k^3|V|^3)$.

- **Múltiples carteros**

  Otra variante es considerar k rutas en un grafo ponderado G, tal que cada arista sea recorrida por al menos uno de los carteros. Formalmente:

  - **Entrada:** un grafo no dirigido y conexo $G = (V, E)$ con coste $c(e) \geq 0$ para cada $e \in E$ y con un coste máximo de c_{max}. Además, un vértice distinguido $v \in V$ (la oficina de correos).
  - **Objetivo:** Encontrar k rutas T^* que empiece y acabe en V, tal que ninguna ruta sea vacía y cada arista en E sea recorrida al menos una vez, con $w(T^*) \leq c_{max}$.

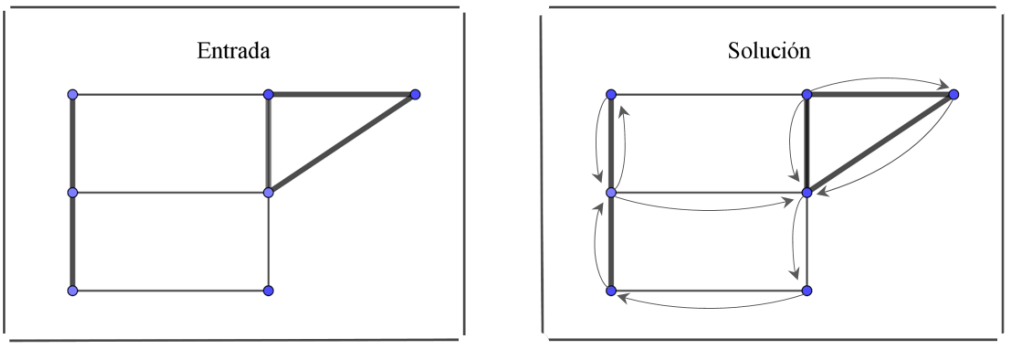
  Centrándonos en el caso no dirigido, la complejidad del k-PCC depende enormemente de que función objetivo w elijamos: Minimizar la suma de los costes de la ruta da lugar al Min-Sum-k-PCC, resoluble en tiempo polinomial, mientras que minimizar el máximo del coste de la ruta genera el Min-Max-k-PCC, que es NP-duro. Esto se sigue manteniendo incluso si sustituimos un grafo dirigido por G. Si ambos arcos y aristas están presentes, entonces ambas variantes tienen el PCC mixto como caso especial, y son NP-duras.

- **Más variantes**

  Otras variantes existentes serían considerar que hay ciertas formas de atravesar cruces prohibidas [3], obligar a pasar por al menos una arista de un subconjunto [14] o maximizar el beneficio de repartir en una calle múltiples veces [20].

1.3.2. Complejidad del Problema del Cartero Chino Rural

El PCC sirve para modelar el reparto de cartas en casas en cada calle. En áreas rurales, sin embargo, no en todas las calles hay casas a las que repartir cartas. A continuación, veremos una representación gráfica de un ejemplo concreto, donde el coste de las aristas vuelve a ser proporcional a su longitud, así como la formalización del problema:



- **Entrada:** un grafo no dirigido y conexo $G = (V, E)$ con aristas con coste $c(e) \geq 0$ para cada $e \in E$, un conjunto $R \subseteq E$ de *aristas obligatorias* (negrita), y con un coste máximo de c_{max}.
- **Objetivo:** Encontrar una ruta que pase por cada arista en R al menos una vez, con un coste de, a lo sumo, c_{max}.

En la siguiente tabla podemos observar un breve resumen de la complejidad del PCC Rural:

Variante del PCC Rural	Aproximación
No dirigido	Factor 3/2 en tiempo $O(\|V\|^3)$ Factor 3/2 en tiempo $O(\|V\|^3)$ para problemas más generales
Mixto	Si se cumple la desigualdad triangular, se tiene: factor 15/8 para el caso simétrico factor 9/5 si solo los arcos dirigidos son los obligatorios
Variante del PCC Rural	**Complejidad Parametrizada**
No dirigido	Algoritmo en tiempo $O(\|V\|^{2\gamma}/\gamma! \cdot n)$ Problema kernel de $2\|R\|$ vértices Algoritmo en tiempo $O(2^{3\|R\|} \cdot \|R\|^2 + \|V\|^3)$ Algoritmo aleatorizado en tiempo $O(2^\gamma \cdot (c_{max} + \|V\|)^d)$, con d = cte.
Dirigido	Algoritmo en tiempo $O(4^k \cdot \|V\|^3)$ Algoritmo en tiempo $O(4^{\gamma log(b\gamma^2)} \cdot \|V\|^4)$ Algoritmo aleatorizado en tiempo $O(2^\gamma \cdot (c_{max} + \|V\|)^d)$, con d = cte. No existen problemas kernel con tamaño polinomial con respecto de γ y k

Todas las variantes del Problema del Cartero Rural son NP-completas si no todas las aristas son obligatorias, lo cual haría al PCC Rural equivalente al PCC.

Acerca de la complejidad clásica, decir que si reducimos el Problema del Ciclo Hamiltoniano al PCC Rural, probamos que este último es también NP-duro. Como es análogo para el caso dirigido, podemos afirmar que las versiones dirigida, mixta y windy del PCC Rural son también NP-duras.

Los problemas de los carteros son muy similares a problemas en grafos relacionados con la propiedad euleriana. De hecho, un circuito cerrado en un grafo puede considerarse como un multigrafo. Así, el PCC Rural puede transformarse en otro, al que llamaremos su Extensión Euleriana (y viceversa), que se describiría de la siguiente manera:

- **Entrada:** un multigrafo $G = (V, E)$ con coste $c(\{u, v\}) \geq 0 \ \forall u, v \in V$, y con un coste máximo de c_{max}.
- **Objetivo:** Encontrar un multiconjunto E' de pares $\{u, v\}$ con $u, , v \in V$ añadiendo las aristas en E' a G para hacer que G sea euleriano y con $c(E') \leq c_{max}$.

Acerca de la aproximabilidad, señalar que, aunque el Problema del Viajante esté muy relacionado con el PCC Rural, éste probablemente no puede ser aproximado al óptimo con ningún factor constante. Sin embargo, existen ciertas aproximaciones de factor constante para casos especiales del Problema del Viajante son la base para los algoritmos de aproximación de factor constante del PCC Rural.

Para el caso de los grafos dirigidos, es desconocido si es posible encontrar una aproximación con factor constante. Para la resolución de grafos mixtos *simétricos* (donde para cada arco $\{u, v\}$ en la entrada, hay un arco $\{v, u\}$ con igual coste), se ha descubierto una aproximación de factor 15/8 si se cumple la condición de la desigualdad triangular. Bajo la misma condición también se ha encontrado una aproximación de factor 9/5 si solo los arcos dirigidos son los obligatorios.

Sobre la complejidad de parametrización, señalar que la transformación anteriormente comentada del PCC Rural en el Problema del Vendedor Ambulante crea casos con a lo sumo $3|R|$ vértices. Esto puede usarse para algoritmos parametrizados combinándolo con el conocido algoritmo de programación dinámica, resolviendo el PCC Rural en tiempo $O(2^{3|R|} \cdot |R|^2 + |V|^3)$, siendo R el conjunto de aristas obligatorias. El PCC Rural también admite un problema kernel sencillo con $2|R|$ vértices. Para el caso de grafos dirigidos y dado un entero k, existe un algoritmo de programación dinámica que computa el mínimo coste de una ruta pasando por todos los arcos obligatorios más, a lo sumo, k arcos adicionales. Dicho algoritmo se ejecuta en tiempo $O(4^k \cdot |V|^3)$.

Teorema 6. *El PCC Rural es TPF con respecto al número de arcos que recorre una solución óptima más los arcos obligatorios.*

Aunque posteriormente se descubrió un algoritmo ejecutable en tiempo $O(|V|^{2\gamma}/\gamma! \cdot n)$, aún no se sabe si el PCC Rural es TPF respecto a γ. Se ha demostrado que el PCC Rural respecto a γ es "TPF-equivalente"a una variante de emparejamiento NP-dura. Existe además un algoritmo que resuelve el PCC Rural en tiempo $O(4^{\gamma log(b\gamma^2)} \cdot |V|^4)$, donde b representa el número de vértices no equilibrados (con grado de entrada distinto del de salida).

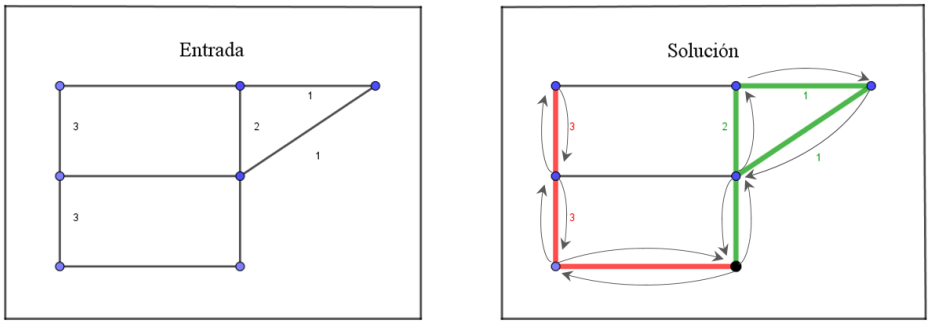
Teorema 7. *El PCC Rural puede resolverse en tiempo* $O(2^\gamma \cdot (c_{max} + |V|)^{d+l})$ *para dado un* $l > 0$ *real y* $d > 0$ *constante, describiendo un caso sí como un caso no, con probabilidad a lo sumo* $1/(c_{max} + |V|)^l$.

Dicho resultado es válido tanto para el caso dirigido como el no dirigido.

1.3.3. Complejidad del Problema de las Rutas por Arcos con Capacidades

El problema más general de rutas por arcos que vamos a estudiar es el de Rutas por Arcos con Capacidades (RAC), introducido por Golden y Wong [26]. Este generaliza el PCC y el PCC Rural, y centrándonos en el caso concreto de la mensajería, añade la posibilidad de repartir el correo a un conjunto de calles obligatorias a través de múltiples vehículos capaces de llevar un número limitado de paquetes.

En un RAC, se nos da un grafo no dirigido, con un coste asociado a cada arista y algunas de ellas con una cierta demanda y un vértice especial, al que denominaremos el *almacén*. Los almacenes contienen un número ilimitado de vehículos, cada uno con una capacidad, que pueden *repartir* paquetes en las aristas o simplemente *recorrerlas* sin repartir. En el siguiente gráfico podemos ver un ejemplo en el que el vértice negro es el almacén, las demandas respectivas aparecen como etiquetas de las aristas, y las longitudes de estas son proporcionales a sus costes:



Todos los vehículos tienen capacidad seis. El objetivo es repartir en cada arista, de acuerdo con su demanda, con un único vehículo, sin que la suma de las demandas de las aristas en las que se reparte excedan la capacidad del vehículo. Además, buscamos que el coste de recorrer las aristas sea mínimo. Las rutas de los dos vehículos de la solución se representan en distintos colores. Formalmente tenemos:

- **Entrada:** un grafo no dirigido $G = (V, E)$, con un vértice almacén $v_0 \in V$, con aristas con coste $c(e) \geq 0$ y demanda $d(e) \geq 0$ para cada $e \in E$, una capacidad W de vehículo, y un coste máximo de c_{max}.
- **Objetivo:** Encontrar un conjunto $\mathscr{C}$ de ciclos en G, cada uno correspondiendo a la ruta de un vehículo, pasando por el vértice v_0, tal que:

1. $\sum_{C \in \mathscr{C}} \sum_{e \in C} c(e) \leq c_{max}$.
2. Cada arista e con $d(e) > 0$ es recorrida por al menos un ciclo, y en ella reparte exactamente un vehículo.
3. La suma de las demandas de las aristas en las que se reparte con un único vehículo es a lo sumo W.

Si consideráramos que todas las demandas son 1 y que la capacidad de los vehículos es el número de aristas del grafo, tendríamos el PCC. Si solo consideráramos un subconjunto de aristas con demanda 1, tendríamos el PCC Rural.

Acerca de la complejidad clásica comentaremos que el RAC es NP-duro no solo referido al problema de rutas subyacente, sino al problema de distribuir la demanda de las aristas entre los diversos vehículos. Una forma posible de abordaje es reducir el RAC a un problema denominado Bin Packing. En este, los objetos de diferentes volúmenes deben empaquetarse en un número finito de contenedores de volumen W, de manera que se minimice el número de contenedores usados.

De tal reducción obtenemos el siguiente resultado, extendiendo la NP-dureza a problemas de RAC más generales:

Teorema 8. *El problema de Rutas por Arcos con Capacidades es NP-duro incluso si todas las aristas son obligatorias de ser visitadas, si el coste máximo y el número de vehículos requeridos es constante, y si el mayor grado de los vértices es al menos dos.*

Acerca de la complejidad clásica, decir que si reducimos el Problema del Ciclo Hamiltoniano al PCC Rural, probamos que este último es también NP-duro. Como es análogo para el caso dirigido, podemos afirmar que las versiones dirigida, mixta y windy del PCC Rural son también NP-duras.

Otra opción es transformar el RAC en el problema de Rutas de Vehículos con Capacidades, que nos proporciona soluciones exactas para el RAC. Este es un problema de rutas por nodos que se plantea de la siguiente manera:

- **Entrada:** Un conjunto V de vértices con un vértice almacén $v_0 \in V$, con costes $c(\{u, v\}) \geq 0 \ \forall u, v \in V$ y con demanda en vértices de $d(v) \geq 0$ para todo $v \in V$, un vehículo de capacidad W, y un coste máximo de c_{max}.

- **Objetivo:** Encontrar una partición $C_1 \uplus \cdots \uplus C_l = \{v \in V \setminus \{V_0\} | d(v) > 0\}$, siendo C_i una permutación σ_i tal que, para $1 \leq i \leq l$, se cumple que $\sum_{v \in C_l} d(v) \leq W$, y tal que:
$$\sum_{i=1}^{l} \left(c(\{v_0, \sigma_{i,1}\}) + c(\{v_0, \sigma_{i,|C_i|}\}) + \sum_{j=1}^{|C_i|-1} c(\{\sigma_{i,j}, \sigma_{i+1,j}\}) \right) \leq c_{max}$$

Esta transformación no es correcta en general, excepto si el caso de entrada de Rutas de Vehículos respeta la desigualdad triangular, pues una solución para el caso de salida del RAC puede ser transformada en una solución de Rutas de Vehículos de igual coste.

Acerca de la aproximabilidad, cabe destacar el siguiente resultado [35]:

Teorema 9. *Computar una aproximación de factor 3/2 para el problema de Rutas por Arcos con Capacidades es NP-duro. Una aproximación de factor* $(7/2 - 3/W)$ *puede ser computada en tiempo polinomial.*

Este factor de aproximación puede obtenerse sin la desigualdad triangular.

La complejidad parametrizada del RAC es aun un campo desconocido. Si no requerimos que los datos de entrada respeten la desigualdad triangular, el Teorema 8. excluye los parámetros "coste máximo c_{max}", anchura del camino, el máximo número de vehículos, y el grado máximo del grafo de entrada con respecto al carácter tratable de los parámetros fijos. Así, el RAC es NP-duro incluso si todos los parámetros son constantes. Es más, uno de los grandes desafíos acerca de la complejidad parametrizada del RAC consiste en identificar parámetros significativos y pequeños que hacen el problema tratable.

1.4. Conclusión

En los últimos años, los mayores avances en este problema han tenido lugar en el diseño de algoritmos de programación lineal entera, basados en la ramificación y acotación, la generación de columnas, y su forma híbrida.

El desarrollo de algoritmos exactos y heurísticos para problemas de rutas por arcos ha alcanzado un nivel muy alto de sofisticación. Es posible que se haya llegado a un punto en el que sea deseable diseñar algoritmos más simples, aunque pudiera

dar como resultado una pequeña pérdida de precisión. Es innegable incluso que aparecerán nuevas variantes de problemas de rutas por arcos.

Por otro lado, clasificar la complejidad computacional de un problema se encuentra en el núcleo de su desarrollo, así como justificar las líneas de ataque para resolverlo. De hecho, muchos métodos para resolver estos problemas se basan en la heurística y la programación matemática, pues la mayoría son NP-duros. Puede ser posible identificar casos especiales resolubles en tiempo polinomial, para desarrollar algoritmos de aproximación eficientes, o para identificar parámetros específicos del problema a explotar. Finalmente, señalar el planteamiento realizado de ciertos problemas abiertos, igual de importantes, pues su respuesta ayudaría a obtener una visión más refinada sobre la complejidad computacional de problemas de rutas por arcos.

2

El Problema del Cartero Chino no dirigido

2.1. Planteamiento

El **Problema del Cartero Chino** (PCC) podría considerarse el problema más importante en el ámbito de las rutas por arcos. En esta sección, estudiaremos con detalle la versión no dirigida del PCC y como abordarla, tanto desde el punto de vista de la programación dinámica como desde la programación matemática. También estudiaremos sus cuatro variantes: el PCC generalizado, el PCC acumulativo, el PCC jerárquico, y el PCC con ventanas temporales.

El PCC está definido sobre un grafo no dirigido $G = (V, E)$, donde:

- $V = \{v_1, \cdots, v_n\}$ es el conjunto de vértices
- $E = \{(i, j) : \ v_i, v_j \in V, \ i < j\}$ es el conjunto de aristas
- c_{ij} es el coste asociado a cada arista (i, j) por atravesarla.

El problema es determinar el camino cerrado de menor coste que pasa por todas las aristas del grafo. Cuando G es conexo y todos los vértices tienen grado par, el grafo es Euleriano, lo que significa que existe una solución para el PCC que pasa por cada arista una sola vez. Como vimos en la introducción, la conexión y la paridad son dos condiciones necesarias y suficientes para la caracterización euleriana.

Por tanto, para que una solución factible del PCC exista, el grafo debe ser conexo. Si también tuvieramos la paridad, solo sería necesario encontrar el camino Euleriano en nuestro grafo Euleriano. Sin embargo, la dificultad aparece cuando

esto no ocurre. En tal caso, buscaríamos encontrar la solución óptima.

2.2. Resolución

La forma de abordarlo consistiría en determinar un aumento del grafo de menor coste que haga que todos los grados sean pares, es decir, identificando un subconjunto de aristas que serán atravesadas más de una vez. Además, nunca será óptimo pasar por la misma arista más de dos veces, ya que los pares de pases pueden eliminarse sin desconectar el grafo y sin cambiar la paridad de los grados. Un aumento de menor coste se obtiene determinando un emparejamiento de coste mínimo de vértices de grado impar de G, donde el coste del emparejamiento de v_i a v_j es el coste del camino más corto de v_i a v_j de G.

Veremos dos maneras de hallar este grafo de aumento, y por consiguiente, de resolver el PCC, mediante programación dinámica y mediante programación matemática.

2.2.1. Programación dinámica

Una manera de resolver el PCC es mediante la programación dinámica. Los primeros en abordarlo de esta forma fueron Bellman y Cook [2]. Definimos así:

- V^- como el conjunto formado por los vértices de V con grado impar.
- M como un conjunto de caminos μ_{ij} de G entre los vértices v_i y v_j, $\forall v_i, v_j \in V^-$, tal que no haya dos caminos con ningún vértice igual, es decir, que sean caminos disjuntos de vértices de V^-, formando así un emparejamiento.

Comentar en primer lugar ciertos aspectos que nos conducirán posteriormente al algoritmo que resuelva nuestro problema:

- La suma de los grados d_i de todos los vértices $v_i \in V$ es igual a dos veces el número de aristas de E (pues cada arista une dos vértices, aumentando el grado de éstos), y por lo tanto es un número par $2m$, siendo $m = |E|$. Luego:

$$\sum_{v_i \in V} d_i = \sum_{v_i \in V^+} d_i + \sum_{v_i \in V^-} d_i = 2m$$

Ambos sumandos son pares, lo que significa que como todos los d_i en el segundo sumando son impares, el número $|V^-|$ de vértices de grado impar, es par.

- El número de caminos μ_{ij} en M es $\frac{1}{2}|V^-|$. Este está bien definido, pues al ser $|V^-|$ es par, entonces será entero.

Suponemos ahora que todas las aristas que forman parte de un camino μ_{ij} se añaden a G como aristas artificiales, e irán en paralelo con las aristas de G que ya estaban ahí (es decir, ahora todo camino μ_{ij} está doble). Esto ocurre para todo camino $\mu_{ij} \in M$ y el grafo resultante se denomina $G^-(M)$. Como algunas aristas de G puede que aparezcan en más de un camino μ_{ij}, algunas aristas de $G^-(M)$ pueden estar tres veces, cuatro, etc.

Formulamos a continuación el siguiente teorema:

Teorema 10. *Para cualquier circuito de G, hay alguna opción de M para la cual $G^-(M)$ posee un circuito euleriano correspondiente al circuito de G. La correspodencia es tal que si un circuito pasa por una arista (i, j) de G l veces, hay l aristas (una real y $l-1$ artificial) entre v_i y v_j en $G^-(M)$, cada una de ellas es atravesada por exactamente un circuito Euleriano de $G^-(M)$, y recíprocamente.*

Demostración. Si un circuito pasa por G, entonces al menos una arista incidente en todo vértice $v_i \in V$ con grado impar debe ser atravesada dos veces (por el Teorema 1). Una arista atravesada dos veces puede ser considerada como dos aristas paralelas, una real y otra artificial, ambas atravesadas una vez.
Sea dicha arista (i, k):

- Si el grado d_k de v_k en G es impar, entonces al añadir la arista artificial convertimos a d_k en par, y solo esta arista necesitará ser atravesada dos veces hasta que v_i y v_k estén asociados.

- Si d_k es par, entonces al añadir la arista artificial hará ahora que d_k sea impar y una segunda arista que vaya desde v_k debe ser atravesada dos veces (es decir, se añade otra arista artificial).

El razonamiento continua desde v_k hasta que un vértice de grado impar sea alcanzado como hemos mencionado anteriormente. Así, para satisfacer la condición

de traversabilidad en v_i, se debe atravesar dos veces un camino completo desde v_i hasta algún otro vértice v_r de V^- de grado impar. Esto satisface automáticamente la condición de traversabilidad del vértice v_r.

Tal proceso es análogo para el resto de vértices v_i de V^-, lo que significa que todo el conjunto M de caminos de G debe ser atravesado dos veces, y como esto implica que cada arista de $G^-(M)$ debe ser atravesada una vez, el teorema se cumple. □

El algoritmo para la solución del Problema del Cartero Chino es consecuencia inmediata del teorema anterior, ya que ahora lo que tenemos es que encontrar tal conjunto de caminos M^* (hacer un emparejamiento de vértices de grado impar) que produce el menor coste adicional. El circuito de menor coste adicional que pasa por G deberia entonces tener un coste igual a la suma de los costes de las aristas de G más la suma de los costes de las aristas en los caminos de M^*. Esto es lo mismo que la suma de los costes de todas las aristas, reales y artificiales, del grafo $G^-(M^*)$. A continuación, describiremos el algortimo que dará solución a PCC. Para ello, utilizaremos otro que veremos previamente, denominado el Algoritmo del Camino Más Corto.

Algoritmo del Camino Más Corto

Este algoritmo calcula el camino más corto del vértice a al b, considerando que los costes son no negativos. El método consiste en asignar temporalmente etiquetas a los vértices, donde tal etiqueta será un límite superior en la longitud de la rute desde a hasta ese vértice. La etiquetas están continuamente reduciéndose mediante un proceso iterativo.

Descripción: Sea C la matriz de costes, con $c_{ij} \geq 0$. Sea $l(v_i)$ la etiqueta del vértice v_i.

Inicialización:

- Paso 1: Consideramos $l(a) = 0$ y marcamos la etiqueta como permanente (tomaremos como marca un +). Consideramos también $l(v_i) = \infty$ para todo $v_i \neq a$ y marcamos tales etiquetas como temporales. Tomamos $p = a$.

Actualización de etiquetas:

- Paso 2: Para todo $v_i \in \Gamma(p)$ (siendo $\Gamma(p)$ el conjunto de los vértices adyacentes de p) actualizamos aquellas etiquetas que sean temporales de la forma siguiente:

$$l(v_i) = min[l(v_i), l(p) + c(p, v_i)]$$

Fijación de una etiqueta como permanente:

- Paso 3: Entre todos los vértices etiquetados temporalmente encontrar el v_i^* para el que $l(v_i^*) = min[l(v_i)]$.
- Paso 4: Marcar la etiqueta de v_i^* como permanente y fijar $p = v_i^*$.
- Paso 5:
  - Si todos los vértices son etiquetados de forma permanente, entonces las etiquetas son las longitudes de los caminos más cortos. *Stop.*
  - Si algunas etiquetas son temporales volvemos al paso 2. *Stop.*

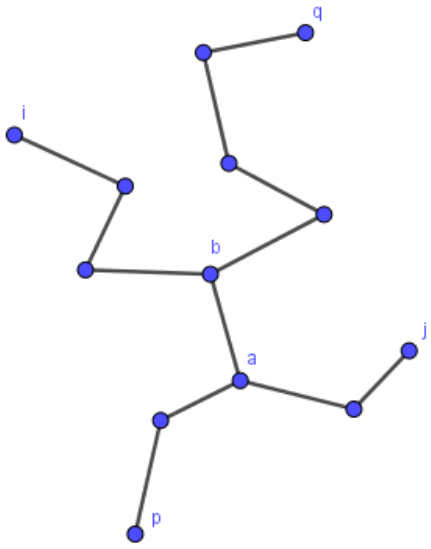
Algoritmo para resolver el PCC

Descripción:

- Paso 1: Sea $[c_{ij}]$ la matriz de coste de las aristas del grafo G. Usando el algoritmo anterior del camino más corto formamos la matriz $D = [d_{ij}]$, de dimensión $|V^-|$ por $|V^-|$, donde d_{ij} es el coste del camino más corto desde un vértice $v_i \in V^-$ hasta otro vértice $v_j \in V^-$.
- Paso 2: Encontrar el emparejamiento M^* de vértices en V^- que produce el menor coste de acuerdo con la matriz de costes D.
- Paso 3: Si un vértice v_α es asociado con otro vértice v_β, identificamos el camino de menor coste $\mu_{\alpha\beta}$ (desde v_α hasta v_β) correspondiente al coste $d_{\alpha\beta}$ del paso 1. Insertamos artistas artificiales en G correspondientes a las aristas en $\mu_{\alpha\beta}$ y repetimos para el resto de caminos del emparejamiento M^* para obtener el grafo $G^-(M^*)$.

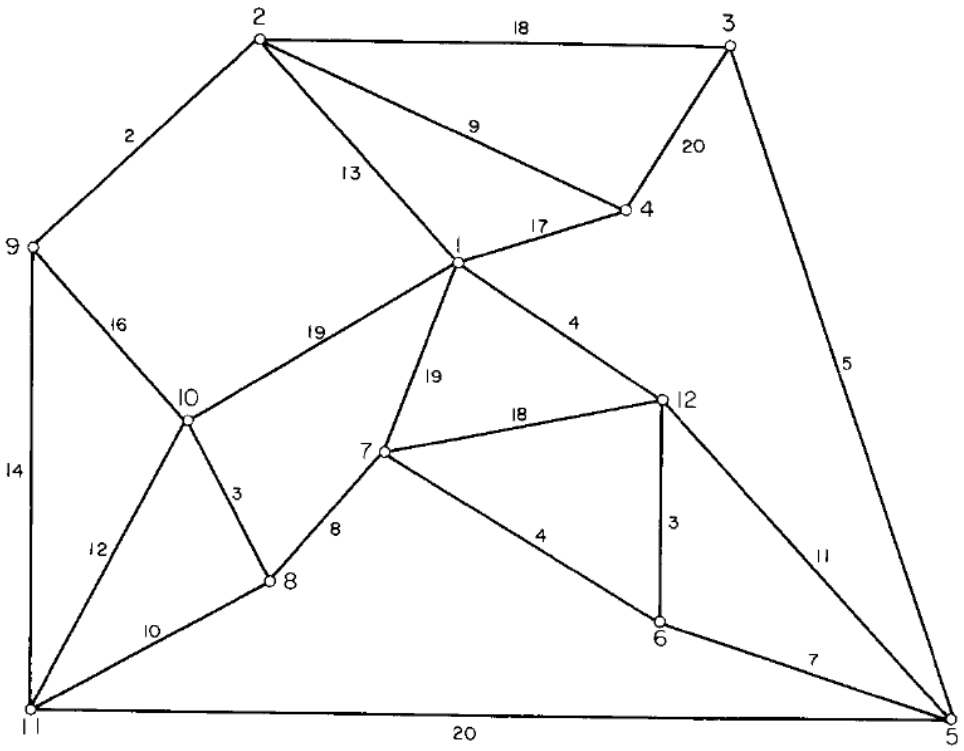
- Paso 4: La suma de los costes de la matriz $[c_{ij}]$ de todas las aristas en $G^-(M^*)$ (considerando que el coste de una arista artificial es el mismo que el de una arista real en paralelo con ella) es el circuito de mínimo coste que pasa por G. El número de veces que este circuito pasa por una arista (v_i, v_j) es el número total de aristas en paralelo entre v_i y v_j en $G^-(M^*)$.

Señalar que, si se verifica la desigualdad triangular, como en el paso 2 estamos usando un emparejamiento mínimo, no hay dos caminos más cortos, μ_{ij} y μ_{pq} (es decir, los que van de v_i a v_j, y de v_p a v_q), en el emparejamiento con aristas en común. Esto es porque si tienen una arista en común (v_a, v_b) como podemos ver en el siguiente dibujo, entonces un emparejamiento de v_i a v_q (usando los subcaminos que van de v_i a v_b, y de v_b a v_q), y un emparejamiento de v_p a v_j (usando los subcaminos que van de v_p a v_a, y de v_a a v_j), producen un emparejamiento global de coste $2c_{ab}$ menor que el del emparejamiento original. Y esto es una contradicción a la suposición de que el emparejamiento original era mínimo.



Esto significa que el grafo $G^-(M^*)$ no tiene más de dos aristas en paralelo entre dos vértices cualquiera, es decir, el circuito óptimo nunca pasa más de dos veces por ninguna arista de G.

Ejemplo: Consideramos el grafo G siguiente:



Como vemos, G tiene 12 vértices y 22 aristas, cada cual con su coste correspondiente. El problema es encontrar un circuito que pase por todas las aristas de G al menos una vez y tenga coste mínimo.

1. El conjunto V^- formado por vértices de grado impar para el grafo es $\{1, 3, 4, 6, 8, 9\}$. Usando el algoritmo del camino más corto, construimos la matriz D:

   Primera iteración:

   - Paso 1: Fijamos:
     $$\begin{cases} l(v_1) = 0^+ \\ l(v_i) = \infty \ \forall v_i \neq v_1,\ p = v_1 \end{cases}$$
   - Paso 2: Como:
     $\Gamma(p) = \Gamma(v_1) = \{v_2, v_4, v_7, v_{10}, v_{12}\}$
     y todos tienen etiquetas temporales, actualizamos las etiquetas (primero lo haremos con v_2):
     $l(v_2) = min[\infty, 0^+ + 13] = 13$
     De forma análoga obtenemos que:

$l(v_4) = 17$, $l(v_7) = 19$, $l(v_{10}) = 19$, $l(v_{12}) = 4$

- Paso 3: Calculamos el $min\ [\underbrace{13}_{v_2}, \underbrace{17}_{v_4}, \underbrace{19}_{v_7}, \underbrace{19}_{v_{10}}, \underbrace{4}_{v_{12}}, \underbrace{\infty}_{v_3,v_5,v_6,v_8,v_9,v_{11}}] =$ 4, el cual corresponde con v_{12}.
- Paso 4: Marcamos como permanente $l(v_{12}) = 4^+$, con $p = v_{12}$.
- Paso 5: Como no todos los vértices están etiquetados como permanentes, vamos al paso 2.

Segunda iteración:

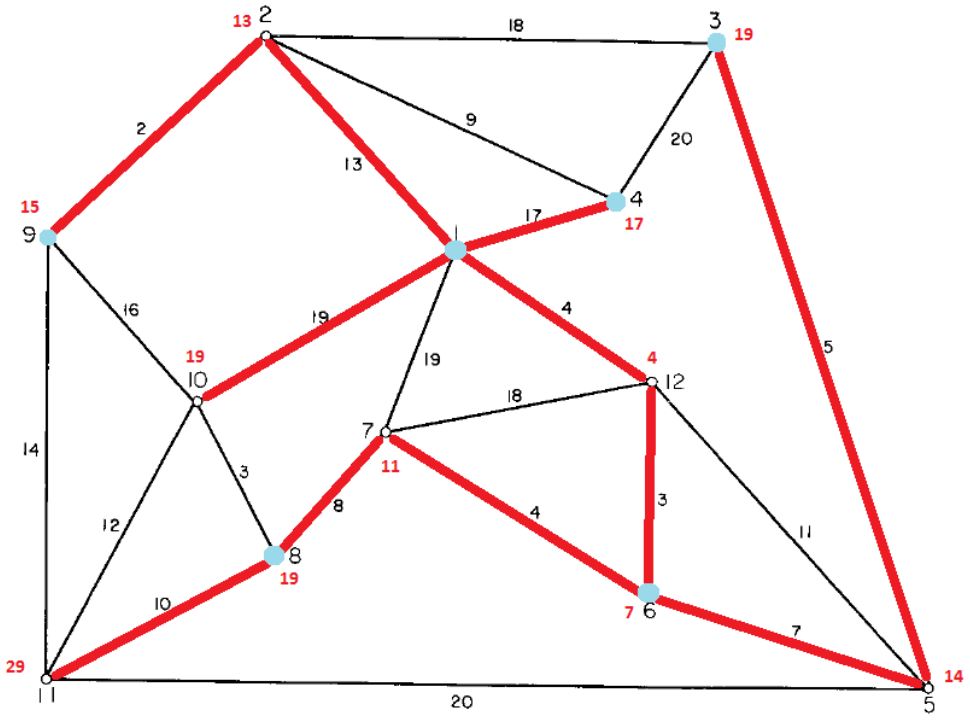
- Paso 2: Como:
  $\Gamma(p) = \Gamma(v_{12}) = \{v_5, v_6, v_7\}$
  y todos tienen etiquetas temporales, actualizamos las etiquetas (primero lo haremos con v_5):
  $l(v_7) = min[19, 4^+ + 18] = 19$
  De forma análoga obtenemos que:
  $l(v_5) = 15$, $l(v_6) = 7$
- Paso 3: Calculamos el $min\ [\underbrace{13}_{v_2}, \underbrace{17}_{v_4}, \underbrace{15}_{v_5}, \underbrace{7}_{v_6}, \underbrace{19}_{v_7}, \underbrace{19}_{v_{10}}, \underbrace{\infty}_{v_3,v_8,v_9,v_{11}}] =$ 7, el cual corresponde con v_6.
- Paso 4: Marcamos como permanente $l(v_6) = 7^+$, con $p = v_6$.
- Paso 5: Vamos al paso 2.

Tercera iteración:

- Paso 2: Como:
  $\Gamma(p) = \Gamma(v_6) = \{v_5, v_7, v_{12}\}$
  donde $\{v_5, v_7\}$ tienen etiquetas temporales. Actualizamos las etiquetas y obtenemos:
  $l(v_5) = 13$, $l(v_7) = 14$
- Paso 3: Calculamos el $min\ [\underbrace{13}_{v_2}, \underbrace{17}_{v_4}, \underbrace{13}_{v_5}, \underbrace{7}_{v_6}, \underbrace{14}_{v_7}, \underbrace{19}_{v_{10}}, \underbrace{\infty}_{v_3,v_8,v_9,v_{11}}] =$ 13, el cual corresponde con v_5.
- Paso 4: Marcamos como permanente $l(v_5) = 13^+$, con $p = v_5$.

- Paso 5: Vamos al paso 2.

Continuamos haciendo iteraciones hasta tener todas los vértices etiquetados de forma permanente. El resultado final del proceso visualmente es el siguiente:



Las líneas rojas representan los caminos más cortos desde el vértice v_1 hasta cada uno de los vértices de G. Una vez que tenemos todos estos resultados, nos interesa quedarnos solo con los correspondientes a los vértices de grado impar. Estos aparecen representados en azul. De esta manera, ya podemos formar la primera fila de nuestra matriz D, es decir, la fila correspondiente a v_1 (de grado también impar).

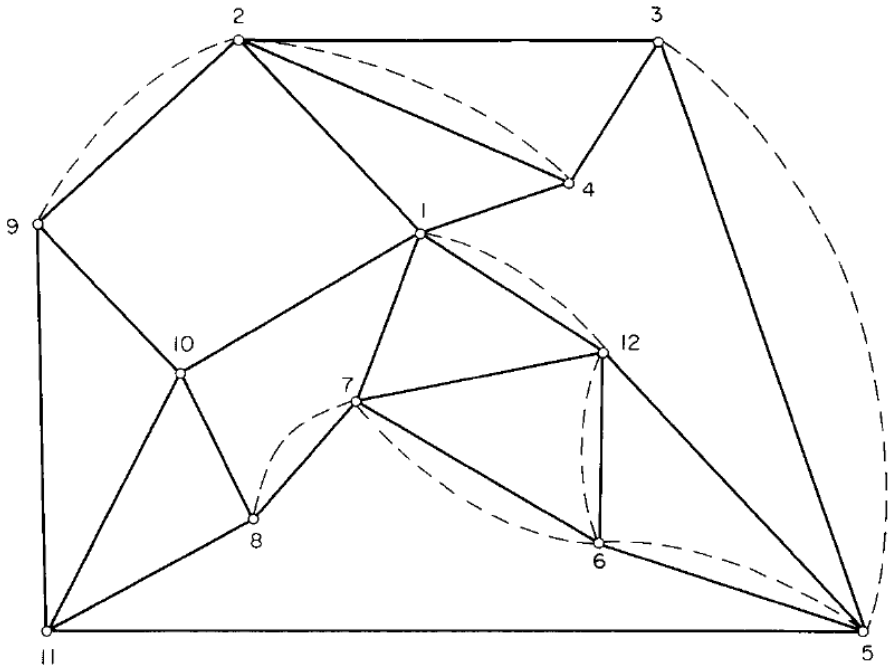
Repetimos el proceso para los vértices de G con grado impar restantes, y obtenemos que D tiene la siguiente forma:

	1	3	4	6	8	9
1	-	19	17	7	19	15
3	19	-	20	12	24	20
4	17	20	-	24	30	11
6	7	12	24	-	12	22
8	19	24	30	12	-	19
9	15	20	11	22	19	-

2. En el paso 2 el algoritmo del emparejamiento mínimo relaciona los siguientes vértices:

   1 con 6, camino: 1-12-6, coste 7
   3 con 8, camino: 3-5-6-7-8, coste 24
   9 con 4, camino: 9-2-4 coste 11

3. El grafo $G^-(M^*)$ al final del paso 3 tiene la siguiente forma, donde las aristas artificiales son representadas como líneas discontinuas:



Una vez que hemos construido el grafo $G^-(M^*)$, un circuito Euleriano de

este grafo y, por lo tanto, el circuito óptimo correspondiente que atraviesa el gráfico original G, puede construirse inmediatamente mediante la regla de Fleury que mencionaremos posteriormente.

2.2.2. Programación matemática

El documento de Edmonds y Johnson [18] nos proporcionó el primer tratamiento de programación matemática del PCC. Se dice así que $S \subset V$ es impar si es un conjunto formado por un número impar vértices de grado también impar. Sea $E(S) = \{(i,j) : \ v_i \in S, \ v_j \in V \setminus S, \ \text{o} \ v_i \in V \setminus S, \ v_j \in S\}$. Consideraremos las variables x_{ij}, tales que:

$$x_{ij} = \begin{cases} 1 & \text{si y solo si la arista } (i,j) \text{ aparece en el aumento} \\ 0 & \text{en caso contrario} \end{cases}$$

El problema puede formularse de la siguiente forma:

$$\begin{aligned} \text{(PCC)} \quad min \quad & \textstyle\sum_{(i,j)\in E} c_{ij} x_{ij} \\ s.t. \quad & \textstyle\sum_{(i,j)\in E(S)} x_{ij} \geq 1, \quad S \subset V, \forall \text{ S impar}, && (2.1) \\ & x_{ij} \geq 0, \quad (i,j) \in E, \\ & x_{ij} \text{ entero}, \quad (i,j) \in E, && (2.2) \end{aligned}$$

Las restricciones anteriores representan las desigualdades *blossom*. Éstas representan que al menos una copia de una arista incidente en un conjunto impar S debe formar parte del grafo aumentado, con el fin de lograr la paridad. El poliedro de soluciones de (2.2) es igual a la envolvente convexa de las soluciones del problema y estas desigualdades blossom pueden separarse en tiempo polinomial, resolviendo el problema mediante una adaptación del algoritmo blossom de problemas de emparejamiento (Edmonds [18]). Por tanto (2.2) no es necesario y el problema se puede resolver mediante programación lineal continua.

Una vez que obtenemos el grafo Euleriano, determinar el ciclo Euleriano en el puede lograrse aplicando el algoritmo de Fleury [22]:

- Paso 1: Consideramos un vértice v_i. Empezando por v_i, pasamos por una arista (v_i, v_j) que no sea una arista puente (es decir, que su eliminación no suponga la desconexión del grafo), a no ser que sea una arista final. Eliminamos (vi, vj) del grafo.

- Paso 2: Si todas las aristas han sido eliminadas, paramos.
- Paso 3: Fijamos $v_i := v_j$, y volvemos al paso 1.

La principal dificultad de esta algortimo es determinar si una arista es arista puente.

Centrandonos en el ejemplo anterior, y aplicando este algoritmo, un circuito Euleriano posible es el dado por la siguiente secuencia de vértices (empezando por el vértice 1):

$$\{1, 4, 3, 5, 3, 2, 4, 2, 1, 12, 5, 6, 7, 8, 10, 1, 7, 6, 12,$$
$$6, 5, 11, 9, 2, 9, 10, 11, 8, 7, 12, 1\}$$

Este circuito se corresponde con el óptimo que atraviesa G. Aunque no es único, pues hay otro que pasa por el mismo conjunto de 8 aristas dos veces. Por lo tanto, una vez que se determinan las aristas de G que deben atravesarse dos veces por un circuito óptimo, puede haber, en general, más que un circuito que atraviese G.

3

El Problema del Cartero Chino para grafos dirigidos, mixtos y windy

Sea $G = (V, E, A)$ un grafo mixto, donde:

- V es un conjunto de vértices
- E es un conjunto de aristas no dirigidas
- A es un conjunto de arcos dirigidos

Además, usaremos el término *eje* para referirnos tanto a aristas como arcos. Usualmente, los ejes tienen costes asociados no negativos, tanto en grafos no dirigidos, como dirigidos y windy.

Diremos que G es un grafo windy si es no dirigido y hay dos costes relacionados con cada arista representando el coste de atravesarla en cada dirección. Además, un grafo mixto es un caso especial de grafo windy, ya que cada arco (i,j) con coste c_{ij} puede ser modelado como una arista (i,j) con el mismo coste que el arco y con infinito coste en la dirección contraria.

El PCC consiste en encontrar una ruta cerrada de coste mínimo que pasa por cada arista de G al menos una vez. Si un grafo G (dirigido, mixto o windy) es euleriano, existe una ruta óptima que pasa por cada arista una sola vez. Si no, el problema consistirá en encontrar el conjunto de copias de ejes con un coste mínimo, tal que cuando las añadamos a G, consigamos un grafo euleriano.

Recordemos finalmente que, para el caso dirigido, el PCC se llama *Problema del Cartero Chino dirigido* (PCCD), para el caso mixto, *Problema del Cartero Chino*

mixto (PCCM) y para el caso windy, se denomina *Problema del Cartero windy* (PCW). Estos serán los problemas que estudiaremos en este capítulo.

3.1. El Problema del Cartero Chino dirigido

Dado un grafo $G = (V, A)$ dirigido fuertemente conexo, con costes c_{ij} no negativos para cada $(i, j) \in A$, el PCCD consiste en encontrar una ruta de coste mínimo que pase por cada arco en A al menos una vez.

Para formular el problema introduciremos la siguiente notación. Para cada vértice v_i, sea:

- $\delta^+(i)$ el conjunto de arcos que salen de v_i
- $\delta^-(i)$ el conjunto de arcos que entran en i
- $s_i = |\delta^-(i)| - |\delta^+(i)|$

Además, para cada arco $(i, j) \in A$, sea x_{ij} el número de copias de cada arco que deben ser añadidas a G con el fin de obtener un grafo Euleriano. Edmonds y Johnson [18] probaron que el PCCD puede formularse mediante programación lineal (PL) de la siguiente manera:

$$\begin{aligned} \text{(PCCD)} \quad & min \quad \textstyle\sum_{(i,j)\in A} c_{ij}x_{ij} \\ & s.t. \quad \textstyle\sum_{(i,j)\in\delta^+(i)} x_{ij} - \sum_{(i,j)\in\delta^-(i)} x_{ij} = s_i, \quad \forall v_i \in V, \qquad (3.1) \\ & \qquad\qquad x_{ij} \geq 0, \quad \forall (i,j) \in A, \qquad (3.2) \end{aligned}$$

Esto corresponde con un problema de flujo de coste mínimo en G con suministros s_i para los vértices v_i con $s_i > 0$, y demandas $-s_i$ para los vértices v_i $s_i < 0$ ([36], [18]).

3.2. El Problema del Cartero Chino mixto

Dado un grafo $G = (V, E, A)$ con:

- Un conjunto de vértices V,
- Un conjunto de aristas E,

- Un conjunto de arcos A, y
- Un coste no negativo c_a por cada $a \in E \cup A$.

El PCCM consiste en encontrar una ruta de mínimo coste que pase por cada arista $a \in E \cup A$ al menos una vez.

Dado $S \subset V$, sean entonces los siguientes conjuntos:

- $\delta(S)$ es el conjunto de aristas tal que únicamente uno de sus dos puntos extremos está en S.
- $\delta^+(S) = \{(i,j) \in A : v_i \in S, v_j \in V \backslash S\}$
- $\delta^-(S) = \{(i,j) \in A : v_i \in V \backslash S, v_j \in S\}$
- $\delta^*(S) = \delta(S) \cup \delta^+(S) \cup \delta^-(S)$

Dado un vértice v_i, definimos:

- El grado (d_i) es el número de ejes incidentes con v_i.
- El grado interior (d_i^-) es el número de arcos entrantes en v_i.
- El grado exterior (d_i^+) es el número de arcos salientes de v_i.

Señalar además que $d_i = |\delta^*(\{i\})|$.

Por otra parte recordemos que, si G es par, el PCCM puede resolverse en tiempo polinomial de forma exacta. La forma de proceder, descrita por Minieka [38], es obtener un grafo par y simétrico para poder aplicar el Teorema (4), ya que si G es par y simétrico, también es equilibrado:

Algoritmo para PCCM par:

1. Dado un grafo mixto fuertemente conexo y par, $G = (V, E, A)$, sea A_1 el conjunto de arcos obtenidos por asignar arbitrariamente una dirección a las aristas en E y con los mismos costes. Calculamos $s_i = d_i^- - d_i^+$ para cada vértice v_i en el grafo dirigido $(V, A \cup A_1)$. Un vértice v_i con $s_i > 0$ se considerará una fuente con ofertas s_i. Por el contrario, un vértice v_i con $s_i < 0$ se considerará un sumidero con demandas $-s_i$. Además, si G es un grafo par, todos los suministros y demandas son números pares (el cero se considera también par).

2. Sea A_2 el conjunto de arcos en dirección contraria a los de A_1, y con los costes de las aristas correspondientes, y sea A_3 el conjunto de arcos paralelos a los de A_2 de coste 0.

3. Para satisfacer las demandas s_i de todos los vértices, resolvemos el problema de flujos de mínimo coste en el grafo $(V, A \cup A_1 \cup A_2 \cup A_3)$, en los que cada arco en $A \cup A_1 \cup A_2$ tiene capacidad infinita y cada arco en A_3 tiene capacidad 2. Sea x_{ij} el flujo óptimo.

4. Para cada arco (i, j) en A_3 hacemos:

   - Si $x_{ij} = 2$, entonces orientamos la arista correspondiente en G desde v_i hasta v_j (la dirección, desde v_j hasta v_i, asignada a la arista asociada en el paso 1 es "incorrecta")
   - Si $x_{ij} = 0$,entonces orientamos la arista correspondiente en G desde v_j hasta v_i (en este caso, la orientación dada en el paso 1 es "correcta")
   - El caso $x_{ij} = 1$ es imposible, ya que todos los valores de los flujos por los arcos en A_3 son números pares.

5. Calculamos el aumento de G resultante de añadir x_{ij} copias de cada arco en $A \cup A_1 \cup A_2 \cup A_3$. El grafo resultante es simétrico y par.

3.2.1. Algoritmos heurísticos

Cuando el grafo mixto $G = (V, E, A)$ no es par, entonces, al no poder resolver el problema de forma exacta, el algoritmo anterior es la base para dos heurísticas. Estas van a proporcionar una solución entre todas las posibles, pero no existe garantía de que la encontrada sea la mejor. El *primer algoritmo para grafos mixtos* sería el equivalente a la primera transformación de G en un grafo par (posteriormente sería necesario aplicar el *Algoritmo para PCCM par*). Este tiene la siguiente forma:

Primer algoritmo para grafos mixtos:

1. **Transformamos el grafo en uno de grado par:** Sea $G = (V, E, A)$ un grafo mixto fuertemente conexo. Ignoramos las direcciones de los arcos

y resolvemos el emparejamiento de coste mínimo de los vértices de grado impar, y hallamos el aumento de G añadiendo todos los ejes usados para la solución del emparejamiento. El grafo resultante $G' = (V, E', A')$ es par.

2. **Obtenemos un grafo simétrico, sin garantía de paridad:** Con suministros y demandas calculados en el grafo (V, A'), resolvemos el problema de flujos de mínimo coste en G' para obtener un grafo simétrico $G'' = (V, E'', A'')$. Señalar que después de este paso, algunos vértices en G'' puede que tengan grado impar.

3. **Conseguimos paridad y simetría:** Sea V_O el conjunto de nodos de grado impar en G''. Identificamos ciclos que consisten en alternar caminos en $A'' \setminus A$ y E'', con cada camino empezando y terminando en vértices de V_O. Los caminos en $A'' \setminus A$ estarán formados sin considerar las direcciones de los arcos. Como los ciclos están cubiertos, sus arcos están duplicados o borrados, y sus aristas son dirigidas, así que el grafo resultante es par, a la vez que mantenemos la simetría para cada vértice.

El paso 2 anterior muestra que hay una solución de mínimo coste en el problema de flujos. Este preserva la propiedad de que cada vértice tenga grado par, lo que prueba la existencia de tal solución.

El segundo algoritmo para grafos mixtos puede considerarse la versión inversa del primero.

Segundo algoritmo para grafos mixtos:

1. **Transformamos el grafo en uno simétrico:** Resolvemos el problema de flujos de coste mínimo en G, con suministros y demandas calculadas en el grafo (V, A), para obtener un grafo simétrico (V, E', A').

2. **Conseguimos simetría y paridad:** Resolvemos el PCC (no dirigido) en cada componente conexa del subgrafo (V, E') para obtener finalmente un grafo par y simétrico.

Posteriormente, Raghavachari y Veerasamy [43] propusieron un algoritmo mejorado que consistía en lo siguiente:

Algoritmo modificado para grafos mixtos:

- Dado $G = (V, E, A)$, resolvemos el problema de flujos de coste mínimo en G con suministros y demandas calculadas en el grafo (V, A). El grafo resultante es (V, E', A'), donde $E' \subseteq E$ son las aristas de G que no tienen orientación (usadas) en la solución de flujos, y $A' \supseteq A$ son los arcos que satisfacen que el grado de salida es igual al de entrada en cada vértice.
- Antes de ejecutar el Paso 1 del primer algoritmo, reseteamos los costes de todos los arcos y aristas en A' a 0, obligando a que se dupliquen las aristas y arcos en A' cuando sea posible.
- Usamos los costes originales para el resto del primer algoritmo.
- No hay cambios en el segundo algoritmo.

Este algoritmo funciona bien en la práctica, especialmente en grafos con un porcentaje medio-alto de arcos, como probaron Corberán, Martí, y Sanchis [7]. Resultados presentados en [7] nos llevan a afirmar que el algoritmo modificado es significantemente mejor que el primer algoritmo original, excepto en casos con un bajo porcentaje de arcos. Además, su actuación mejora cuando el porcentaje de arcos aumenta, mientras que en el primer algoritmo original, la calidad de la solución va deteriorándose. Por otro lado, el segundo algoritmo presenta los peores resultados, excepto para los casos en los que tengamos un gran número de arcos.

3.2.2. Formulación del Problema

El PCCM, así como otros problemas de rutas por arcos definidos en grafos mixtos, puede formularse de dos formas, dependiendo de si solo se usan una o dos variables asociadas con cada arista del grafo. En el primer caso, la variable x_a asociada con la arista a representa el número de veces que esta arista es atravesada (en cualquier dirección) por la solución. En el segundo caso, las variables representan el número de veces que se atraviesa en cada dirección. Ambas formulaciones han sido muy usadas.

Formulación F1

Está basada en la condición necesaria y suficiente dada por el Teorema 4. Sea x_a, variable que representa el número de veces que una arista $a \in E \cup A$ aparece

en la ruta. Para un subconjunto de aristas F, definimos:

$$x(F) = \sum_{a \in F} x_a$$

Por tanto, el PCCM puede formularse de la siguiente forma:

$$\text{(PCCM-F1)} \quad min \quad \sum_{a \in E} c_a x_a + \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{3.3}$$

$$s.t. \qquad x(\delta^*(i)) \equiv 0 \bmod 2, \quad \forall v_i \in V, \tag{3.4}$$

$$x(\delta^+(S)) - x(\delta^-(S)) \leq x(\delta(S)), \quad \forall S \subset V, \tag{3.5}$$

$$x_a \geq 1 \text{ y entero}, \quad \forall a \in E, \tag{3.6}$$

donde (3.4) asegura que el grafo resultante será par, (3.5) garantiza que las condiciones de que los conjuntos estén equilibrados se cumplan y (3.6) implica que todas las aristas están en la solución.

Señalar además que las restricciones (3.4) no son lineales, y no es posible una expresión lineal de ellas sin usar variables enteras. El efecto de (3.4) puede conseguirse parcialmente mediante las siguientes desigualdades de corte impar (ver Corberán, Romero y Sanchis [11]):

$$x(\delta^*(S)) \geq |\delta^*(S)| + 1 \qquad \forall S \subset V \text{ con } |\delta^*(S)| \text{ impar}. \tag{3.7}$$

Sin embargo, añadir estas desigualdades no nos garantiza la paridad de la solución, mientras que con (3.4) se garantiza la condición.

Las desigualdades (3.5) y (3.7), aunque sean un número exponencial de desigualdades, pueden separarse en tiempo polinomial. Padberg y Rao [41] probaron que las desigualdades de corte impar violadas pueden identificarse en tiempo polinomial.

Consideramos ahora un conjunto $S \subset V$ tal que $\delta(S) = \emptyset$. Entonces, la condición del conjunto equilibrado (3.6) correspondiente con S y con $V \setminus S$ implica la denominada *ecuación de simetría* $x(\delta^+(S)) = x(\delta^-(S))$. Por tanto, la formulación anterior incluye una ecuación asociada con cada conjunto $S \subset V$ con $\delta(S) = \emptyset$. Aunque la mayoría de estas ecuaciones serán linealmente dependientes, si consideramos q subgrafos de G inducidos por las aristas, puede demostrarse que cualquiera $q - 1$ de las ecuaciones de simetría correspondientes son linealmente independientes. En

Corberán, Romero, y Sanchis [11], estas ecuaciones,

$$x(\delta^+(K_i)) = x(\delta^-(K_i)),\ i = 1, \ldots, q, \tag{3.8}$$

se conocen como las *ecuaciones del sistema*, donde $K_1, \ldots, K_q$ denota el conjunto de vértices de las componentes conexas del grafo (V, E).

Sea $\text{PCCM}_{F1}(G)$ la envolvente convexa de todos los vectores $x \in \mathbb{R}^{E \cup A}$ satisfaciendo de (3.4) a (3.6). Por tanto, $\text{PCCM}_{F1}(G)$ es un poliedro no acotado, se cumple que $\dim(\text{PCCM}_{F1}(G)) = |E \cup A| - q + 1$, y las siguientes desigualdades inducen facetas para $\text{PCCM}_{F1}(G)$:

1. Desigualdades triviales: $x_a \geq 1$, $\forall a \in E \cup A$,
2. Desigualdades de conjuntos equilibrados (3.5),
3. Desigualdades de corte impar (3.7),
4. Desigualdades de zigzag impar (descritas en [10]).

Formulación F2

La formulación F2 usa dos variables x_{ij} y x_{ji} asociadas con cada arista $a = (i, j)$, representando el número de veces que la arista a es atravesada en la dirección correspondiente. Como hemos señalado anteriormente, esta formulación está basada en las condiciones suficientes para que un grafo mixto sea Euleriano, y tiene la siguiente forma:

$$\text{(PCCM-F2)} \quad min \sum_{a=(i,j)\in E} c_a(x_{ij} + x_{ji}) + \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{3.9}$$

$$s.t. \quad x_{ij} + x_{ji} \geq 1, \quad \forall a = (i, j) \in E, \tag{3.10}$$

$$x(\delta^+(S)) - x(\delta^-(S)) + \sum_{(i,j)\in\delta(i)}(x_{ij} + x_{ji}) = 0, \quad \forall v_i \in V, \forall S \tag{3.11}$$

$$x_{ij} \geq 1 \text{ y entero}, \quad \forall (i, j) \in A, \tag{3.12}$$

$$x_{ij}, x_{ji} \geq 0 \text{ y entero}, \quad \forall a = (i, j) \in E, \tag{3.13}$$

donde (3.10) y (3.12) implican que todas las aristas y arcos están en la solución,

y (3.11) implica que se cumplen las condiciones de simetría. Señalar que las condiciones de que las variables sean enteras y las de simetría implican que el grafo resultante será par.

Además, se ha demostrado que las coordenadas de los puntos extremos del poliedro asociado a la relajación lineal de F2 toman valores de 0.5 o enteros no negativos, y que los valores fraccionarios ocurren solamente en algunas aristas $a = (i, j)$, donde $x_{ij} = x_{ji} = 0{,}5$. Así como con PCW, este resultado podría ser la base de un algoritmo heurístico simple que consiste en redondear las variables de valores 0.5 a 1.

Sea $\text{PCCM}_{F2}(G)$ la envolvente convexa de todos los vectores $x \in \mathbb{R}^{E \cup A}$ satisfaciendo de (3.10) a (3.13). $\text{PCCM}_{F2}(G)$ es un poliedro no acotado, con dimensión $2|E| + |A| - |V| + 1$, y tal que las siguientes desigualdades inducen facetas:

- Desigualdades triviales: $x_{ij} \geq 1 \ \forall (i, j) \in A$, y $x_{ij}, x_{ji} \geq 0 \ \forall a = (i, j) \in E$,
- Desigualdades de paso (por una arista) (3.10).
- Desigualdades de corte impar (3.6) .
- Desigualdades k-rueda propuestas por Win [48].
- Desigualdades de zigzag impar [10].

Comparación de las formulaciones

Es fácil ver que las formulaciones (enteras) F1 y F2 son equivalentes, pero ¿qué podemos decir acerca de sus relajaciones en programación lineal? Recordemos que las restricciones (3.4) no son lineales, y que no es posible una expresión de ellas sin usar variables enteras. Por tanto, estas restricciones no pueden ser eliminadas, y las desigualdades de corte impar (3.7) se añadirán a F1.

Por otro lado, cabe señalar que una vez que hayamos eliminado las condiciones de que las variables sean enteras de F2, las condiciones de simetría solas no nos garantizan la paridad de los vértices. Por lo que las restricciones (3.13) también se añadirán a F2.

Por consiguiente, denominaremos a la relajación lineal de F1 reforzada con (3.7),

$$
\begin{aligned}
x_{ij} &\geq 1, \quad \forall (i,j) \in A, \\
x_a &\geq 1, \quad \forall a \in E, \\
x(\delta^+(S)) - x(\delta^-(S)) &\leq x(\delta(S)), \quad \forall S \subset V, \qquad (3.5) \\
x(\delta^*(S)) &\geq |\delta^*(S)| + 1, \quad \forall S \subset V \text{ con } |\delta^*(S)| \text{ impar}, \quad (3.7)
\end{aligned}
$$

e igualmente la relajación lineal de F2 reforzada con (3.7),

$$
\begin{aligned}
x_{ij} &\geq 1, \quad \forall (i,j) \in A, \\
x_{ij}, x_{ji} &\geq 0, \quad \forall (i,j) \in E, \\
x_{ij} + x_{ji} &\geq 1, \quad \forall (i,j) \in E, \qquad (3.10) \\
x(\delta^*(S)) &\geq |\delta^*(S)| + 1, \quad \forall S \subset V \text{ con } |\delta^*(S)| \text{ impar}, \quad (3.7) \\
x(\delta^+(S)) - x(\delta^-(S)) + \textstyle\sum_{(i,j)\in\delta(i)}(x_{ij} - x_{ji}) &= 0, \quad \forall v_i \in V, \qquad (3.11)
\end{aligned}
$$

Como las restricciones (3.5) y (3.7) en PL_{F1} y (3.7) en PL_{F2} son exponenciales, solo un subconjunto de ellas pueden ser añadidas de forma explícita a PL para ser resuelto. Sin embargo, dichas desigualdades pueden separarse en tiempo polinomial. Se ha probado, además, en [8] que ambas relajaciones producen la misma cota.

Existen varios procedimientos exactos propuestos para hallar la solución óptima del PCCM, entre ellos el algoritmo de Ramificación y Acotación de Christofides, [6].

3.3. Problema del Cartero windy

El PCW puede definirse de la siguiente manera. Dado un grafo no dirigido y conexo $G = (V, E)$ con dos costes no negativos asociados con cada arista $(i, j) \in E$ tal que:

- c_{ij} se corresponde al coste de atravesarla desde v_i hasta v_j
- c_{ji} se corresponde al coste de atravesarla desde v_j hasta v_i

El PCW consiste en encontrar una ruta de coste mínimo en G que pase por cada arista una vez. Fue propuesto por Minieka [39], quien le puso tal nombre debido a que el coste de ir de un punto a otro puede depender de si el viento sopla a favor o en contra. El PCW contiene el PCCM como caso especial cuando, para cada arista $(i,j) \in E$, o $c_{ij} = c_{ji}$, o $max\{c_{ij}, c_{ji}\} = \infty$. Por tanto, es fácil ver que PCW es NP-duro, aunque puede resolverse en tiempo polinomial si G es par [**?**], si el coste de los dos sentidos opuestos de cada ciclo en G es el mismo [29], o si G es un grafo serie paralelo [53].

Si G satisface que el coste en los dos posibles sentidos de cada uno de los ciclos es el mismo, el PCW puede resolverse como el PCC no dirigido donde cada arista (i,j) tiene coste $(c_{ij} + c_{ji})/2$. Vamos a describir a continuación el algoritmo propuesto por Win [**?**] para encontrar una ruta en un grafo par (y Euleriano) $G = (V, E)$. Este será la base para desarrollar otros algoritmos para grafos generales.

Algoritmo de Win para grafos Eulerianos:

1. Para cada arista $(i,j) \in E$, si $c_{ij} \leq c_{ji}$, entonces orientamos (i,j) desde v_j hacia v_i. De esta forma obtenemos un grafo dirigido $D_G = (V, A)$.

2. Construimos un grafo dirigido $D' = (V, A')$ con conjunto de arcos A', costes y capacidades de arcos, y demanda en los nodos definida de la siguiente manera:

   - Para cada arco $(i,j) \in A$, hay tres arcos en A': una copia de (i,j) con coste c_{ij} y capacidad infinita, una copia de (j,i), denotada por $(j,i)'$, con coste $(c_{ij} - c_{ji})/2$ y capacidad dos (estos arcos se llaman arcos *artificiales*).
   - Para cada nodo $v_i \in V$, la demanda es $d_i = d^+(i) - d^-(i)$ en D_G (las demandas negativas son interpretadas como suministros).

3. Encontramos un flujo de coste mínimo en D' satisfaciendo las demandas y suministros de los vértices.

4. Denotamos por y^*_{ij}, y^*_{ji} y $y^*_{(ji)'}$ a los valores del flujo en los arcos (i,j), (j,i), y $(j,i)'$, respectivamente. Para cada arco $(i,j), (j,i), (j,i)' \in A'$, hacemos lo siguiente:

- Si $y^*_{(ji)'} = 0$, entonces ponemos $y^*_{ij} + 1$ copias de (i, j) (y ninguna copia de (j, i)) en A''
- Si $y^*_{(ji)'} = 2$, ponemos $y^*_{ji} + 1$ copias de (j, i) (y ninguna copia de (i, j)) en A''.

$D'' = (V, A'')$ es Euleriano, y cualquier camino Euleriano dirigido en D'' es una ruta óptima del PCW en G.

3.3.1. Algoritmos heurísticos

Al igual que ocurría en el PCCM, el algoritmo anterior para grafos pares es la base para un procedimiento heurístico para el caso general (Win [**?**]). Consiste en transformar primero el grafo original en uno par, y después aplicar el algoritmo anterior para grafos Eulerianos.

Algoritmo de Win para grafos generales:

1. Transformamos G en un grafo Euleriano.
   - Para cada arista $a = (i, j) \in E$, definimos $\overline{c}_e = (c_{ij} + c_{ji})/2$.
   - Identificamos los nodos impares de G, y calculamos las distancias del camino más corto entre cada par de ellos usando las distancias $\overline{c}_e$.
   - Encontramos un emparejamiento perfecto de coste mínimo de los nodos impares.
   - Añadimos a E una copia adicional de cada arista que está en el camino más corto uniendo dos nodos impares emparejados (!!!!!!!). El grafo resultante $G' = (V, E')$, es Euleriano.
2. Aplicamos el algoritmo de Win a los grafos Eulerianos de G' con la función de coste original c_e.

Este algoritmo tiene una cota para el peor de los casos de dos veces el valor del óptimo. Win [48] propuso otra heurística que implicaba el redondeo de las soluciones óptimas de la relajación lineal de la formulación del PCW que veremos a continuación. Este algoritmo está basado en el siguiente resultado: Los valores de las variables en cualquier solución óptima PL son enteros o 0.5, y $x_{ij} = 0{,}5$ si y solo si $x_{ji} = 0{,}5$ ([33], [48]). Entonces, si imponemos que $x_{ij} = x_{ji} = 1$ podemos demostrar que es posible alcanzar una solución factible del PCW.

3.3.2. Formulación del problema

Sea $G = (V, E)$ el grafo donde hemos hallado la ruta de coste mínimo para el PCW. Para $S_1, S_2 \subseteq V$, $(S_1 : S_2)$ denota el conjunto de aristas con vértices uno en S_1, y el otro en S_2. Dado $S \subseteq V$, consideramos:

- $\delta(S) = (S : V \setminus S)$
- $E(S) = (S : S)$

Sea x_{ij} el número de veces que una arista (i, j) es atravesada desde v_i hacia v_j en una ruta del PCW. Definimos así:

- Para cada $F \subseteq E$, consideramos $x(F) = \sum_{(i,j)\in F}(x_{ij} + x_{ji})$
- Para $S_1, S_2 \subset V$, consideramos:
  - $x(S_1, S_2) = \sum_{v_i \in S_1, v_j \in S_2} x_{ij}$
  - $x(S_1 : S_2) = x(S_1, S_2) + x(S_2, S_1)$

La formulación PLE (Programación Lineal y Entera) del PCW es la siguiente [48], [28]:

$$\text{(PCW)} \quad min \quad \sum_{(i,j)\in E}(c_{ij}x_{ij} + c_{ji}x_{ji}) \tag{3.14}$$

$$s.t. \quad x_{ij} + x_{ji} \geq 1, \quad \forall (i,j) \in E, \tag{3.15}$$

$$\sum_{(i,j)\in\delta(i)}(x_{ij} - x_{ji}) = 0, \quad \forall v_i \in V, \tag{3.16}$$

$$x_{ij}, x_{ji} \geq 0 \text{ y entero}, \quad \forall (i,j) \in E, \tag{3.17}$$

donde las condiciones (3.15) implican que cada arista será atravesada al menos una vez, y las condiciones (3.16) obligan a que el grafo (dirigido) representado por la ruta sea simétrico. Además, el sistema anterior incluye una ecuación asociada a cada vértice. Nos referiremos así a estas $|V|$ ecuaciones (3.16) como *ecuaciones de simetría*, y cualquier conjunto que tomemos con $|V| - 1$ de ellas son linealmente independientes.

Sea PCW$(G) \subseteq \mathbb{Z}^{2|E|}$ la envolvente convexa de los vectores que satisfacen las condiciones (3.15) a la (3.17). PCW(G) es un poliedro no acotado con dimensión $2|E| - |V| + 1$, y está demostrado que las desigualdades triviales inducen facetas ([27],[48]).

Además de estas desigualdades, cabe señalar dos familias de desigualdades más que no están presentes en la formulación anterior pero de gran importancia:

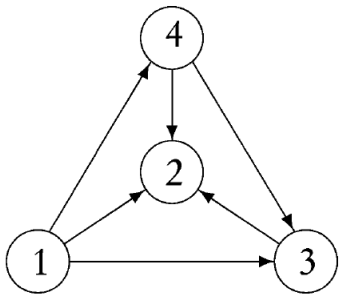
- **Desigualdades de corte impar**

  Inducen facetas para el PCW(G) y se basan en el hecho de que cualquier arista de un conjunto de corte debe ser atravesada un número par de veces. Tienen la forma siguiente:

  $$x(\delta(S)) \geq |\delta(S)| + 1, \forall S \subset V, |\delta(S)| \text{ impar.}$$

- **Desigualdades k-rueda**

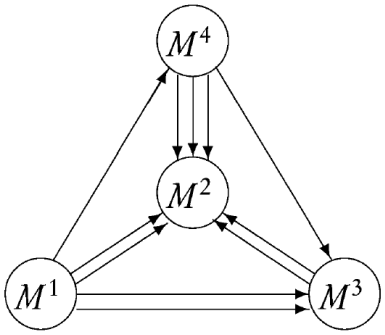
  Supongamos una rueda formada por un ciclo (la llanta), con cada nodo incidente con un nodo común (el centro). A las aristas que unen los nodos de la llanta con el centro las llamaremos radios. Así, una k-rueda (denotada como W_k), es una rueda con una llanta de k nodos. Por ejemplo, una 3-rueda tendría la forma siguiente:

  Y una desigualdad 3-rueda sería de la forma:

  $$x_{12} + x_{13} + x_{14} + x_{32} + x_{42} + x_{43} \geq 3$$

  la cual es válida e induce facetas para el poliedro de PCW correspondiente. Además, es generalizable reemplazando cada vértice por un subgrafo conexo y cada arco por un subconjunto de arcos paralelos con la misma dirección:

Consideramos así una partición del conjunto de vértices V en cuatro subconjuntos, M^1, M^2, M^3 y M^4, donde cada M^i contiene un número impar de vértices impares, y cada subgrafo $G(M^i)$ está conectado. Consideramos también $x(A')$ como el conjunto de variables correspondientes al paso por las aristas en la dirección que aparece en el gráfico anterior. La desigualdad 3-rueda es cuando:

$$x(A') \geq \frac{1}{2}(|(M^1 : M^3)| + |(M^2 : M^4)| + |(M^1 : M^4)| + |(M^2 : M^3)|) + 1$$

Las desigualdades 3-rueda pueden extenderse a las k-rueda, mucho más generales, que contienen un número impar k de nodos en la llanta [48].

Señalar finalmente acerca de la existencia de algoritmos exactos, que en Corberán et al. [9] se presenta el algoritmo de ramificación y acotación para el PCW. Este usa la heurística y procedimientos exactos para separar las desigualdades de corte impar violadas e incorporar los algoritmos de separación para todas las familias de desigualdades de zigzag. Además, como está demostrado que todas las desigualdades de zigzag son desigualdades de Chvátal-Gomory de rango a lo sumo 2, se añade un algoritmo en tiempo polinomial para identificar las desigualdades de módulo k violadas. Básicamente, el método consiste en resolver un sistema de congruencias donde el sistema está compuesto de todas las desigualdades obligatorias del último PL más una desigualdad para cada ecuación de simetría.

Una aproximación completamente diferente a la solución del PCW fue el propuesto por Yan y Thompson [50], quienes modificaron la formulación del PCW para obtener un problema de cobertura de conjuntos. Teniendo en cuenta esta transformación, se propone un algoritmo exacto y un algoritmo heurístico que produce buenos resultados para casos con hasta 90 vértices y 180 aristas. Ambos métodos están basados en la ramificación y acotación, previamente diseñados para dar solución a problemas de cubrimiento de conjuntos, división y problemas *packing*.

4

Aplicaciones

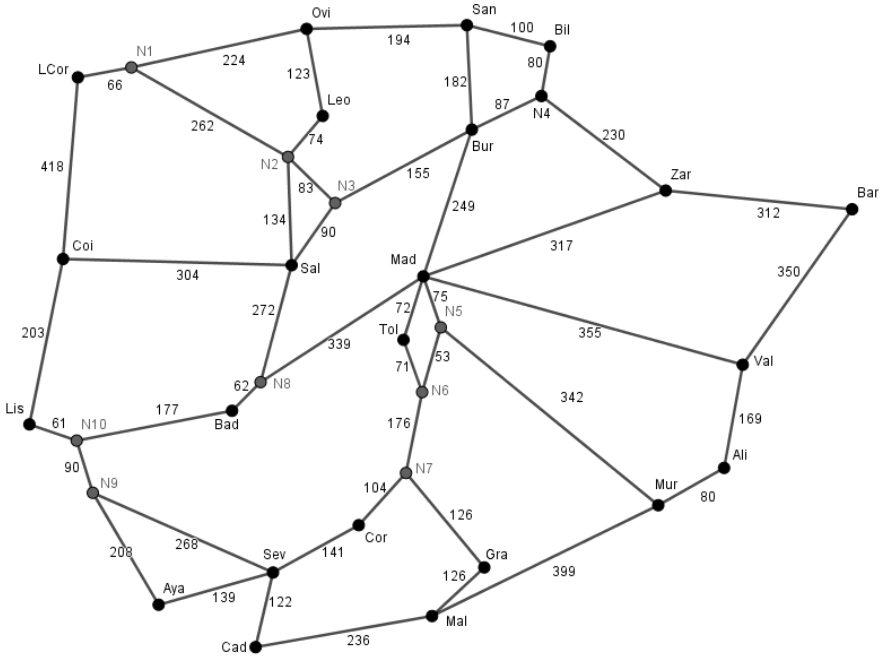
En este capítulo, nos centraremos en ciertos casos para los cuales aplicaremos los resultados vistos anteriormente para el PCC no dirigido, el PCC dirigido, el PCC mixto y el Problema del Cartero windy. Para la obtención de soluciones utilizaremos un programa en Xpress, en el que iremos introduciendo los datos respectivos a cada caso.

4.1. Ejemplo del Problema del Cartero Chino no dirigido

Consideremos la siguiente red de carreteras de la Península Ibérica:

En el mapa anterior podemos visualizar las carreteras que conformarán las aristas de nuestro grafo en azul, y los vértices de este representados como puntos. Estos son negros sin representan una ciudad, o grises si representan un mero cruce de carreteras.

Centrémonos pues en el grafo resultante:



donde las ponderaciones de la arista (i, j) representa la distancia del punto i al j (en kilómetros).

Supongamos que nos interesa recorrer todas estas carreteras (con la finalidad de realizar un análisis de calidad de éstas, por ejemplo), de forma que minimicemos los kilómetros recorridos por razones obvias. Introducimos los datos correspondientes en el programa realizado en Xpress. Este está diseñado para grafos dirigidos, lo cual no supone un problema, pues para grafos no dirigidos basta con introducir cada arista dos veces, una en un orden (i, j) y otra en el orden contrario (j, i). De esta manera, el resultado obtenido es el siguiente:

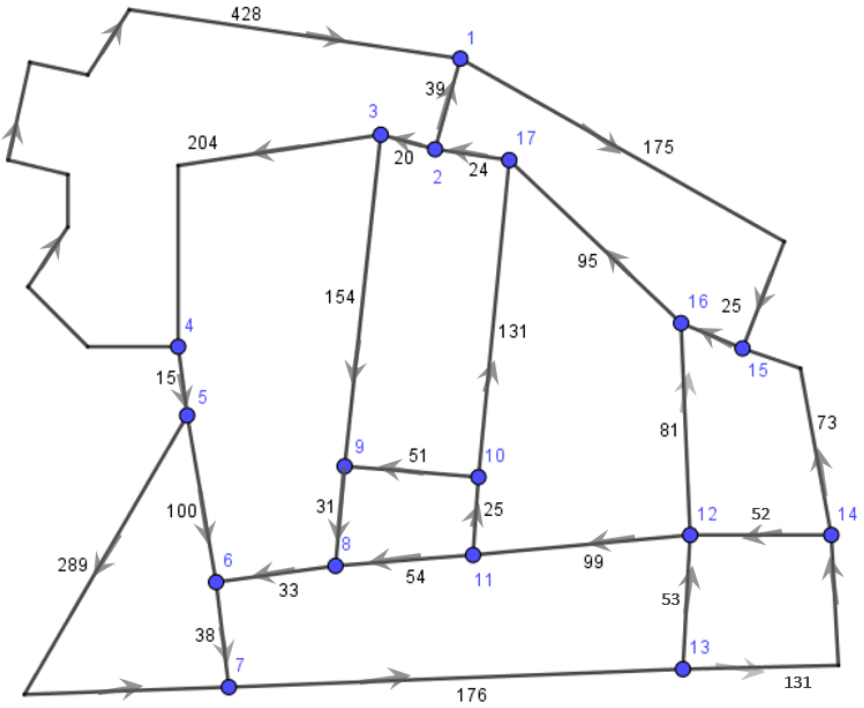
- La longitud total del camino es: 17002 kilómetros
- La ruta a seguir seria:
  [”$Murcia$”,”$N5$”,”$Madrid$”,”$Zaragoza$”,”$Barcelona$”,”$Valencia$”,”$Alicante$”,
  ”$Valencia$”,”$Barcelona$”,”$Zaragoza$”,”$N4$”,”$Bilbao$”,”$Santander$”,”$Oviedo$”,

"$N1$","$N2$","$Leon$","$N2$","$Salamanca$","$N2$","$N3$","$N2$","$N1$","$La\ Coruña$",
"$Coimbra$","$Lisboa$","$N10$","$N9$","$Sevilla$","$Cadiz$","$Malaga$","$Granada$",
"$Malaga$","$Cadiz$","$Sevilla$","$Cordoba$","$N7$","$N6$","$Toledo$","$N6$","$N7$",
"$Granada$","$N7$","$Cordoba$","$Sevilla$","$Ayamonte$","$Sevilla$","$N9$",
"$Ayamonte$","$N9$","$N10$","$Badajoz$","$N8$","$Badajoz$","$N10$","$Lisboa$",
"$Coimbra$","$La\ Coruña$","$N1$","$Oviedo$","$Leon$","$Oviedo$","$Santander$"
,"$Bilbao$","$N4$","$Burgos$","$Santander$","$Burgos$","$N4$","$Zaragoza$",
"$Madrid$","$Toledo$","$Madrid$","$N8$","$Salamanca$","$N3$","$Salamanca$",
"$Coimbra$","$Salamanca$","$N8$","$Madrid$","$Valencia$","$Madrid$","$Burgos$",
"$N3$","$Burgos$","$Madrid$","$N5$","$N6$","$N5$","$Murcia$",
"$Alicante$","$Murcia$","$Malaga$","$Murcia$"]

4.2. Ejemplo del Problema del Cartero Chino dirigido

Consideremos las calles de Sevilla señaladas en color naranja. Todas tienen en común que son de un único sentido, representado con flechas.

Supongamos pues que es de nuestro interés recorrérnoslas (para repartir cartas, por ejemplo, siguiendo el caso original sobre el que se definió el PCCD), así como encontrar la ruta más corta para hacerlo. Consideramos pues el grafo que representa nuestro problema, donde los vértices representan las intersecciones entre calles y las aristas están ponderadas con la longitud de la calle correspondiente (en metros).



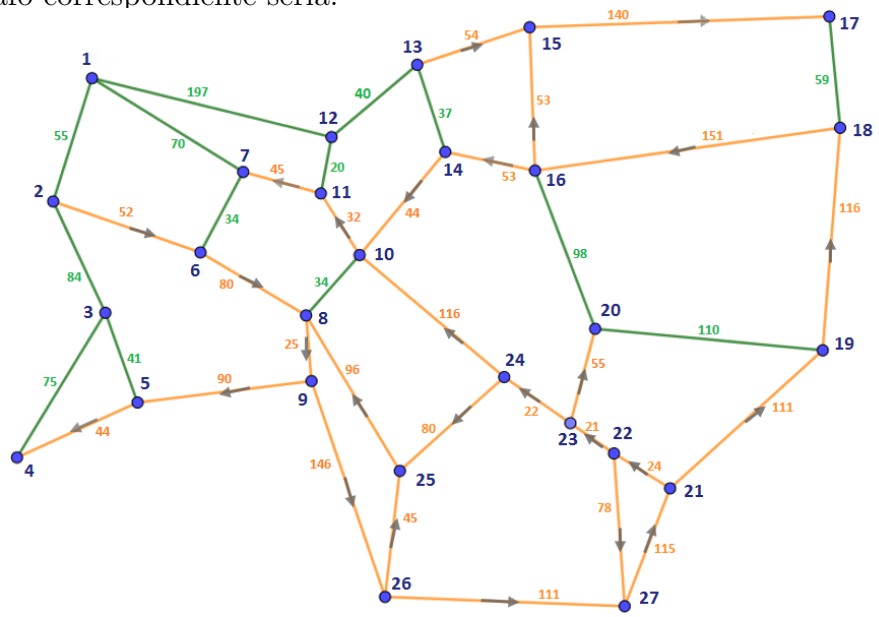
La solución que obtenemos al introducir nuestros datos en el programa es la siguiente:

- La longitud total del camino es: 5522 metros.
- La ruta a seguir seria:
  $[1, 15, 16, 17, 2, 3, 9, 10, 17, 2, 3, 4, 5, 6, 7, 13, 14, 12, 13, 14, 15, 16, 17, 2, 3, 4, 5, 7, 13, 14, 15, 16, 17, 2, 3, 9, 8, 6, 7, 13, 14, 15, 16, 12, 11, 8, 6, 7, 13, 14, 12, 11, 10, 17, 2, 1, 15, 16, 17, 2, 3, 4, 1]$

4.3. Ejemplo del Problema del Cartero Chino mixto

En esta sección veremos la aplicación del PCCM a un caso real en el pueblo de Hinojosa del Duque (Córdoba). Serán de nuestro interés las siguientes calles:

El grafo correspondiente sería:



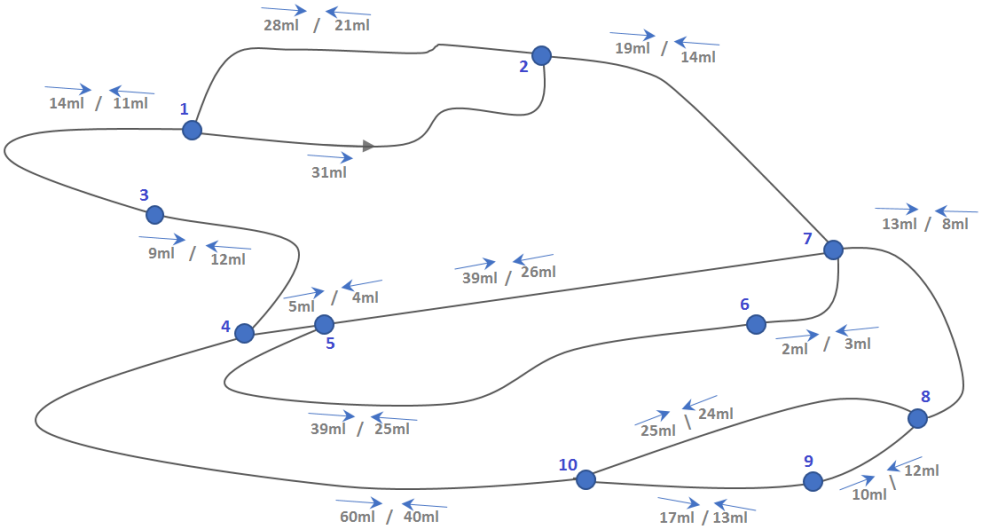
Al igual que en el ejemplo anterior, las calles de un único sentido aparecen representadas en naranja (tanto en el mapa como en el grafo), mientras que las de doble sentido están en verde. Además, dichos sentidos están representados mediante flechas, y la ponderación de cada arista o arco corresponde con la longitud de la calle respectiva. Buscamos a continuación recorrernos todas estas calles (repartiendo cartas, por ejemplo) de forma de describamos una ruta de longitud mínima. Al introducir los datos en nuestro, programa (donde las aristas no dirigidas se introducirán igual que en el ejemplo de la sección 4.1), obtenemos la siguiente solución:

- La longitud total del camino es: 6934 metros.
- La ruta a seguir seria:
  $[1, 2, 3, 4, 3, 5, 4, 3, 2, 1, 7, 6, 8, 9, 26, 25, 8, 9, 26, 27, 21, 22, 27, 21, 19, 20, 16, 14, 13, 12, 11, 12, 13, 14, 13, 15, 17, 18, 17, 18, 16, 15, 17, 18, 16, 20, 19, 18, 16, 14, 10, 8, 9, 26, 27, 21, 22, 23, 20, 16, 14, 10, 8, 9, 26, 27, 21, 22, 23, 24, 10, 8, 9, 26, 27, 21, 22, 23, 24, 25, 8, 10, 11, 7, 1, 7, 6, 7, 6, 8, 9, 5, 3, 2, 1, 12, 1]$

4.4. Ejemplo del Problema del Cartero windy

Para ver un caso concreto y real de este problema, nos centraremos en la pedanía de Cumbres Verdes (Granada):

Éste destaca por sus numerosas y notables cuestas. Ahora, las calles que nos interesa recorrer son las señaladas en naranja, y buscamos la ruta en la que gastemos la mínima cantidad de combustible, de forma que pasemos por todas las calles al menos una vez. Para ello, vamos a considerar que el vehículo del que disponemos para realizar tal tarea gasta 7 litros de combustible cada 100 kilómetros que recorre en llano, y $7(1+2\alpha)$ litros cada 100 kilómetros con pendiente α, que puede ser negativa o positiva. El grafo a considerar sería el siguiente:



En él, los vértices representan las intersecciones entre las calles y los puntos de una misma calle donde la pendiente pasa de ser positiva a negativa (o viceversa). Las aristas, al igual que en los ejemplos del PCCD y el PCCM, representan las calles, y las ponderaciones de estas son los mililitros de gasolina que se gasta al atravesar la calle entera. Lógicamente, debido a las cuestas anteriormente comentadas, no supondrá el mismo coste atravesar una arista en un sentido que en otro. Ambos costes aparecen también en nuestro gráfico. Señalar por último que la arista (12) es la única con sentido único.

La solución que obtenemos al introducir nuestros datos en el programa es la siguiente:

- La longitud total del camino es: 496 mililitros.

- La ruta a seguir sería:
  $[1, 2, 7, 5, 4, 10, 8, 9, 8, 10, 9, 10, 4, 3, 4, 5, 6, 5, 7, 6, 7, 8, 7, 2, 1, 3, 1]$

Conclusiones

Como conclusión general del trabajo, podemos decir que los problemas de rutas por arcos son de gran importancia en la vida real. Vivimos en un mundo en el que se busca continuamente la optimización de los recursos, así como del tiempo empleado a la hora de realizar cualquier tarea. Esto es crucial especialmente en el ámbito de la logística y el transporte, cada vez más desarrollado para satisfacer mayor volumen de demanda en el menor tiempo posible. La manera de obtener mayores ganancias es aplicar ciertas técnicas de la investigación operativa, y aquí es donde entra en juego lo que hemos estudiado en este trabajo, usando herramientas de Teoría de Grafos y diversos modelos de Programación Matemática.

Si nos centramos únicamente en el problema de recorrer una red de calles o carreteras eficientemente, es indispensable el uso de los resultados derivados del estudio del Problema del Cartero Chino, donde debemos aplicar una u otra versión dependiendo del tipo de grafo que represente el problema en el que estamos trabajando. Existen así infinidad de variantes, todas ellas con diversas formas propuestas de abordarlas.

A pesar de que el PCC tuvo su origen en la forma de repartir cartas a lo largo de unas determinadas calles, su aplicación va mucho más allá. Ha sido utilizado a la hora de diseñar rutas de autobuses y camiones de mercancías, ha ayudado a reducir notablemente el coste de la recogida de basuras, de limpieza de calles y de lectura de contadores. Ha mejorado la retirada de nieve en carreteras y calles, y ha sido indispensable en el diseño de las rutas de tren y metro. Incluso se ha utilizado para minimizar el coste total de una grúa, ya que esta debe, a partir de

una posición de partida, realizar una serie de movimientos y volver a su posición inicial. Lo mismo ocurre con los movimientos de ciertas máquinas de producción en serie.

Por otro lado es imprescindible considerar el problema windy en las trayectorias de aviones y barcos, donde la intensidad y la dirección del viento son determinantes a la hora de optimizar el combustible.

Por los motivos precedentes, los problemas abordados en este trabajo no solo resultan muy interesantes, sino que tienen una importancia indiscutible tanto en el mundo empresarial e industrial como en la vida cotidiana.

Bibliografía

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and Aproximation: Combinatorial Optimization Problems and Their Approximability Properties*, Springer, New York, 1999.

[2] R. Bellman y K. L. Cooke, *The Konigsberg bridges problem generalized*, Jl. of Math. Anal. and Appl., 25 (1969), p. 1.

[3] E. Benavent y D. Soler, *The directed rural postman problem with turn penalties*, Transportation Science, 33 (1999), pp. 408-418.

[4] R. G. Busacker y T. L. Saaty, *Finite graphs and networks*, McGraw-Hill, New York (1965).

[5] N. Christofides, *The optimum traversal of a graph*, Omega - The Int. Jl. of Management Science, 1 (1973), p. 1.

[6] N. Christofides, E. Benavent, V. Campos, Á. Corberán, y E. Mota, *An optimal method for the mixed postman problem*, in System Modelling and Optimization, P. Thoft-Christensen, ed., Lecture Notes in Control and Information Sciences 59, Springer, Berlin, 1984, pp. 641-649.

[7] Á. Corberán, R. Martí, y J. M. Sanchis, *A GRASP heuristic for the mixed Chinese postman problem*, European Journal of Operational Research, 142 (2002), pp. 70-80.

[8] Á. Corberán, R. Martí, y J. M. Sanchis, *A comparison of two different formulations for arc routing problems on mixed graphs*, Computers and Operations Research, 33 (2006), pp. 3384-3402.

[9] Á. Corberán, M. Oswald, I. Plana, G. Reinelt, y J. M. Sanchis, *New results on the windy postman problem*, Mathematical Programming, 132 (2012), pp. 309-332.

[10] Á. Corberán, I. Plana, y J. M. Sanchis, *Zigzag inegualities: A new class of facet-inducing inequalities for arc routing problems*, Mathematical Programming, 108 (2006), pp. 79-96.

[11] Á. Corberán, A. Romero, y J. M. Sanchis, *The mixed general routing polyhedron*, Mathematical Programming, 96 (2003), pp. 103-137.

[12] E. W. Dijkstra, *A note on two problems in connection with graphs*, Numerische Mathematik, 1 (1959), p. 269.

[13] R.G. Downey y M.R. Fellows, *Fundamentals*, Chinese Mathematics, 1 (1962), pp. 237-277.

[14] M. Dror y M. Haquari, *Generalized Steiner problems and other variants*, Journal of Combinatorial Optimization, 4 (2000), pp. 415-436.

[15] J. Edmonds, *Paths, trees and flowers*, Canadian Journal of Mathematics, 17 (1965), pp. 449-467.

[16] J. Edmonds, *The Chinese postman's problem*, Bulletin of the Operations Research Soc. of America, 13, Supplement 1 (1965), p. B-73.

[17] J. Edmonds, *The Chinese's Postman Problem*, Operations Research Bulletin, 13 (1965), pp. B73-B77.

[18] J. Edmonds y E.L. Johnson, *Matching, Euler tours and the Chinese postman*, Mathematical Programming, 5 (1973), pp. 88-124.

[19] L. Euler, *Solutio problemais ad geometriam situs pertinentis*, Commentarii Academiae Scientiarum Imperialis Petropolitanae, 8 (1736), pp. 128-140.

[20] M.R. Fellows, B.M.P. Jansen, y F.A. Rosamond, *Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity*, European Journal of Combinatorics, 34 (2013), pp. 541-566.

[21] C.G. Fernandes, 0. Lee, y Y. Wakabayashi, *Minimum cycle cover and Chinese postman problems on mixed graphs with bounded tree-width*, Discrete Applied Mathematics, 157 (2009), pp. 272-279.

[22] M. Fleury, *Deux problèmes de geómétrie de situation*, Journal de Mathématiques Élémentaires, 2 (1883), pp. 257-261.

[23] J. Flum and M. Grohe, *Parametrized Complexity Theory*, Springer, New York, 2006.

[24] L.R. Ford y D.R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[25] G.N. Frederickson, *Approximation algorithms for some postman problems*, Journal of the ACM, 26 (1979), pp. 538-554.

[26] B.L. Golden y R.T. Wong, *Capacitated arc routing problems*, Networks, 11 (1981), pp. 305-315.

[27] M. Grötschel y Z. Win, *On the windy postman polyhedron*, Technical report 75, University of Augsburg, Augsburg, Germany, 1988.

[28] M. Grötschel y Z. Win, *A cutting plane algorithm for the windy postman problem*, Mathematical Programming, 55 (1992), pp. 339-358.

[29] M. G. Guan, *On the windy postman problem*, Discrete Applied Mathematics, 9 (1984), pp. 41-46.

[30] M.G. Guan, *Graphic programming using odd or even points*, Chinese Mathematics, 1 (1962), pp. 237-277.

[31] M.G. Guan, *Paths, trees and flowers*, Canadian Journal of Mathematics, 17 (1965), pp. 449-467.

[32] C. Hierholzer, *Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfabren*, Mathematische Annalen, 6 (1873), pp. 30-32.

[33] C. H. Kappauf y G. J. Koehler, *The mixed postman problem*, Discrete Applied Mathematics, 1 (1979), pp. 89-103.

[34] D. König, *Theorie der endlichen und unendlichen Graphen*, Akademische Verlagsgesellschaft, Leipzig, 1936.

[35] G. Laporte y A. Corberán, *Arc Routing: Problems, Methods and Applications*, Society for Industrial and Applied Mathematics, 2014.

[36] T. M. Liebling, *Graphentheorie in Planungs- und Tourenproblemen*, Lecture Notes in Operations Research and Mathematical Systems 21, Springer, Berlin, 1970.

[37] J.B. Listing, *Vorstudien zur Topologie*, Vandenhoeck und Ruprecht, Göttingen, 1847.

[38] E. Minieka, *Optimization Algorithms for Networks and Graphs*, Marcel Dekker, New York, 1979.

[39] E. Minieka, *The Chinese postman problem for mixed networks*, Management Science, 25 (1979), pp. 643-648.

[40] R. Niedermeier, *Invitation to Fixed-Parameter Algorithms*, Oxford University Press, Oxford, UK, 2006.

[41] M. W. Padberg y M. R. Rao, *Odd minimum cut-sets and b-matchings*, Mathematics of Operations Research, 7 (1982), pp. 67-80.

[42] L. Poinsot, *Mémoire sur les polygones et les polyèdres*, Journal de l'École Polytechnique, 4 (1810), pp. 16-49.

[43] B. Raghavachari and J. Veerasamy, *A 3/2-approximation algorithm for the mixed postman problem*, SIAM Journal on Discrete Mathematics, 12 (1999), pp. 425-433.

[44] B. Raghavachari y J. Veerasamy, *Approximation algorithms for the asymmetric postman problem*, in Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '99), SIAM, Philadelphia, 1999, pp. 734-741.

[45] V.V. Vazirani, *Approximation Algorithms*, Springer, New York, 2001.

[46] O. Vleben, *An application of modular equations in analysis situs*, Annals of Mathematics, 2 (1913), pp. 86-94.

[47] D.P. Williamson y D.B. Shmoys, *The Design of Approximation Algorithms*, Cambridge University Press, Cambridge, UK, 2011.

[48] Z. Win, *Constributions to Routing Problems*, Ph.D. thesis, University of Augsburg, Augsburg, Germany, 1987.

[49] Z. Win, *On the windy postman problem on Eulerian graphs*, Mathematical Programming, 44 (1989), pp. 97-112.

[50] H. Yan y G. Thompson, *Finding postal carrier walk paths in mixed graphs*, Computational Optimization and Applications, 9 (1998), pp. 229-247.

[51] F. J. Zaragoza Martínez, *Complexity of the mixed postman problem with restrictions on the edges*, in Proceedings of the Seventh Mexican International Conference on Computer Science (ENC'06), IEEE, Washington, DC, 2006, pp. 3-10.

[52] F. J. Zaragoza Martínez, *Postman Problems on Mixed Graphs*, Ph.D. thesis, University of Waterloo, Waterloo, Ontario, Canada, 2003.

[53] F. J. Zaragoza, *Series-parallel graphs are windy postman perfect*, Discrete Mathematics, 308 (2008), pp. 1366-1374.