Universidad de Cantabria

Facultad de Ciencias

EL ALGORITMO DE BELLMAN-FORD COMO HERRAMIENTA PARA LA RESOLUCIÓN DE PROBLEMAS DE PROGRAMACIÓN LINEAL

(The Bellman-Ford Algorithm as a Tool for Solving Linear Programming Problems)

Trabajo Fin de Grado para acceder al

GRADO EN MATEMÁTICAS

Autora: Yolanda Díaz Asensio

Director: Rafael Duque Medina

Co-Director: Camilo Palazuelos Calderón

Junio - 2025

Agradecimientos

A mis tutores, Rafael y Camilo, por aceptar guiarme en este proyecto y escuchar mis dudas.

A mis profesores, por compartir sus conocimientos. Gracias Fernando, por guiarme como tutor a lo largo de la carrera.

A mis padres, por apoyarme siempre. Gracias por animarme a seguir cuando pensaba que no podía más y por escuchar cada preocupación que me pasaba por la cabeza. Por ser mucho más que mis padres: mi ejemplo a seguir, mis amigos y mi refugio en los momentos difíciles. Por no dudar en convertiros en expertos en cualquier campo cada vez que necesitaba vuestra ayuda.

A mi familia, por confiar siempre en mí. En especial a la yaya, por recordarme cada vez que podía que cómo no iba a ser capaz de hacerlo, si cuando estaba en su casa de pequeña siempre hacía los deberes de matemáticas en un periquete.

A mi segunda familia, mis amigos de la carrera. Gracias por hacer que Santander se convirtiese durante estos cuatro años en nuestro nuevo hogar. Por las horas en el aula búho deseando que llegase la hora del café. Por hacer las horas de estudio mucho más amenas y las de ocio mucho más divertidas. Sin vosotros esto hubiese sido mucho más difícil. A Nerea, por contagiarme diariamente su ilusión y por convertir cada momento que pasamos juntas en una aventura.

A mis amigos de siempre, a los de Logroño, porque hace cuatro años soñaba con irme y ahora sólo pienso en volver. Gracias Alejandro, por entenderme y apoyarme incondicionalmente. Por buscar siempre la manera de sacarme una sonrisa y por transmitirme paz en los días más complicados.

Y a todas las personas que han hecho de estos años de carrera una experiencia inolvidable, gracias.

Resumen

El problema de los caminos más cortos de fuente única es fundamental en la teoría de grafos y en la optimización combinatoria. Este tipo de problema consiste en encontrar la ruta más corta desde un vértice de origen hacia todos los demás vértices de un grafo, y tiene una amplia variedad de aplicaciones en áreas como redes de transporte, logística y planificación de recursos.

Este trabajo propone un enfoque progresivo: se parte de la formulación del problema en grafos y se demuestran una serie de propiedades de los caminos más cortos. Después, se analiza el algoritmo de Bellman-Ford, junto con la mejora de Yen. Finalmente, se estudia un caso especial de programación lineal que se puede reducir a la resolución de un problema de caminos más cortos con el algoritmo de Bellman-Ford, mostrando así la conexión entre la optimización en grafos y la programación lineal.

Palabras clave: problema de los caminos más cortos de fuente única, grafo, relajación de aristas, algoritmo de Bellman-Ford, programación lineal

Abstract

The single-source shortest paths problem is essential in graph theory and combinatorial optimization. This type of problem involves finding the shortest path from a source vertex to every other vertex in a graph and has a wide variety of applications in areas such as transportation networks, logistics, and resource planning.

This dissertation proposes a progressive approach: it begins with the formulation of the problem in graphs, and proves some important properties of shortest paths. Then, it analyzes the Bellman-Ford algorithm and Yen's improvement. Finally, it studies a special case of linear programming that can be reduced to solving a shortest path problem using the Bellman-Ford algorithm, thereby showing the connection between graph optimization and linear programming.

Key words: single-source shortest paths problem, graph, edge relaxation, Bellman-Ford algorithm, linear programming

Índice general

Introducción

Los algoritmos forman parte del día a día de todas las personas. Son el motor detrás de las plataformas digitales, determinando el contenido que aparece en redes sociales y optimizando las búsquedas en internet para ofrecer resultados más atractivos y acordes a los gustos de cada uno. Sin embargo, su influencia va mucho más allá del ámbito informático; los algoritmos están presentes en una amplia variedad de disciplinas, desempeñando un papel fundamental en la toma de decisiones.

A lo largo de la historia, los algoritmos han coexistido con la humanidad de manera similar a otras ramas fundamentales de la Matemática. Si bien existen evidencias de métodos algorítmicos tempranos, el término *algoritmo* no apareció hasta el siglo IX, cuando el matemático persa Muhammad Ibn Musa al-Juarismi lo introdujo en sus tratados sobre aritmética y cálculo. Por otro lado, la relación entre los algoritmos y la informática no se estableció hasta el siglo XIX, cuando la matemática británica Ada Lovelace escribió lo que hoy se considera como el primer algoritmo informático de la historia. Esta mujer anticipó el potencial de los algoritmos para ir más allá de simples cálculos numéricos, sentando las bases de la programación moderna.

Este trabajo aborda, en concreto, la aplicación e importancia que tienen los algoritmos a la hora de recorrer grafos. Buscar en un grafo significa recorrer sus vértices y aristas de manera sistemática para encontrar información específica, como la existencia de un camino entre dos vértices. Cientos de problemas computacionales interesantes están formulados en términos de grafos. Uno de estos problemas, y el objeto de este estudio, es el llamado *problema de los caminos más cortos de fuente única*, que trata de encontrar el camino más corto desde un determinado vértice de un grafo hasta cada vértice restante de dicho grafo.

La importancia de este problema radica en su aplicabilidad a múltiples disciplinas, sirviendo como base para modelar y resolver problemas de optimización. Muchos de estos problemas pueden transformarse en grafos, facilitando su análisis y resolución mediante algoritmos.

En la actualidad, la resolución de este problema ha impulsado el desarrollo de tecnologías clave como el GPS o la planificación optimizada de rutas de autobuses, trenes y aviones, reduciendo tiempos de espera y maximizando el uso eficiente de recursos.

Contenido

A continuación, se describe brevemente la estructura de la memoria.

- **Capítulo 1:** En primer lugar, se introducen los conceptos fundamentales de la teoría de grafos necesarios para una compresión clara del trabajo, con un enfoque en el pro-

blema de los caminos más cortos de fuente única. Se comienza con un repaso de nociones básicas y se introducen términos clave como el peso de un camino, la estimación del peso de un camino más corto y el subgrafo de predecesores. Después, se enuncian las variantes del problema de los caminos más cortos y se exploran propiedades fundamentales sobre la estructura de estos caminos. Por último, se presenta la técnica de relajación de aristas junto con el procedimiento que se va a utilizar para inicializar los valores de las estimaciones de los pesos de los caminos más cortos y los predecesores.

- **Capítulo 2:** En este capítulo, se desarrollan y demuestran una serie de lemas sobre los caminos más cortos. Primero, se introduce la desigualdad triangular para los pesos de los caminos más cortos. Después, se exploran propiedades relacionadas con el impacto de la relajación de aristas sobre las estimaciones del peso de los caminos más cortos, como la cota superior de estas estimaciones, su convergencia y la relajación de aristas a lo largo de un camino. Finalmente, se analiza el subgrafo de predecesores, demostrando que, bajo ciertas condiciones, forma un árbol de caminos más cortos enraizado en un determinado vértice del grafo. Se incluyen ejemplos y figuras elaborados por la autora para una mejor comprensión de estas propiedades.
- **Capítulo 3:** Una vez expuestas las características del problema de los caminos más cortos de fuente única, este capítulo presenta un método para resolverlo: el algoritmo de Bellman-Ford. Se describe detalladamente el funcionamiento de este algoritmo, ilustrado con un ejemplo de ejecución sobre un grafo específico, y se demuestra la corrección del algoritmo utilizando propiedades del capítulo anterior. También se expone la mejora de Yen para este algoritmo, que reduce el número de iteraciones necesarias para que el algoritmo de Bellman-Ford encuentre los caminos más cortos a cada vértice.
- **Capítulo 4:** Este capítulo explora la relación entre la búsqueda de caminos en grafos y la programación lineal, centrándose en un caso especial de programación lineal que se reduce a encontrar caminos más cortos de fuente única. Inicialmente, se introducen los problemas de programación lineal con un determinado tipo de restricciones de desigualdad. Después, se explica cómo estas restricciones se pueden modelar mediante un grafo de restricciones que transforma el problema de resolver un sistema de restricciones de desigualdad en uno de búsqueda de pesos de caminos más cortos. Finalmente, se justifica que, en efecto, el algoritmo de Bellman-Ford resuelve estos sistemas y puede ser usado como herramienta para la resolución de algunos problemas de programación lineal.

El Capítulo 22 del libro *Introduction to Algorithms* [6] ha servido como esqueleto para elaborar este trabajo. Todos los ejemplos y figuras, excepto la Figura 2.5, que aparecen a lo largo de esta memoria son contribuciones directas de la autora. La autora también ha probado por cuenta propia algunos lemas y observaciones. Al comienzo de cada capítulo se indican cuáles son.

Capítulo 1

Preliminares

El diseño de rutas en el transporte interurbano responde a criterios de eficiencia operativa que pueden no coincidir con la optimización del tiempo de viaje para cada pasajero. En trayectos como el de Santander a Logroño, los autobuses siguen itinerarios predefinidos que incluyen paradas intermedias para maximizar la ocupación y optimizar el consumo de combustible, lo que prolonga la duración del viaje en comparación con un desplazamiento en vehículo privado.

Por otro lado, los conductores particulares pueden seleccionar rutas más directas, apoyándose en aplicaciones de navegación que calculan el trayecto óptimo según múltiples variables, como los límites de velocidad, la tipología de la vía y las condiciones del tráfico en tiempo real. En la Figura 1.1 se muestra el resultado que brinda el navegador *Google Maps* cuando se pide una ruta para hacer un desplazamiento en coche desde Santander hasta Logroño. Esto plantea una cuestión fundamental: ¿cómo determinan estos sistemas la mejor ruta?

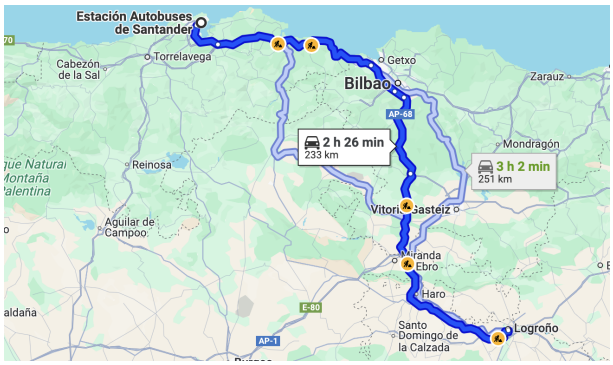


Figura 1.1: Ruta más rápida entre Santander y Logroño obtenida con Google Maps [8]

Estas aplicaciones siguen un algoritmo. En el caso de querer encontrar la ruta más corta entre dos puntos de un mapa, el algoritmo toma como entrada el mapa, el punto de origen y el de destino, junto con las múltiples variables que pueden afectar al tiempo de recorrido de las carreteras; y devuelve un itinerario con las carreteras a seguir, así como la distancia y el tiempo requerido para completarlo. Durante este procedimiento resulta lógico que el algoritmo descarte alternativas en las que el vehículo pase más de una vez por el mismo punto o se aleje significativamente del destino. No obstante, todavía recorrería una gran cantidad de rutas poco relevantes. Es por ello que se diseñan algoritmos para optimizar esta búsqueda, sobre los que se habla en el Capítulo 3 de este trabajo.

En términos matemáticos este tipo de problema se puede formular dentro de la teoría de grafos, de modo que los vértices del grafo representan las intersecciones de carreteras en el mapa, las aristas representan los segmentos de carretera entre estas intersecciones y el peso de las aristas representa, en el ejemplo tratado, la distancia o tiempo que se tarda en recorrer las carreteras. En particular, encontrar la ruta más corta entre dos ciudades, o dos puntos, es una variante del problema más general de los caminos más cortos de fuente única, estudiado desde mediados del siglo XX [12].

En este primer capítulo se presenta el problema de los caminos más cortos de fuente única, sus variantes y algunas cuestiones fundamentales sobre estos caminos. También se plantean algunos procedimientos que tienen en común varios de los algoritmos que resuelven estos problemas. Para ello, se ha utilizado como referencia principal el Capítulo 22 y el Apéndice B del libro *Introduction to Algorithms* [6], junto con las notas del curso *CS3221.01 Algorithm Analysis* de la Universidad de Plymouth [14]. Cabe destacar que la demostración de la Observación 1.17 ha sido realizada por la autora de este trabajo.

1.1. Problema de los caminos más cortos de fuente única

Para definir formalmente el problema de los caminos más cortos de fuente única es necesario concretar qué se entiende por camino más corto. Con este propósito se exponen, a modo de recordatorio, algunas nociones básicas de teoría de grafos y se fija la notación que se va a seguir a lo largo del trabajo. Para definir algunas de estas nociones, se han consultado los apuntes de la asignatura Matemática Discreta, del Grado en Matemáticas de la Universidad de Cantabria [7], y el Capítulo 9 del libro *Matemáticas discretas* [9].

Definición 1.1. *Un **grafo dirigido** es un par ordenado $G = (V, A)$ donde V es un conjunto cuyos elementos se denominan **vértices** o **nodos** y $A \subseteq V \times V$ es un conjunto de pares ordenados de vértices que se denominan **aristas** o **arcos dirigidos**.*

Observación 1.2. *En este trabajo, todos los grafos considerados son grafos finitos.*

Cuando se quiera hacer referencia al conjunto de vértices o al conjunto de aristas de un grafo G dado, se escribirá $V(G)$ o $A(G)$, respectivamente. Por otro lado, al considerar una arista $a_i = (v_{i-1}, v_i) \in A$ en un grafo dirigido G, se usará la siguiente terminología:

1. La arista es incidente desde el vértice v_{i-1}, es decir, comienza en v_{i-1}, y es incidente al vértice v_i, lo que significa que termina en v_i. Así, v_{i-1} es el **origen** o vértice inicial de la arista, y v_i es el **destino** o vértice terminal. En algunas ocasiones, al vértice origen se le llamará **fuente** y se denotará s.

2. v_i es un **sucesor** de v_{i-1} mientras que v_{i-1} es un **predecesor** de v_i.

Definición 1.3. *Sea $G = (V, A)$ un grafo dirigido y $k \in \mathbb{N}$. Un **camino** en G es una secuencia $p = \langle v_0 a_1 v_1 a_2 v_2 \dots v_{k-1} a_k v_k \rangle$, donde, para cada $i \in \{1, \dots, k\}$, $v_i \in V$, $v_0 \in V$ y $a_i \in A$ es una arista dirigida que empieza en v_{i-1} y acaba en v_i, en la que todas las aristas son distintas. Si no hay ambigüedad, un camino se denota por su secuencia de vértices $p = \langle v_0, v_1, v_2, \dots, v_{k-1}, v_k \rangle$. Se dice que un **camino es simple** si todos los vértices del camino son distintos. Por otro lado, si $v_0 = v_k$ y p no contiene ningún otro vértice, es decir $p = \langle v_0, v_k \rangle$, al camino se le llama **lazo**. Si $v_0 = v_k$ y p contiene al menos un vértice adicional, se le llama **ciclo**. Asimismo, si todos los vértices del ciclo son distintos, pasará*

a ser un ***ciclo simple****. Por contra, si un camino o un grafo no tiene ciclos se llamará* ***camino acíclico*** *y* ***grafo acíclico****, respectivamente.*

Existen grafos en los que algunos de sus vértices no se pueden unir mediante un camino. Es por ello que cuando exista un camino cuyo origen es el vértice v y su destino es el vértice v', se dirá que v' **es alcanzable desde** v **a través de** p y se denotará como $v \rightsquigarrow^p v'$. Adicionalmente, si se necesita indicar que el camino pasa por un vértice intermedio u se escribirá $s \rightsquigarrow u \rightsquigarrow v$ y en el caso de que (u, v) sea una arista del grafo, se escribirá $s \rightsquigarrow u \to v$.

En muchos problemas relacionados con grafos, no solo es relevante la existencia de un camino entre dos vértices, sino también ciertas características asociadas a las aristas que lo conforman, como su longitud, coste o capacidad. Para modelar estas situaciones, es necesario introducir la noción de grafo dirigido ponderado, en el que a cada arista se le asigna un valor numérico que representa una magnitud de interés.

Definición 1.4. *Un* ***grafo dirigido ponderado*** *es un grafo dirigido $G = (V, A)$ junto con una aplicación $w : A \to \mathbb{R}$. w es la* ***aplicación peso*** *que asigna a cada arista un peso, longitud, tamaño...*

En la Figura 1.2 se muestra un ejemplo de un grafo dirigido ponderado acíclico, en el que s es el vértice fuente, que cuenta con seis vértices unidos por varias aristas dirigidas, cada una de ellas con un peso determinado.

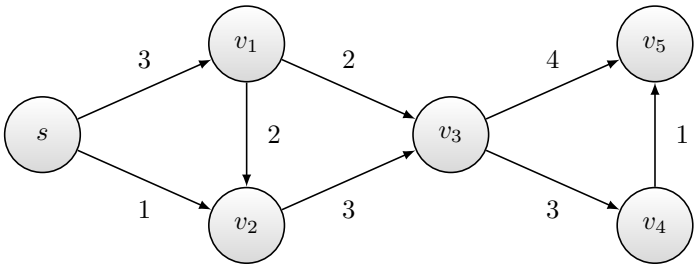


Figura 1.2: Grafo dirigido ponderado acíclico

A la hora de recorrer estos grafos con algoritmos resultará de gran utilidad saber si cuentan con alguna arista con peso negativo o no. Se llama **grafo dirigido ponderado con pesos positivos** a aquel en el que todas las aristas tienen peso positivo, o nulo, y un **grafo dirigido ponderado con pesos negativos** es aquel en el que al menos una de las aristas tiene peso negativo.

Notación. En todo lo que sigue, para simplificar la notación, cuando únicamente se escribe grafo, la autora se refiere a un grafo dirigido ponderado.

Definición 1.5. *Sea $k \in \mathbb{N}$. El* ***peso, $w(p)$, de un camino*** *$p = \langle v_0, v_1, \ldots, v_k \rangle$ se define como la suma de los pesos de las aristas que lo componen, es decir,*

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

En función de si el resultado de la suma es positivo o negativo se dice que el camino es de peso positivo o de peso negativo, respectivamente.

Definición 1.6. *Sea $\langle v_0, v_1, \ldots, v_k\rangle$ un camino más corto, con $k \in \mathbb{N}$, se llama* ***peso de un camino más corto de*** v_0 ***a*** v_k *al peso mínimo entre todos los caminos de v_0 a v_k, o infinito si no existe tal camino, y se denota:*

$$\delta(v_0, v_k) = \begin{cases} \text{mín}\{w(p) : v_0 \rightsquigarrow^p v_k\} & \text{si existe, al menos, un camino de } v_0 \text{ a } v_k, \\ \infty & \text{en otro caso.} \end{cases}$$

Cabe destacar que el peso del camino más corto de un vértice a sí mismo es 0, es decir, dado un grafo G se define $\delta(v, v) = 0$ para todo $v \in V(G)$.

Los algoritmos que aparecen en esta memoria asignan a cada vértice $v \in V$ de un grafo un atributo $v.d$ denominado **estimación del peso de un camino más corto**. Este atributo representa una cota superior, como se prueba detalladamente en el Lema 2.2, para el peso de un camino más corto con origen la fuente s y destino el vértice v. Su valor se puede ir actualizando conforme el algoritmo avanza en sus iteraciones y cuando se iguale, si lo hace, al valor del peso del camino más corto, se dirá que la **estimación es óptima.**

Definición 1.7. *Conociendo todas las anteriores definiciones, se llama* ***camino más corto del vértice*** v_0 ***al vértice*** v_k *a cualquier camino p que comience en el vértice v_0 y termine en v_k, cuyo peso cumpla que $w(p) = \delta(v_0, v_k)$.*

En el grafo de la Figura 1.2, se observa que un posible camino desde s hasta el vértice v_5 es $p_1 = \langle s, v_1, v_3, v_5\rangle$, con peso $w(p_1) = 9$. Sin embargo, existen caminos alternativos con menor peso. Tanto $p_2 = \langle s, v_2, v_3, v_5\rangle$ como $p_3 = \langle s, v_2, v_3, v_4, v_5\rangle$ tienen un peso cuyo valor es 8, que es el mínimo posible para alcanzar el vértice v_5 en el grafo dado si se toma como fuente el vértice s. En este ejemplo se puede ver que el camino más corto entre dos vértices de un grafo no es necesariamente único.

Notación. La posibilidad de que haya varios caminos más cortos hace que en muchas ocasiones, cuando en este trabajo se haga referencia al peso del camino más corto desde el vértice fuente hasta otro vértice, se dé por hecho que es el peso de todos los posibles caminos más cortos desde el vértice fuente hasta dicho vértice.

Problema 1.8 (**Problema de los caminos más cortos de fuente única**)**.** *Dado un grafo $G = (V, A)$, encontrar un camino más corto desde una fuente única dada, $s \in V$, hasta cada vértice restante del grafo, $v \in V \setminus \{s\}$.*

Para resolver este problema, va a resultar útil que los algoritmos lleven un registro de qué vértice precede a cada uno en el camino más corto. Por ello, cada vértice $v \in V$ cuenta con un vértice **predecesor** $v.\pi$, que será NULL si v es el primer vértice del camino. Al igual que la estimación del peso del camino más corto, este valor se puede actualizar conforme el algoritmo avanza en sus iteraciones.

El grafo que se define a continuación, cuyas aristas conectan cada vértice v con su predecesor $v.\pi$, es una forma de representar cómo los algoritmos construyen los caminos más cortos a medida que avanzan en sus iteraciones.

Definición 1.9. *Sea el grafo $G = (V, A)$. El* ***subgrafo de predecesores*** $G_\pi = (V_\pi, A_\pi)$ *inducido por los predecesores $v.\pi$ donde $v \in V(G)$, tomando un vértice $s \in V(G)$ como fuente, está compuesto por los siguientes subconjuntos de V y A, respectivamente:*

$$V_\pi = \{v \in V : v.\pi \neq \text{NULL}\} \cup \{s\},$$
$$A_\pi = \{(v.\pi, v) \in A : v \in V_\pi \setminus \{s\}\}.$$

Definición 1.10. *Sea $G = (V, A)$ un grafo no dirigido. Se dice que G es un **árbol** si es un grafo no dirigido, conexo y acíclico. Si alguno de sus vértices se distingue de los demás pasa a llamarse **árbol enraizado**, de modo que el vértice distinguido se convierte en la **raíz** del árbol.*

Definición 1.11. *Sea $G = (V, A)$ un grafo con aplicación peso $w : A \to \mathbb{R}$. Se asume que G no contiene ciclos de peso negativo alcanzables desde un vértice fuente $s \in V$, para que los caminos más cortos estén bien definidos. Un **árbol de caminos más cortos enraizado en** s es un subgrafo dirigido $G' = (V', A')$ donde $V' \subseteq V$ y $A' \subseteq A$ de modo que:*

1. *V' es el conjunto de vértices en G alcanzables desde s,*
2. *G' forma un árbol enraizado cuya raíz es s, y*
3. *para cada $v \in V'$, el único camino simple con origen s y destino v en G' es un camino más corto con origen s y destino v en G.*

Cabe destacar que el subgrafo de predecesores G_π no es necesariamente un árbol de caminos más cortos en todo momento ya que, como se ha mencionado anteriormente, el predecesor de un vértice cualquiera, $v.\pi$, puede cambiar a lo largo de la ejecución de un algoritmo. Sin embargo, al finalizar la ejecución de una serie de pasos del algoritmo, como se demuestra en la Sección 2.3, G_π siempre va a ser un árbol de caminos más cortos enraizado en s.

1.2. Variantes del problema

La resolución del problema de los caminos más cortos de fuente única, en concreto, es muy interesante ya que pequeñas modificaciones en sus algoritmos de búsqueda permiten resolver otros problemas similares, como ocurre con los siguientes:

Problema 1.12 (**Problema de los caminos más cortos de destino único**)**.** *Dado un grafo $G = (V, A)$, encontrar un camino más corto a un vértice destino dado $t \in V$ desde cada vértice $v \in V \setminus \{t\}$.*

Solución: Basta con invertir la dirección de cada arista en el grafo y resolver el problema como uno de fuente única. Finalmente, habrá que invertir el camino más corto desde la fuente $t \in V$ resultante para obtener la solución del problema original.

Problema 1.13 (**Problema de los caminos más cortos entre un par de vértices dados**)**.** *Dado un grafo $G = (V, A)$, encontrar un camino más corto que sale de u y entra en v para unos vértices $u, v \in V$ determinados.*

Esta es la formulación concreta del ejemplo de encontrar la ruta de menor tiempo entre Santander y Logroño que se ha presentado al principio del capítulo.

Solución: Es un problema de fuente única, cuya fuente es el vértice $u \in V$, en el que únicamente es necesario fijarse en el camino más corto con destino el vértice $v \in V$.

Problema 1.14 (**Problema de los caminos más cortos para todos los pares de vértices**). *Dado un grafo $G = (V, A)$, encontrar un camino más corto que sale de u y entra en v para cada par de vértices $u, v \in V$.*

Solución: Este problema se puede resolver ejecutando un algoritmo de fuente única para cada vértice. Sin embargo esta resolución no resulta muy práctica ya que, atendiendo a su estructura, existen otros algoritmos mucho más rápidos. Estos algoritmos se alejan de los objetivos de este trabajo, no obstante su estudio está recogido en el Capítulo 23 del libro *Introduction to Algorithms* [6].

1.3. Subestructura óptima de los caminos más cortos

Los problemas de caminos más cortos suelen basarse en el Lema 1.16, el cual se enuncia y demuestra a continuación. Este resultado cobra gran importancia a la hora de diseñar los algoritmos de búsqueda, ya que señala que los problemas de caminos más cortos presentan subestructura óptima, es decir, se puede construir la solución óptima de estos problemas a partir de soluciones óptimas de subproblemas más pequeños.

Definición 1.15. *Sea $G = (V, A)$ un grafo, $k \in \mathbb{N}$ y $p = \langle v_0, v_1, v_2, \ldots, v_{k-1}, v_k \rangle$ un camino en G. Un **subcamino del camino** p es una subsecuencia contigua de sus vértices, manteniendo el orden original y respetando las aristas del camino original. Es decir, para cualesquiera $i, j \in \mathbb{N}$ tales que $0 \leq i \leq j \leq k$, la subsecuencia de vértices $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ es un subcamino de p.*

Lema 1.16 (**Los subcaminos de caminos más cortos son caminos más cortos**). *Sea $G = (V, A)$ un grafo cuya aplicación peso es $w : A \to \mathbb{R}$ y $p = \langle v_0, v_1, \ldots, v_k \rangle$ un camino más corto que comienza en el vértice v_0 y termina en el vértice v_k. Sea, para cualesquiera $i, j \in \mathbb{N}$ tales que $0 \leq i \leq j \leq k$, $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ el subcamino de p cuyo origen es v_i y destino es v_j. Entonces, se puede afirmar que p_{ij} es un camino más corto que comienza en v_i y termina en v_j.*

Demostración. Se descompone el camino más corto p, que comienza en v_0 y termina en v_k, en tres subcaminos denotados p_{0i}, p_{ij}, p_{jk} tales que $v_0 \rightsquigarrow^{p_{0i}} v_i \rightsquigarrow^{p_{ij}} v_j \rightsquigarrow^{p_{jk}} v_k$. Por cómo se ha definido el peso de un camino, se cumple la igualdad $w(p) = w(p_{0i}) + w(p_{ij}) + w(p_{jk})$.

Por reducción al absurdo, se supone que existe un camino p'_{ij} que comienza en v_i y termina en v_j cuyo peso cumple que $w(p'_{ij}) < w(p_{ij})$. En este caso, se tendría que $v_0 \rightsquigarrow^{p_{0i}} v_i \rightsquigarrow^{p'_{ij}} v_j \rightsquigarrow^{p_{jk}} v_k$ es un camino con origen v_0 y destino v_k cuyo peso cumple que $w(p_{0i}) + w(p'_{ij}) + w(p_{jk}) < w(p)$, contradiciendo así la hipótesis de que p es un subcamino más corto cuyo origen es v_0 y destino es v_k. □

1.4. Ausencia de ciclos en los caminos más cortos

Al inicio del capítulo se ha destacado cómo los conductores de vehículos particulares pueden seleccionar rutas óptimas orientadas a minimizar su tiempo de desplazamiento. En este contexto, no tiene sentido permitir que el peso de las aristas, es decir, el tiempo de desplazamiento, sea negativo. Esto sucede en muchos otros problemas, lo que lleva a que algunos algoritmos de búsqueda estén diseñados bajo el supuesto de que todos los pesos de las aristas del grafo de entrada sean no negativos.

Por otro lado, también se ha señalado que los conductores de vehículos de transporte interurbano siguen rutas que buscan optimizar otras variables, como la ocupación y el consumo de combustible. En este contexto, se puede considerar un grafo $G = (V, A)$ donde $V(G)$ representa el conjunto de ciudades en las que el vehículo puede parar, y cada arista $(u, v) \in A(G)$ tiene un peso $w(u, v)$ que indica las ganancias netas obtenidas al viajar de la ciudad u a la ciudad v. En ciertas situaciones, como cuando el vehículo se desplaza entre dos ciudades sin pasajeros o con baja ocupación, los costes asociados al viaje (consumo de combustible, gastos operativos, etc.) pueden superar a los ingresos, resultando en un valor negativo para $w(u, v)$. Este tipo de modelo justifica la necesidad de algoritmos que pueden recibir como entrada grafos con aristas de pesos negativos y que sean capaces de devolver caminos más cortos que las puedan llegar a contener.

También resulta importante destacar que, en el contexto del problema de los caminos más cortos de fuente única, los grafos estudiados sí pueden contener ciclos. En el caso en el que el grafo contenga un ciclo de peso negativo, este no podrá ser alcanzable desde el vértice fuente. De este modo, como el ciclo no es alcanzable desde la fuente, el camino más corto desde ese vértice hasta cualquier otro del grafo nunca incluirá al ciclo. Por este motivo, para ver si en un grafo existe un camino más corto desde la fuente hasta cualquier otro vértice del grafo, resulta muy útil contar con algoritmos capaces de determinar si los ciclos de peso negativo del grafo son alcanzables o no desde la fuente. Por ejemplo, el algoritmo de Bellman-Ford, desarrollado en detalle en el Capítulo 3, está diseñado para detectar la existencia de ciclos de peso negativo.

Observación 1.17. *Si el grafo $G = (V, A)$ del problema contiene un ciclo de peso negativo alcanzable desde un vértice fuente $s \in V$, entonces el peso de los caminos más cortos no está bien definido.*

Demostración. Sea $w : A \to \mathbb{R}$ la aplicación peso y $k \in \mathbb{N}$. Por reducción al absurdo se supone que $p_1 = \langle v_0, \ldots, v_k \rangle$ es un camino más corto con origen v_0 y destino v_k que contiene un ciclo $c_1 = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ de peso negativo, es decir $w(c_1) < 0$ y $v_i = v_j$, para cualesquiera $i, j \in \mathbb{N}$ tales que $0 \leq i \leq j \leq k$.

Aplicando inducción, se ve que al recorrer el ciclo un número arbitrario de veces, el peso del camino p_1 va disminuyendo, obteniéndose cada vez un nuevo camino más corto (de menor peso).

Caso base: Si el ciclo se recorre dos veces, se obtiene el camino $p_2 = \langle v_0, \ldots, v_i, \ldots, v_j, v_{i+1}, \ldots, v_j, \ldots v_k \rangle$ que contiene al ciclo $c_2 = \langle v_i, v_{i+1}, \ldots, v_j, v_{i+1}, \ldots, v_j \rangle$ cuyo peso cumple que $w(c_2) = 2w(c_1) < 0$ y entonces $w(p_2) < w(p_1)$ de modo que p_1 ya no es un camino más corto, contradiciendo la hipótesis.

Paso inductivo: Se toma como hipótesis de inducción que p_n es un camino más corto que contiene al ciclo c_n cuyo peso cumple que $w(c_n) = nw(c_1) < 0$, es decir, el ciclo se ha recorrido n veces. A continuación se estudia qué ocurre si el ciclo se recorre $n + 1$ veces. En este caso, se tiene el camino p_{n+1} que contiene al ciclo c_{n+1}, el cual cumple que $w(c_{n+1}) = (n + 1)w(c_1) = nw(c_1) + w(c_1) < nw(c_1) \overset{\text{H.I.}}{=} w(c_n)$, ya que $w(c_1) < 0$. De este modo, p_n ya no es un camino más corto, contradiciendo la hipótesis.

Una vez probado que, bajo las condiciones de la observación, siempre se puede encontrar un camino de menor peso, se concluye que el peso del camino más corto p_1 no está bien definido. □

Notación. Cuando un grafo contiene un ciclo de peso negativo y dicho ciclo forma

parte de algún camino de s a v, se define $\delta(s, v) = -\infty$.

Observación 1.18. *Un camino más corto no puede contener un ciclo de peso positivo.*

Demostración. Sea G un grafo, $w : A \to \mathbb{R}$ la aplicación peso y $k \in \mathbb{N}$. Por reducción al absurdo se supone que $p = \langle v_0, \ldots, v_k \rangle$ es un camino más corto con origen v_0 y destino v_k que contiene un ciclo $c = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ de peso positivo, es decir $w(c) > 0$ y $v_i = v_j$, para cualesquiera $i, j \in \mathbb{N}$ tales que $0 \leq i \leq j \leq k$. Si se elimina este ciclo se obtiene un nuevo camino $p' = \langle v_0, v_1, \ldots, v_i, v_{j+1}, \ldots, v_k \rangle$ cuyo peso $w(p') = w(p) - w(c)$ es menor que $w(p)$, ya que $w(c) > 0$, contradiciendo así que p sea un camino más corto con origen v_0 y destino v_k. □

Observación 1.19. *Si existe un camino más corto que contenga un ciclo de peso nulo, basta con eliminar el ciclo de dicho camino para obtener otro camino más corto con el mismo peso y sin ciclos.*

De acuerdo con estas observaciones se asume que los caminos más cortos no tienen ciclos, luego todos los vértices del camino son distintos, es decir, son caminos simples. Además, como cualquier camino simple en un grafo $G = (V, A)$ contiene a lo sumo $|V|$ vértices, también contiene a lo sumo $|V| - 1$ aristas. Así, se concluye que cualquier camino más corto contiene a lo sumo $|V| - 1$ aristas. Esta propiedad va a resultar útil a la hora de ver cuántas veces ejecutar ciertas instrucciones en los algoritmos.

1.5. Técnica de relajación de aristas

Muchos de los algoritmos que resuelven el problema de los caminos más cortos comienzan dando un valor inicial a la estimación del peso del camino más corto y al predecesor de cada vértice del grafo. Para ello, llaman al procedimiento INITIALIZE-SINGLE-SOURCE(G,s), donde G es un grafo y $s \in V(G)$ el vértice que se toma como fuente, cuyas instrucciones son las siguientes:

Algoritmo INITIALIZE-SINGLE-SOURCE(G, s)
1: **Para** cada vértice $v \in V(G)$ **hacer**
2: $\quad v.d = \infty$
3: $\quad v.\pi = \text{NULL}$
4: **Fin para**
5: $s.d = 0$
6: **Fin algoritmo**

Una vez inicializados estos valores, los algoritmos ejecutan el proceso de relajación de aristas. Relajar una arista $(u, v) \in A(G)$ consiste en verificar si atravesar el vértice u permite obtener un camino más corto con destino v de menor peso en comparación con el camino más corto encontrado hasta el momento. En caso afirmativo, se decrece el valor de la estimación $v.d$ y se modifica el predecesor del vértice v, $v.\pi$. En el caso en que el peso sea el mismo, no se actualiza el predecesor.

Algoritmo RELAX(u, v, w)
1: **Si** $v.d > u.d + w(u, v)$ **entonces**
2: $\quad v.d = u.d + w(u, v)$
3: $\quad v.\pi = u$
4: **Fin algoritmo**

El número de veces que se relaja cada arista de un grafo y el orden en el que lo hacen varía según el algoritmo.

Ejemplo 1.20 (**Relajación de dos aristas de la Figura 1.2**). *Sean* (v_2, v_3) *y* (v_4, v_5) *dos aristas del grafo presentado anteriormente en la Figura 1.2, cuyos pesos son* $w(v_2, v_3) = 3$ *y* $w(v_4, v_5) = 1$. *Este ejemplo muestra qué ocurre cuando se les aplica el proceso de relajación.*

Antes de relajarse, la arista de la izquierda de la Figura 1.3 cumple que $v_3.d = 8 > 1 + 3 = v_2.d + w(v_2, v_3)$, *por lo que al aplicar el algoritmo el valor de* $v_3.d$ *decrece. Esto se observa en la variación del valor de la estimación del peso del camino más corto del vértice destino, que se encuentra indicada encima de cada nodo. Por contra, en la arista de la derecha se tiene que* $v_5.d = 8 \leq 7 + 1 = v_4.d + w(v_4, v_5)$ *y entonces la arista no se relaja. En este caso, la estimación no varía.*

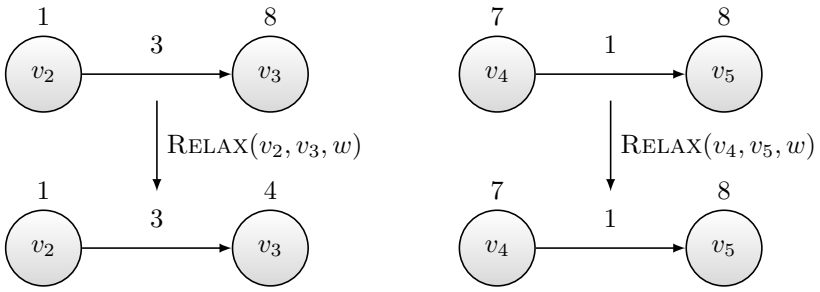


Figura 1.3: Ilustración del proceso de relajación de dos aristas de un grafo dirigido. A la izquierda, la relajación de (v_2, v_3) actualiza $v_3.d$, mientras que, a la derecha, la relajación de (v_4, v_5) no modifica $v_5.d$.

Capítulo 2

Propiedades de los caminos más cortos

En este capítulo se desarrollan y demuestran una serie de propiedades acerca de los caminos más cortos que resultarán de gran importancia a la hora de probar la corrección del algoritmo presentado en el Capítulo 3 dedicado a la búsqueda de caminos más cortos en grafos.

La principal referencia bibliográfica utilizada en este capítulo ha sido la Sección 22.5 del libro *Introduction to Algorithms* [6]. Para completar la demostración del Lema 2.11, la autora ha demostrado por cuenta propia los Lemas 2.8 y 2.9, para lo que ha consultado el Apéndice B del mismo libro. También es propia la demostración del caso 2 del Lema 2.1.

2.1. Desigualdad triangular

La desigualdad triangular, a veces denotada D.T., establece una cota superior para el peso de un camino más corto que comienza en el vértice fuente y termina en cualquier otro vértice del grafo, asegurando que este peso no supera el coste de atravesar cualquier vértice intermedio.

Lema 2.1 (**Desigualdad triangular**). *Sea $G = (V, A)$ un grafo con aplicación peso $w : A \to \mathbb{R}$ y sea $s \in V$ un vértice fuente. Entonces, para todas las aristas $(u, v) \in A$ se cumple que $\delta(s, v) \leq \delta(s, u) + w(u, v)$.*

Demostración. Se fijan dos vértices $u, v \in V$ y se considera la arista $(u, v) \in A$.

Caso 1: Se supone que existe un camino más corto $p = \langle s, v_1, \ldots, v_k = v \rangle$ desde la fuente s hasta el vértice v. Por definición de camino más corto, se tiene que $w(p) = \delta(s, v) \leq w(p')$ donde p' es cualquier otro camino en G con origen s y destino v. En particular, se considera el camino $\tilde{p} = \langle s, \ldots, u, v \rangle$, compuesto por el camino más corto con origen s y destino u seguido de la arista (u, v). Este camino alternativo tiene peso $\delta(s, u) + w(u, v)$ y entonces se puede afirmar que $w(p) = \delta(s, v) \leq \delta(s, u) + w(u, v) = w(\tilde{p})$.

Caso 2: Se supone que no existe un camino más corto desde la fuente s hasta el vértice v. Esto implica que v no es alcanzable desde s en el grafo G, ya que si lo fuese, el camino que uniese estos dos vértices pasaría a ser el camino más corto. De este modo se tiene que $\delta(s, v) = \infty$ y se dan dos subcasos:

- **Subcaso 2.1:** Si $\delta(s,u) = \infty$, es decir, si el vértice u tampoco es alcanzable desde s, la desigualdad $\infty \leq \infty + w(u,v)$ se cumple trivialmente para cualquier $w(u,v) \in \mathbb{R}$.
- **Subcaso 2.2:** Si $\delta(s,u) < \infty$, es decir, si el vértice u sí es alcanzable desde s, entonces existiría un camino más corto con origen s y destino u seguido de la arista $(u,v) \in A$, lo que implicaría que v es alcanzable desde s. Esto contradice la hipótesis del caso 2. Por lo tanto, este subcaso nunca se da, bajo las condiciones del grafo.

□

A continuación, se muestran dos grafos muy similares con cuatro vértices, pero con distintas ponderaciones en una de las aristas, en los que se señala un posible camino para ir desde el vértice fuente hasta otro vértice determinado:

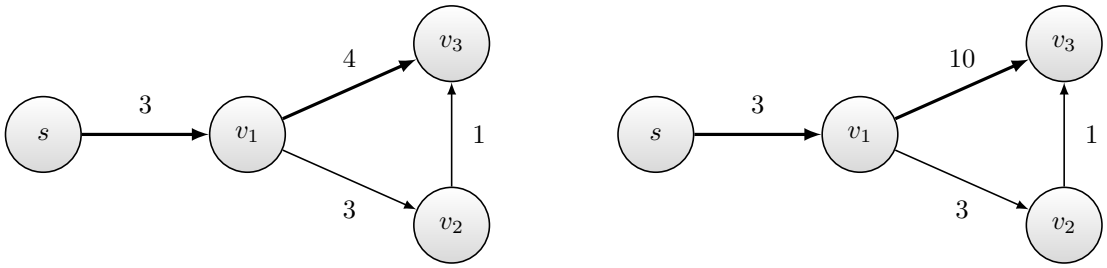


Figura 2.1: Ilustración de dos grafos en los que se señalan con mayor grosor las aristas que conforman el camino $p = \langle s, v_1, v_3 \rangle$.

En el grafo de la izquierda de la Figura 2.1, el camino $p = \langle s, v_1, v_3 \rangle$ es un camino más corto con origen el vértice fuente s y destino v_3, cumpliendo que $\delta(s,v_3) = 7 \leq 6 + 1 = \delta(s,v_2) + w(v_2,v_3)$. Por otro lado, p no es un camino más corto en el grafo de la derecha ya que, si lo fuese, se tendría que $\delta(s,v_3) = 13 > 6 + 1 = \delta(s,v_2) + w(v_2,v_3)$, es decir, al atravesar el vértice intermedio v_2 se encuentra un camino alternativo $\tilde{p} = \langle s, v_1, v_2, v_3 \rangle$ de menor peso.

2.2. Estimaciones del peso de caminos más cortos

En esta sección se presentan diversas propiedades que describen cómo las estimaciones de los caminos más cortos de un grafo, inicializado por INITIALIZE-SINGLE-SOURCE, se ven afectadas por la ejecución de una secuencia de pasos de relajación sobre las aristas de dicho grafo.

Lema 2.2 (**Propiedad de cota superior**). *Sea $G = (V,A)$ un grafo con aplicación peso $w : A \to \mathbb{R}$. Sea $s \in V$ un vértice fuente y sea el grafo inicializado por* INITIALIZE-SINGLE-SOURCE(G,s). *Entonces, $v.d \geq \delta(s,v)$ para todo $v \in V$, y esta desigualdad se mantiene invariante a lo largo de cualquier secuencia de pasos de relajación sobre las aristas de G. Es más, una vez que la estimación $v.d$ alcanza su cota inferior $\delta(s,v)$, su valor nunca cambia.*

Demostración. Primero, se demuestra que $v.d \geq \delta(s,v)$ para todo $v \in V$ aplicando inducción sobre el número de aristas relajadas.

Caso base: Tras la inicialización, cuando todavía no se ha relajado ninguna arista, se tiene que $v.d = \infty$ para todo $v \in V \setminus \{s\}$ y $s.d = 0$. Entonces, resulta trivial que para

todo $v \in V \setminus \{s\}$ se cumpla $\infty = v.d \geq \delta(s,v)$ y, que para el vértice fuente, se cumpla

$$0 = s.d \geq \delta(s,s) = \begin{cases} -\infty & \text{si } s \text{ forma parte de un ciclo de peso negativo} \\ 0 & \text{en otro caso.} \end{cases}$$

Paso inductivo: Se fijan dos vértices $u, v \in V$ y se considera la relajación de la arista $(u,v) \in A$. Para este paso se toma como hipótesis de inducción que $x.d \geq \delta(s,x)$ para todos los vértices $x \in V$ previos a la relajación de esta arista, incluyendo $u \in V$.

Al relajar la arista (u,v), la única estimación del peso del camino más corto que se puede ver afectada es la del vértice v, es decir $v.d$. Es por ello que sólo resulta necesario comprobar si para este vértice se sigue manteniendo la desigualdad:

- **Caso 1:** Si $v.d > u.d + w(u,v)$ antes de la relajación, entonces al relajar la arista varía la estimación y se tiene que

$$v.d \overset{\text{RELAX}(u,v,w)}{=} u.d + w(u,v) \overset{\text{H.I.}}{\geq} \delta(s,u) + w(u,v) \overset{\text{D.T.}}{\geq} \delta(s,v),$$

  manteniéndose así la desigualdad.

- **Caso 2:** Si $v.d < u.d + w(u,v)$ antes de la relajación, entonces $v.d$ no cambia y la desigualdad no se ve afectada.

Finalmente, se verifica, que una vez se alcanza la cota $v.d = \delta(s,v)$, el valor de $v.d$ no cambia. Esto se debe a que, por un lado, el valor de $v.d$ no puede aumentar ya que el proceso de relajación de una arista nunca aumenta la estimación y, por otro lado, el valor de $v.d$ no puede disminuir ya que se acaba de probar que $v.d \geq \delta(s,v)$. □

Una vez probado que la estimación del peso de un camino más corto actúa como cota superior para el peso de dicho camino más corto, se deriva un caso particular que aborda los vértices que no son alcanzables desde la fuente.

Corolario 2.3 (**Propiedad de inexistencia de camino**). *Sea $G = (V,A)$ un grafo con aplicación peso $w : A \to \mathbb{R}$ y sea $s \in V$ un vértice fuente. Si no existe un camino con origen s y destino un vértice $v \in V$, entonces, tras inicializar el grafo por el procedimiento* INITIALIZE-SINGLE-SOURCE(G,s)*, se cumple que $v.d = \delta(s,v) = \infty$, y esta igualdad se mantiene invariante a lo largo de cualquier secuencia de pasos de relajación sobre las aristas de G.*

Demostración. Se fija un vértice destino $v \in V$. Como, por hipótesis, no existe un camino con origen s y destino v, se tiene, por definición de peso de camino más corto, que $\delta(s,v) = \infty$. Entonces, por la propiedad de cota superior se debe cumplir que $\infty = \delta(s,v) \leq v.d$, luego, necesariamente, $v.d = \infty = \delta(s,v)$. □

El siguiente lema explora el efecto inmediato de la relajación de una arista sobre las estimaciones del peso de los caminos más cortos, el cual podría entenderse como una desigualdad triangular para esta estimación. Así, sienta las bases para entender cómo evoluciona el valor de dichas estimaciones a lo largo de la ejecución de algoritmos que relajan repetidamente las aristas de un grafo.

Lema 2.4. *Sea $G = (V,A)$ un grafo con aplicación peso $w : A \to \mathbb{R}$ y y sea $(u,v) \in A$. Entonces, inmediatamente después de que la arista (u,v) haya sido relajada mediante una llamada a* RELAX(u,v,w)*, se cumple que $v.d \leq u.d + w(u,v)$.*

Demostración. Esta demostración se divide en dos casos:

Caso 1: Si inmediatamente antes de la relajación de la arista (u, v) se tiene que $v.d > u.d + w(u, v)$, entonces al hacer la llamada a RELAX(u,v,w) se obtiene como resultado que $v.d = u.d + w(u, v)$.

Caso 2: Si, por contra, antes de esta relajación se tiene que $v.d < u.d + w(u, v)$, entonces la llamada al algoritmo no variará el valor de $u.d$ ni de $v.d$, manteniéndose así la desigualdad. □

A continuación, se presenta un lema que establece que, tras inicializar un grafo, la relajación de una arista en un camino más corto propaga una estimación óptima.

Lema 2.5 (**Propiedad de convergencia**). *Sea $G = (V, A)$ un grafo con aplicación peso $w : A \to \mathbb{R}$, sea $s \in V$ un vértice fuente y sea $s \rightsquigarrow u \to v$ un camino más corto en G para dos vértices $u, v \in V$ tales que la arista $(u, v) \in A$. Supóngase que G es inicializado por* INITIALIZE-SINGLE-SOURCE(G,s) *y que después se ejecuta una secuencia de pasos de relajación sobre sus aristas que incluye relajar la arista (u, v). Entonces, si $u.d = \delta(s, u)$ en cualquier momento antes de relajar esta arista, se tiene que $v.d = \delta(s, v)$ en todo momento después de esta relajación.*

Demostración. Se fijan $u, v \in V$ y se tiene, por hipótesis, que $s \rightsquigarrow u \to v$ es un camino más corto, luego, por el Lema 1.16, se tiene que el subcamino $s \rightsquigarrow u$ también es un camino más corto. Como además se conoce que $u.d = \delta(s, u)$ en cualquier momento antes de relajar la arista (u, v) entonces, por la propiedad de cota superior (Lema 2.2), se tiene que el valor de esta estimación nunca cambiará. En particular, una vez se haya relajado la arista (u, v) se tiene que

$$v.d \overset{\text{Lema 2.4}}{\leq} u.d + w(u, v) \overset{\text{hipótesis}}{=} \delta(s, u) + w(u, v) \overset{\text{Lema1,16}}{=} \delta(s, v).$$

Por otro lado, si se aplica la propiedad de cota superior al camino que comienza en s y termina en v, se tiene que $v.d \geq \delta(s, v)$ y esta desigualdad se mantiene invariante en todo momento después de la relajación.

Así se concluye por doble desigualdad que $v.d = \delta(s, v)$ en todo momento después de la relajación. □

Para finalizar esta sección, se extiende este resultado a caminos más cortos en los que se relajan todas sus aristas en un orden determinado, asegurando que esta secuencia ordenada de relajaciones hace que la estimación del peso de un camino más corto al vértice destino sea óptima, independientemente de lo que ocurra después.

Lema 2.6 (**Propiedad de relajación de caminos**). *Sea $G = (V, A)$ un grafo con aplicación peso $w : A \to \mathbb{R}$, sea $s \in V$ un vértice fuente y sea $k \in \mathbb{N}$. Se considera un camino más corto cualquiera $p = \langle v_0, v_1, ..., v_k \rangle$ que comienza en $s = v_0$ y termina en v_k. Si G es inicializado por* INITIALIZE-SINGLE-SOURCE(G,s) *y luego se ejecuta una secuencia de pasos de relajación sobre sus aristas que incluye relajar las aristas $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ en este orden, entonces $v_k.d = \delta(s, v_k)$ después de estas relajaciones y en todo momento a partir de entonces. Además, esta propiedad se mantiene independientemente de qué otras relajaciones de aristas ocurran, incluyendo aquellas que se intercalan con las relajaciones de las aristas de p.*

Demostración. Este lema se demuestra aplicando inducción sobre i, siendo i el número de aristas relajadas, en orden, del camino p.

Caso base: Se estudia el caso $i = 0$, es decir antes de que ninguna arista de p haya sido relajada. Como G ha sido inicializado por INITIALIZE-SINGLE-SOURCE(G,s), se tiene que $v_0.d = s.d = 0$, luego ya se da la igualdad buscada, esto es, $s.d = \delta(s, s) = 0$. Además, por la propiedad de cota superior (Lema 2.2) se tiene que el valor de $s.d$ nunca cambia tras la inicialización.

Paso inductivo: Se toma como hipótesis de inducción que la igualdad se cumple tras relajarse, en orden, las aristas $(v_0, v_1), (v_1, v_2), \ldots (v_{i-2}, v_{i-1})$ y se quiere ver qué ocurre tras relajarse adicionalmente la siguiente arista en el orden establecido, esta es (v_{i-1}, v_i). Como, por hipótesis de inducción, se tiene que $v_{i-1}.d = \delta(s, v_{i-1})$ antes de la llamada a RELAX(v_{i-1},v_i, w) entonces, aplicando la propiedad de convergencia (Lema 2.5) se verifica que $v_i.d = \delta(s, v_i)$ en todo momento después de esta relajación. □

Ejemplo 2.7. *Para entender mejor esta propiedad de relajación de caminos se considera el siguiente grafo (Figura 2.2) con vértice fuente s, en el cual se puede ver que el camino más corto con origen s y destino el vértice v_4 es $p = \langle s, v_1, v_3, v_4\rangle$, con peso $\delta(s, v_4) = 6$.*

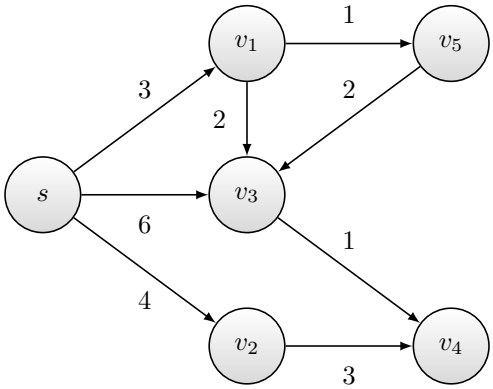


Figura 2.2: Grafo dirigido ponderado acíclico

Para poder aplicar la propiedad, se supone que el grafo ha sido inicalizado por INITIALIZE-SINGLE-SOURCE(G,s)*, de modo que los valores iniciales para las estimaciones del peso de los caminos más cortos con origen s y destino cada vértice del grafo son $s.d = 0$ y $v_i.d = \infty$ para todo $i \in \{1, 2, 3, 4, 5\}$. Se toma, de manera arbitraria, la siguiente secuencia de aristas del grafo sobre las que se aplica la técnica de relajación: $(s, v_2), (s, v_1), (v_1, v_3), (v_3, v_4)$, $(s, v_3), (v_1, v_5), (v_3, v_4)$. Como este proceso incluye relajar las aristas de p en orden, esto es, relajar $(s, v_1), (v_1, v_3), (v_3, v_4)$, por la propiedad de relajación de caminos (Lema 2.6) se sabe que tras relajarse estas tres aristas se tendrá que $v_4.d = \delta(s, v_4) = 6$. A continuación, se muestra lo que ocurre en cada paso de relajación de la secuencia:*

$$\begin{gathered}
\text{RELAX}(s,v_2,w) \rightarrow \textit{como } v_2.d = \infty > 0 + 4 = s.d + w(s, v_2) \rightarrow v_2.d = 4 \\
\text{RELAX}(s,v_1,w) \rightarrow \textit{como } v_1.d = \infty > 0 + 3 = s.d + w(s, v_1) \rightarrow v_1.d = 3 \\
\text{RELAX}(v_1,v_3,w) \rightarrow \textit{como } v_3.d = \infty > 3 + 2 = v_1.d + w(v_1, v_3) \rightarrow v_3.d = 5 \\
\text{RELAX}(v_3,v_4,w) \rightarrow \textit{como } v_4.d = \infty > 5 + 1 = v_3.d + w(v_3, v_4) \rightarrow v_4.d = 6 \\
\text{RELAX}(s,v_3,w) \rightarrow \textit{como } v_3.d = 5 < 0 + 6 = s.d + w(s, v_3) \rightarrow v_3.d = 5\textit{, no se actualiza} \\
\text{RELAX}(v_1,v_5,w) \rightarrow \textit{como } v_5.d = \infty > 3 + 1 = v_1.d + w(v_1, v_5) \rightarrow v_5.d = 4 \\
\text{RELAX}(v_3,v_4,w) \rightarrow \textit{como } v_4.d = 6 = 5 + 1 = v_3.d + w(v_3, v_4) \rightarrow v_4.d = 6\textit{, no se actualiza}
\end{gathered}$$

Como indica la propiedad, se obtiene que $v_4.d = 6 = \delta(s, v_4)$ y las relajaciones posteriores a la primera llamada a RELAX(v_3,v_4,w) *no afectan este resultado.*

2.3. Árboles de caminos más cortos

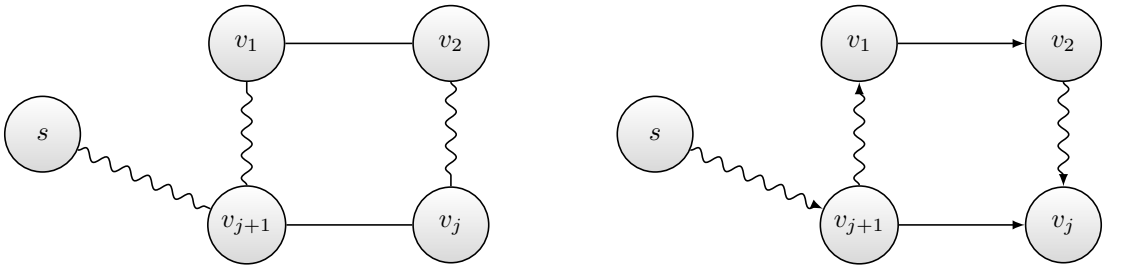
Una vez establecido que, tras una secuencia de pasos de relajación, la estimación del peso de un camino más corto converge al peso de ese camino más corto, en esta sección se prueba que, tras realizarse dichas relajaciones, el subgrafo de predecesores es un árbol de caminos más cortos. Para ello, se presentan previamente unos lemas que sustentan esta propiedad del subgrafo de predecesores.

Lema 2.8. *Sea $G = (V, A)$ un grafo dirigido acíclico en el que existe un vértice $s \in V$ tal que, para cada vértice $v \in V$, existe un único camino que empieza en el vértice s y termina en v. Entonces, la versión no dirigida de G es un árbol.*

Demostración. Sea $\tilde{G}$ la versión no dirigida del grafo G. $\tilde{G}$ será un árbol si es conexo y acíclico.

El grafo no dirigido $\tilde{G}$ será **conexo** si cada vértice $v \in V(\tilde{G})$ es alcanzable desde el resto de vértices de este grafo. Por como está definido, $\tilde{G}$ cumple que $V(\tilde{G}) = V(G)$ y sus aristas son las mismas que las de G pero pudiendo ser recorridas en ambas direcciones, es decir, si $(x, y) \in A(G)$ entonces se verifica que $(x, y), (y, x) \in A(\tilde{G})$. Como, por hipótesis del lema, se conoce que todos los vértices $v \in V$ son alcanzables desde s en el grafo G, dados dos vértices cualesquiera $u, u' \in V(\tilde{G})$ se tiene que existen los caminos $p_1 = \langle s, \ldots, u \rangle$ y $p_2 = \langle s, \ldots, u' \rangle$, que son únicos. Luego, recorriendo el camino p_1 a la inversa y recorriendo inmediatamente después el camino p_2, se construye un camino $p = \langle u, \ldots, s, \ldots, u' \rangle$ con origen u y destino u'. Así se concluye que cualquier vértice $u' \in V(\tilde{G})$ es alcanzable desde cualquier otro vértice $u \in V(\tilde{G})$.

Que $\tilde{G}$ sea **acíclico** se prueba por reducción al absurdo. Se supone que existe un ciclo en $\tilde{G}$, que se denota como $c = \langle v_1, \ldots, v_k \rangle$ donde $v_1, \ldots, v_k \in V(\tilde{G})$ y $v_1 = v_k$. Como se ha indicado para probar que $\tilde{G}$ es conexo, cada arista $(x, y) \in A(G)$ corresponde a dos aristas $(x, y), (y, x) \in A(\tilde{G})$ luego, cada arista (v_i, v_{i+1}) en c, con $i \in \{1, \ldots, k-1\}$, proviene o bien de una arista $(v_i, v_{i+1}) \in A(G)$ o bien de $(v_{i+1}, v_i) \in A(G)$. Si todas las aristas de c estuvieran orientadas en la misma dirección (por ejemplo, $(v_i, v_{i+1}) \in A(G)$ para todo $i \in \{1, \ldots, k-1\}$), c sería un ciclo dirigido en G, lo cual contradice la hipótesis de que el grafo dirigido G es acíclico. Por lo tanto, c debe incluir al menos una arista con distinta dirección, se supone que esta es, por ejemplo, $(v_j, v_{j+1}) \in A(\tilde{G})$, donde $(v_{j+1}, v_j) \in A(G)$.



(a) Grafo no dirigido en el que existe un ciclo $c = \langle v_1, v_2, \ldots, v_j, v_{j+1}, \ldots, v_k \rangle$ (donde $v_k = v_1$).

(b) Grafo dirigido acíclico en el que existen dos caminos distintos con origen s y destino v_j

Figura 2.3: Representación de un grafo no dirigido cíclico, grafo a), y su versión dirigida acíclica, grafo b)

Por hipótesis se conoce que en G existe un único camino desde s hasta cada vértice

v_i del ciclo, con $i \in \{1, \ldots, k-1\}$. Se fija un v_j de este ciclo y, en particular, se tiene que existe un único camino $p_j = \langle s, \ldots, v_1, v_2, \ldots v_j \rangle$ en G. Como se ha probado que $\tilde{G}$ es conexo, existe, por lo menos, un camino con origen s y destino v_{j+1} en $\tilde{G}$. En G, uno de estos caminos existe pero será único, por hipótesis, y se denota $p_{j+1} = \langle s, \ldots, v_{j+1} \rangle$, que también existe en $\tilde{G}$ ya que todas las aristas de G están en $\tilde{G}$. Desde v_{j+1}, el ciclo c en $\tilde{G}$ permite usar la arista (v_{j+1}, v_j), que por cómo se ha definido el ciclo está en G, para llegar a v_j. Esto crea un nuevo camino en G, $\langle s, \ldots, v_{j+1}, v_j \rangle$ con origen s y destino v_j, contradiciendo la unicidad de los caminos del grafo G que toma como hipótesis el lema. De este modo, c no puede existir y $\tilde{G}$ debe ser acíclico. □

La versión no dirigida del grafo dirigido ponderado acíclico de la Figura 2.2 no es un árbol ya que el grafo resultante cuenta con numerosos ciclos. Esto no sorprende ya que dicho grafo dirigido del que se parte cuenta con varios caminos distintos que comienzan en el vértice fuente y terminan en un mismo vértice, es decir no se cumple la condición del lema de que los caminos sean únincos. Esto es un ejemplo de que, a la hora de aplicar el Lema 2.8 (como se hace para finalizar la demostración del Lema 2.10), resulta crucial comprobar previamente que para cada vértice del grafo exista un camino con origen el vértice fuente y destino ese vértice, y que sea único. A continuación se muestra una modificación del grafo de la Figura 2.2 en la que se han eliminado algunas aristas para que se cumplan las condiciones del Lema 2.8 y cuya versión no dirigida sí es un árbol.

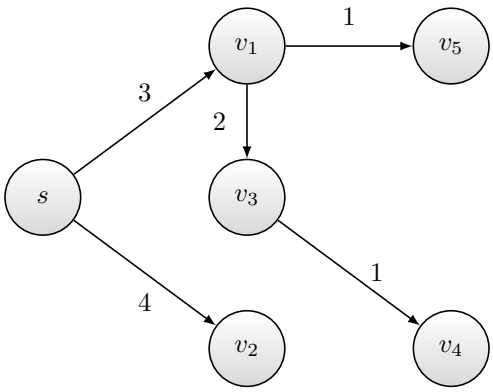


Figura 2.4: Grafo dirigido acíclico en el, que dado el vértice fuente s, existe un único camino con origen s y destino cada vértice restante v_i, con $i \in \{1, 2, 3, 4, 5\}$.

Lema 2.9. *Sea $G = (V, A)$ un grafo con aplicación peso $w : A \to \mathbb{R}$ que no contiene ciclos de peso negativo. Sea $s \in V$ el vértice fuente y G inicializado por* INITIALIZE-SINGLE-SOURCE(G,s). *Entonces, para cada vértice $v \in V_\pi$ existe, en G_π, un camino con origen s y destino v y esta propiedad se mantiene invariante a lo largo de cualquier secuencia de relajaciones.*

Demostración. Este lema se demuestra aplicando inducción sobre el número de pasos de relajación realizados sobre las aristas de G.

Caso base: Tras la inicialización, cuando todavía no se ha relajado ninguna arista, se tiene que $v.\pi =$ NULL para todo $v \in V$ y entonces, por definición, $V_\pi = \{s\}$. Trivialmente se tiene que existe un camino con origen el vértice fuente y destino sí mismo, un lazo, cumpliéndose así la propiedad.

Paso inductivo: Se toma como hipótesis de inducción que tras una secuencia cual-

quiera de n relajaciones, donde $(v_{n-1}, v_n) \in A(G)$ es la última arista que ha sido relajada en dicha secuencia, para cada vértice $v \in V_\pi$ existe en G_π un camino con origen s y destino v. Bajo esta hipótesis, se considera la relajación $(n+1)-$ésima, que consiste en una llamada a RELAX(v_n,v_{n+1},w) para una arista $(v_n, v_{n+1}) \in A(G)$. Para concluir que la propiedad se sigue cumpliendo para todo vértice del conjunto V_π actualizado, se analizan todos los posibles resultados de esta relajación:

- **Caso 1:** Si $v_{n+1}.d \leq v_n + w(v_n, v_{n+1})$, entonces la relajación no modifica $v.d$ ni $v.\pi$. En consecuencia, al no añadirse nuevos vértice a V_π, V_π permanece sin cambios y, por la hipótesis de inducción, se tiene que para todo $v \in V_\pi$ existe un camino con origen s y destino v en G_π.
- **Caso 2:** Si $v_{n+1}.d > v_n + w(v_n, v_{n+1})$, entonces la relajación actualiza $v_{n+1}.d = v_n.d + w(v_n, v_{n+1})$ y $v_{n+1}.\pi = v_n$.
  - **Caso 2.1:** Si $v_{n+1} \notin V_\pi$ antes de la relajación, esto significa que $v_{n+1}.\pi =$ NULL y $v_{n+1} \neq s$. De este modo, como tras ejecutarse la relajación $(n+1)-$ésima se tiene que $v_{n+1}.\pi = v_n$, se añade el vértice v_{n+1} a V_π y se actualiza este conjunto, $V'_\pi = V_\pi \cup \{v_{n+1}\}$. Como $v_n \in V_\pi$, por hipótesis de inducción se sabe que existe un camino con origen s y destino v_n y, por tanto, concatenando este camino con la arista $(v_n, v_{n+1}) \in A(G)$ se construye un nuevo camino con origen s y destino v_{n+1}, el cual es $s \rightsquigarrow v_n \to v_{n+1}$.
  - **Caso 2.2:** Si $v_{n+1} \in V_\pi$ antes de la relajación, entonces ya existía un camino con origen s y destino v_{n+1} en G_π y ya se tenía un predecesor para v_{n+1}. Tras la relajación, este predecesor se actualiza a $v_{n+1}.\pi = v_n$. Como $v_n \in V_\pi$, por hipótesis de inducción se sabe que existe un camino con origen s y destino v_n y, por tanto, concatenando este camino con la arista $(v_n, v_{n+1}) \in A(G)$ se construye un nuevo camino con origen s y destino v_{n+1}, el cual es $s \rightsquigarrow v_n \to v_{n+1}$. Aunque se ha reemplazado un camino por otro, no se pierde la conexión desde s hasta v_{n+1}, simplemente se actualiza el camino por uno más corto. En consecuencia, para todo vértice $v \in V_\pi$ (conjunto que no cambia en este caso), sigue existiendo un camino con origen s y destino v en G_π.

Así, queda demostrado por inducción que en G_π, existe un camino con origen s y destino v para cada vértice $v \in V_\pi$ y esta propiedad se mantiene invariante a lo largo de cualquier secuencia de relajaciones. □

Lema 2.10. *Sea $G = (V, A)$ un grafo con aplicación peso $w : A \to \mathbb{R}$, sea $s \in V$ un vértice fuente y supóngase que G no contiene ciclos de peso negativo que sean alcanzables desde s. Entonces, después de que el grafo haya sido inicializado por* INITIALIZE-SINGLE-SOURCE(G,s)*, el subgrafo de predecesores G_π forma un árbol enraizado con raíz s, y cualquier secuencia de pasos de relajación sobre las aristas de G preserva esta propiedad como invariante.*

Demostración. Para empezar, se recuerda al lector que, por definición, G_π formará un árbol enraizado con raíz s si es un grafo no dirigido, conexo y acíclico. Inmediatamente después de la inicialización del grafo G esto se cumple trivialmente ya que hasta ese momento el subgrafo de predecesores G_π sólo cuenta con el vértice fuente.

En lo que sigue, se estudia el subgrafo de predecesores G_π formado tras una secuencia de pasos de relajación, demostrando primero que este subgrafo es **acíclico**. Por reducción

al absurdo se supone que alguno de estos pasos de relajación origina un ciclo en el subgrafo G_π. Sea este ciclo $c = \langle v_0, v_1, \ldots, v_k \rangle$, con $v_k = v_0$, entonces se tienen los predecesores $v_i.\pi = v_{i-1}$ para todo $i \in \{1, 2, \ldots, k\}$ y, sin pérdida de generalidad, se puede suponer que el paso de relajación que ha originado este ciclo es el correspondiente al de la arista $(v_{k-1}, v_k) \in A(G)$. Para ver que cada vértice de este ciclo es alcanzable en G desde el vértice fuente se considera que, a la vez que se le asignó a cada vértice de c un predecesor $v.\pi$ (cuyo valor no es NULL), también se le asignó una estimación finita del peso del camino más corto ($v.d < \infty$). Entonces, aplicando la propiedad de cota superior (Lema 2.2), cada vértice v del ciclo c cumple que $\delta(s, v) \leq v.d < \infty$, de manera que se puede afirmar que estos vértices son alcanzables desde el vértice fuente.

Una vez probada su existencia, se procede a estudiar las estimaciones del peso del camino más corto a cada vértice de c en el paso inmediatamente anterior a la llamada RELAX(v_{k-1},v_k,w). En este paso se tiene que $v_i.\pi = v_{i-1}$ para todo $i \in \{1, 2, \ldots, k-1\}$. Esto implica que para todo $i \in \{1, 2, \ldots, k-1\}$, la última actualización de la estimación $v_i.d$ fue realizada por la asignación $v_i.d = v_{i-1}.d + w(v_{i-1}, v_i)$. Además, teniendo en cuenta que si el valor para la estimación $v_i.d$ sufre algún cambio tras esta asignación, sólo podrá ser una disminución en su valor, en este paso se tiene que

$$v_i.d \geq v_{i-1}.d + w(v_{i-1}, v_i) \quad \forall i \in \{1, 2, \ldots, k-1\}.$$

Entonces, como se sabe que $v_k.\pi$ va a cambiar al realizar la llamada RELAX(v_{k-1},v_k,w), entonces, en el paso inmediatamente anterior en el que se está trabajando, se cumple necesariamente que

$$v_k.d > v_{k-1}.d + w(v_{k-1}, v_k).$$

Así, sumando esta desigualdad estricta con las $k-1$ desigualdades anteriores, se obtiene la siguiente desigualdad para la suma de las estimaciones de los caminos más cortos para los vértices del ciclo c:

$$\sum_{i=1}^{k} v_i.d > \sum_{i=1}^{k} (v_{i-1}.d + w(v_{i-1}, v_i)) = \sum_{i=1}^{k} v_{i-1}.d + \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Sin embargo, como cada vértice del ciclo c aparece exactamente una vez en cada sumatorio, ya que $v_0 = v_k$, se tiene que

$$\sum_{i=1}^{k} v_i.d = \sum_{i=1}^{k} v_{i-1}.d$$

y sustituyendo esta igualdad en la anterior desigualdad se obtiene que

$$0 > \sum_{i=1}^{k} w(v_{i-1}, v_i) = w(c)$$

es decir, c es un ciclo de peso negativo cuyos vértices son alcanzables desde el vértice fuente, contradiciendo la hipótesis de que G no contiene ciclos de peso negativo que sean alcanzables desde s y confirmando así que G_π es un grafo dirigido acíclico. Además, como es acíclico se tiene que todo camino que se tome en G_π será simple.

Ahora, para concluir que este grafo dirigido acíclico es un **árbol enraizado** con raíz s basta con probar, por el Lema 2.8, que para cada uno de sus vértices $v \in V_\pi$ existe un único camino en G_π que comienza en s y termina en v. La existencia de estos caminos, y el hecho de que esta propiedad es invariante ante cualquier secuencia de relajaciones, se

tiene directamente por el Lema 2.9, y la unicidad se prueba a continuación.

Se fija un vértice cualquiera $v \in V_\pi$ y se supone, por reducción al absurdo, que G_π contiene dos caminos distintos con origen s y destino este vértice v. Estos dos caminos se denotan como p_1, que se puede descomponer como $s \rightsquigarrow u \rightsquigarrow x \to z \rightsquigarrow v$ y p_2, que se puede descomponer como $s \rightsquigarrow u \rightsquigarrow y \to z \rightsquigarrow v$, donde $u, x, y, z \in V_\pi$ tales que $x \neq y$ y $(x, z), (y, z) \in A_\pi$.

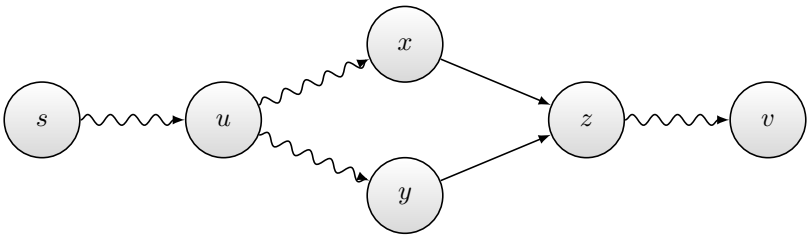


Figura 2.5: Representación de que un camino simple en G_π del vértice fuente s a v es único.

Sin embargo, en este caso se tendría que $z.\pi = x$ y que $z.\pi = y$ lo que implica que $x = y$ (ya que el predecesor de un vértice es único en un árbol de caminos más cortos), obteniéndose así una contradicción y concluyendo que, efectivamente, G_π es un árbol enraizado con raíz s y cualquier secuencia de pasos de relajación sobre las aristas de G preserva esta propiedad como invariante. □

Lema 2.11 (**Propiedad del subgrafo de predecesores**). *Sea $G = (V, A)$ un grafo con aplicación peso $w : A \to \mathbb{R}$, sea $s \in V$ un vértice fuente y supóngase que G no contiene ciclos de peso negativo que sean alcanzables desde s. Entonces, después de la llamada al procedimiento* INITIALIZE-SINGLE-SOURCE(G,s) *y de cualquier secuencia de pasos de relajación sobre las aristas de G que produce $v.d = \delta(s, v)$ para todo $v \in V(G)$, el subgrafo de predecesores G_π es un árbol de caminos más cortos enraizado en s.*

Demostración. Para probar que G_π es un árbol de caminos más cortos enraizado en s, es necesario probar que G_π cumpla los tres puntos de la Definición 1.11:

1. Para ver que $V(G_\pi)$ es el conjunto de vértices en G alcanzables desde s, se recuerda que, por definición, el peso de un camino más corto es finito, es decir, $\delta(s, v) < \infty$, si y sólo si v es alcanzable desde s. Por ello, y como por hipótesis se conoce que $\delta(s, v) = v.d$ para todo $v \in V$, los vértices alcanzables desde s serán aquellos tales que $v.d < \infty$. En las condiciones de este lema, tras la inicialización de los atributos del grafo G se tiene que $v.d = \infty$ para todo $v \in V \setminus \{s\}$, $s.d = 0$ y $v.\pi =$NULL para todo $v \in V$, de modo que aquellos vértices cuyo predecesor $v.\pi$ sea distinto de NULL cumplirán que $v.d < \infty$, ya que ambas asignaciones ocurren en la llamada al procedimiento RELAX. Esto implica que los vértices de G alcanzables desde s serán aquellos tales que $v.\pi \neq NULL$, junto al vértice fuente s, y este conjunto de vértices es, por definición, V_π.

2. Que G_π forma un árbol enraizado cuya raíz es s se acaba de probar en el Lema 2.10.

3. A continuación, se prueba que para cada vértice $v \in V(G_\pi)$, el camino simple único con origen s y destino v en G_π es un camino más corto con origen s y destino v en G.
   Para ello, se fija un vértice $v \in V(G_\pi)$ y se considera el camino más corto con origen s y destino v en G_π, al que se denota $p = \langle v_0, v_1, ..., v_k \rangle$ donde $v_0 = s$ y $v_k = v$.

Se toma una arista (v_{i-1}, v_i) de este camino, es decir $i \in \{1, \ldots, k\}$, arbitrariamente. Como esta arista pertenece al subgrafo G_π, por cómo se define A_π, se sabe que el predecesor óptimo de v_i en el camino más corto es v_{i-1}, luego la última relajación que modificó el valor de $v_i.d$ tuvo que ser la de esta arista.
Tras la relajación de esta arista se tenía que $v_i.d = v_{i-1}.d + w(v_{i-1}, v_i)$. Podría darse el caso de que después se relajase una arista incidente al vértice v_{i-1}, disminuyendo la estimación $v_{i-1}.d$ sin modificar el valor de $v_i.d$ y obteniéndose así que $v_i.d > v_{i-1}.d + w(v_{i-1}, v_i)$. De este modo, combinando estos dos casos, se tiene que $v_i.d \geq v_{i-1}.d + w(v_{i-1}, v_i)$ para cualquier $i \in \{1, \ldots, k\}$. Además, sustituyendo la hipótesis de que $v.d = \delta(s, v)$ para todo $v \in V(G)$, concretamente $v_i.d = \delta(s, v_i)$ para todo $i \in \{1, \ldots, k\}$, en esta desigualdad se tiene que $\delta(s, v_i) \geq \delta(s, v_{i-1}) + w(v_{i-1}, v_i)$ y, reordenando, $\delta(s, v_i) - \delta(s, v_{i-1}) \geq w(v_{i-1}, v_i)$ para todo $i \in \{1, \ldots, k\}$. Conociendo esto, sumando los pesos de las aristas del camino p se obtiene que:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \leq \sum_{i=1}^{k} (\delta(s, v_i) - \delta(s, v_{i-1}))$$

$$= (\delta(s, v_1) - \delta(s, v_0)) + (\delta(s, v_2) - \delta(s, v_1)) + \cdots + (\delta(s, v_{k-1}) - \delta(s, v_{k-2}))$$
$$+(\delta(s, v_k) - \delta(s, v_{k-1})) = \delta(s, v_k) - \delta(s, v_0) = \delta(s, v) - \delta(s, s) = \delta(s, v),$$

es decir, $w(p) \leq \delta(s, v)$. Y como, por definición, $\delta(s, v)$ es cota inferior para el peso de cualquier camino con origen s y destino v en G, se tiene que $\delta(s, v) \leq w(p)$. Entonces, por doble desigualdad, se tiene que $w(p) = \delta(s, v)$, concluyendo así que p es un camino más corto con origen s y destino v en G.

□

Capítulo 3

El algoritmo de Bellman-Ford

El algoritmo de Bellman-Ford, desarrollado por Richard Ernest Bellman y Lester Randolph Ford Jr., permite resolver el problema de los caminos más cortos de fuente única para cualquier grafo dado.

A lo largo de este capítulo se presenta este algoritmo, describiendo su funcionamiento y mostrando un ejemplo de ejecución sobre un grafo concreto, se demuestra por qué este algoritmo es capaz de resolver problemas de caminos más cortos y, finalmente, se expone una mejora para este algoritmo, la mejora de Yen. Se han tomado como principales referencias la Sección 22.1 del libro *Introduction to Algorithms* [6], junto con la Sección 4.4 del libro *Algorithms* [13] y la referencia [3]. Tanto el Corolario 3.2 como los Lemas 3.6 y 3.7 han sido probados por la autora, por cuenta propia.

3.1. Descripción del algoritmo

En la asignatura Matemática Discreta se estudia el algoritmo de Dijkstra: un algoritmo que resuelve el problema de los caminos más cortos de fuente única para grafos en los que todas sus aristas tengan pesos no negativos. Sin embargo, como se ha ejemplificado en la Sección 1.4 de este trabajo, para poder resolver determinados problemas de rutas es necesario que existan algoritmos que puedan recibir como entrada grafos con aristas de pesos negativos. Es por ello que, en este trabajo, se presenta el algoritmo de Bellman-Ford, capaz de resolver el problema de los caminos más cortos de fuente única para cualquier tipo de grafo. Este algoritmo recibe como entrada los siguientes parámetros:

- G: grafo dirigido ponderado $G = (V, A)$.
- w: aplicación peso que asigna a cada arista $(u, v) \in A(G)$ un valor real.
- s: vértice fuente que actúa como el origen de los caminos más cortos a encontrar con destino los vértices restantes del grafo.

En la Observación 1.17 se justificó que el peso de los caminos más cortos no está bien definido para los grafos que contienen ciclos de peso negativo alcanzables desde la fuente. Por este motivo, el algoritmo de Bellman-Ford está diseñado para identificar la existencia de dichos ciclos, retornando FALSE si existe al menos uno y TRUE si no hay ninguno.

El diseño del algoritmo es el que sigue:

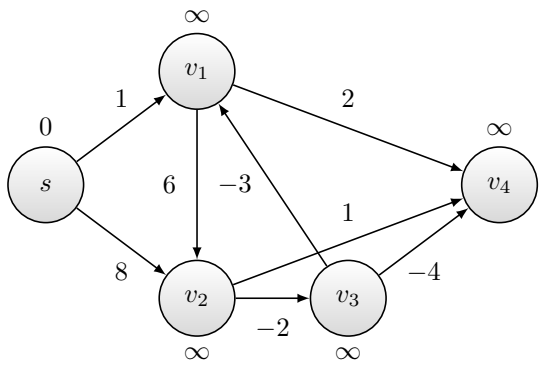
Algoritmo BELLMAN-FORD(G, w, s)
1: INITIALIZE-SINGLE-SOURCE(G, s)
2: **Desde** $i = 1$ **hasta** $|V(G)| - 1$ **hacer**
3: **Para** cada arista (u, v) de $A(G)$ **hacer**
4: RELAX(u, v, w)
5: **Fin para**
6: **Fin desde**
7: **Para** cada arista (u, v) de $A(G)$ **hacer**
8: **Si** $v.d > u.d + w(u, v)$ **entonces**
9: **Devuelve** FALSE
10: **Fin para**
11: **Devuelve** TRUE
Fin algoritmo

El algoritmo comienza realizando una llamada a INITIALIZE-SINGLE-SOURCE(G,s) (ver Sección 1.5), la cual asigna un valor inicial a la estimación del peso del camino más corto, $v.d$, y al predecesor, $v.\pi$, de cada vértice $v \in V(G)$. Una vez inicializados estos atributos, el algoritmo realiza $|V(G)| - 1$ iteraciones, en cada una de las cuales se aplica la técnica de relajación RELAX(u,v,w) (ver Sección 1.5) a todas las aristas $(u, v) \in A(G)$. Así, en cada iteración, se actualizan los valores de los atributos $v.d$ y $v.\pi$ para cada vértice $v \in V(G) \setminus \{s\}$ conforme a los pesos $w(u, v) \in \mathbb{R}$ definidos en el grafo. Es más, para cada vértice $v \in V(G) \setminus \{s\}$, el valor de $v.d$ converge de manera iterativa hacia el peso, $\delta(s, v)$, del camino más corto desde la fuente s hasta ese vértice y se establece la correspondiente secuencia de predecesores asociados a dicho camino más corto.

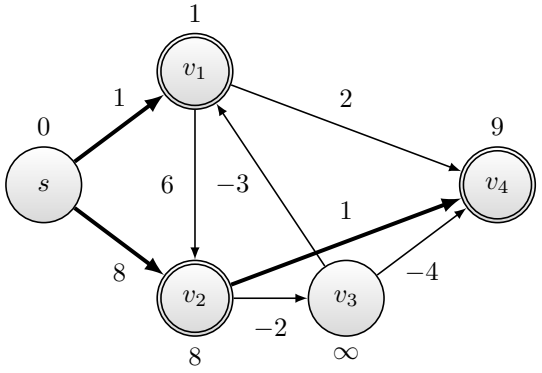
Posteriormente, para detectar la existencia de ciclos de peso negativo alcanzables desde s en el grafo dado como parámetro, el algoritmo ejecuta una verificación adicional para cada arista $(u, v) \in A(G)$ que comprueba si la desigualdad $v.d \leq u.d + w(u, v)$ deja de cumplirse, es decir, si $v.d > u.d + w(u, v)$. Si se da esta desigualdad estricta para alguna arista, no se está bajo las condiciones del Lema 2.4 y por ende no se puede afirmar que se cumpla la propiedad de convergencia (Lema 2.5) para dicha arista, señalando así la presencia de algún ciclo de peso negativo en el grafo. En este caso, el algoritmo devuelve FALSE, indicando que no hay solución válida. Si por contra no existe ninguna arista $(u, v) \in A(G)$ que reduzca el valor de $v.d$, sí se cumple la propiedad de convergencia (Lema 2.5) para toda arista, por lo que el algoritmo retorna TRUE.

Notación 1. Para facilitar la escritura, en lo que sigue de capítulo se llama bucle principal del algoritmo de Bellman-Ford al bucle que ocupa las líneas 2-6 en BELLMAN-FORD.

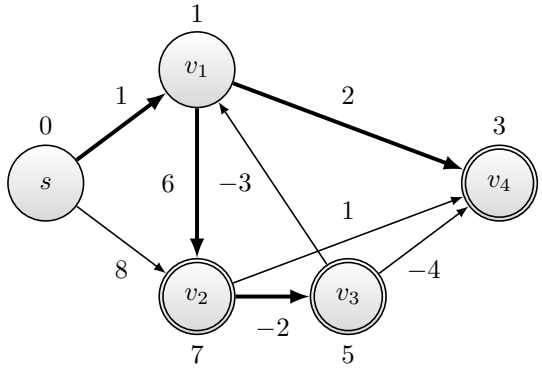
En la Figura 3.1 se muestra la ejecución del algoritmo BELLMAN-FORD sobre un grafo con pesos negativos que cuenta con un ciclo $\langle v_1, v_2, v_3, v_1 \rangle$ de peso positivo. El vértice fuente de este grafo es s. En este ejemplo se supone que cada iteración relaja todas las aristas del grafo en el siguiente orden: $(v_3, v_4), (v_1, v_2), (v_3, v_1), (v_1, v_4), (s, v_1), (v_2, v_3), (s, v_2), (v_2, v_4)$, el cual ha sido tomado arbitrariamente.



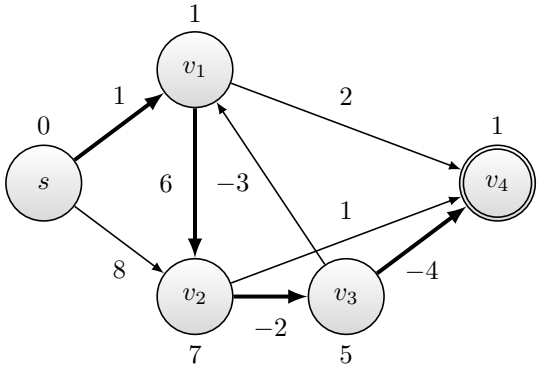
(a) Tras la llamada a INITIALIZE-SINGLE-SOURCE



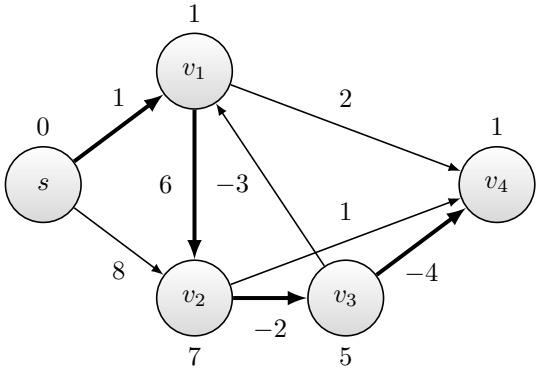
(b) Tras la primera iteración



(c) Tras la segunda iteración



(d) Tras la tercera iteración



(e) Tras la cuarta iteración

Figura 3.1: Ejecución del algoritmo de Bellman-Ford. Tras cada iteración del bucle principal, las estimaciones del peso del camino más corto a cada vértice se muestran como etiqueta de cada vértice. Las aristas (u, v) resaltadas con mayor grosor representan los predecesores determinados hasta el momento, indicando que $v.\pi = u$. Además, los vértices marcados con doble círculo son aquellos cuyos atributos $v.d$ y $v.\pi$ han sido actualizados con respecto a la iteración previa.

Una vez inicializado el grafo, como cuenta con cinco vértices, el algoritmo realiza cuatro iteraciones en el bucle principal. En la primera y segunda iteración, Figuras 3.1(b) y 3.1(c), el algoritmo va buscando los caminos más cortos a cada vértice v, modificando los valores de los atributos $v.d$ y $v.\pi$. Tras la tercera iteración, como se puede observar en la Figura 3.1(d), ya se alcanzan los valores óptimos para la estimación del peso del camino más corto y para el predecesor de cada vértice. Por este motivo, como indica la propiedad de relajación de caminos (Lema 2.6), estos valores no se ven modificados en la cuarta

iteración, Figura 3.1(e). En el siguiente bucle del algoritmo, todas las aristas (u, v) del grafo cumplen que $v.d \leq u.d + w(u, v)$, luego, como se aprecia en la Figura 3.1, no se encuentran ciclos de peso negativo alcanzables desde la fuente, y el algoritmo retorna TRUE.

3.2. Corrección del algoritmo

El objetivo de esta sección es demostrar, usando varios resultados del Capítulo 2, la corrección del algoritmo de Bellman-Ford como herramienta para resolver el problema de los caminos más cortos de fuente única. Para ello, resulta necesario probar previamente el Lema 3.1, el cual formaliza el comportamiento convergente de las estimaciones del peso de los caminos más cortos a cada vértice, observado en la Figura 3.1, a lo largo de las iteraciones del bucle principal del algoritmo.

Lema 3.1. *Sea $G = (V, A)$ un grafo con vértice fuente s y aplicación peso $w : A \to \mathbb{R}$. Si G no contiene ciclos de peso negativo alcanzables desde s, entonces, tras las $|V| - 1$ iteraciones del bucle principal del algoritmo* BELLMAN-FORD*, se tiene que $v.d = \delta(s, v)$ para todos los vértices $v \in V$ alcanzables desde s.*

Demostración. Se fija un vértice $v \in V$ cualquiera que sea alcanzable desde s, y se considera $\langle v_0, v_1, \ldots, v_k \rangle$, con $v_0 = s$, $v_k = v$ y $k \in \mathbb{N}$, un camino más corto con origen s y destino v en el grafo G. En la Sección 1.4 se concluyó que el número de aristas para cualquier camino más corto en un grafo con ciclos de peso negativo no alcanzables desde el vértice fuente es, a lo sumo, $|V| - 1$, luego $k \leq |V| - 1$.

Tras inicializar el grafo, el algoritmo BELLMAN-FORD ejecuta $|V| - 1$ iteraciones del bucle principal, donde cada iteración relaja todas las aristas $(u, v) \in A$ mediante RELAX(u, v, w). Por la propiedad de relajación de caminos (Lema 2.6), si las aristas $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ se relajan en orden, entonces $v_k.d = \delta(s, v_k)$ independientemente de las relajaciones que ocurran sobre otras aristas del grafo. Aunque BELLMAN-FORD no sigue este orden, las $|V| - 1$ iteraciones garantizan que, en algún momento durante dichas iteraciones, se habrá relajado cada arista (v_{i-1}, v_i) (con $i \in \{1, \ldots, k\}$) del camino más corto cumpliendo $v_{i-1}.d = \delta(s, v_{i-1})$ gracias a iteraciones previas, asegurando que $v_i.d = \delta(s, v_i)$. Por este motivo, tras las $|V| - 1$ iteraciones, al aplicar la propiedad de relajación de caminos se obtiene que $v.d = v_k.d = \delta(s, v_k) = \delta(s, v)$. Como v es arbitrario, el lema queda probado para todos los vértices de G alcanzables desde s. □

Basándose en este lema, el siguiente corolario señala la relación encontrada entre la obtención, al finalizar la ejecución de BELLMAN-FORD, de un valor finito para la estimación del peso de los caminos más cortos a un vértice y la existencia de un camino con destino dicho vértice y origen el vértice fuente del grafo.

Corolario 3.2. *Sea $G = (V, A)$ un grafo con vértice fuente s y aplicación peso $w : A \to \mathbb{R}$. Entonces, para cada vértice $v \in V$, existe un camino con origen s y destino v si y sólo si el algoritmo* BELLMAN-FORD *finaliza con $v.d < \infty$ al ser ejecutado sobre G.*

Demostración. Se fija un vértice cualquiera $v \in V$ y se demuestran ambas implicaciones de este corolario.

$\boxed{\Rightarrow}$ Se supone que existe un camino con origen s y destino v luego, por definición de peso de un camino más corto, se tiene que $\delta(s, v) < \infty$. Se consideran dos casos:

Caso 1: Si G no contiene ciclos de peso negativo alcanzables desde s, se cumplen las hipótesis del lema anterior. Entonces, como para esta implicación se ha supuesto que v es alcanzable desde s, por el Lema 3.1 se tiene que tras las $|V|-1$ iteraciones del bucle principal de BELLMAN-FORD se da la igualdad $v.d = \delta(s,v) < \infty$. Así, se puede afirmar que al finalizar BELLMAN-FORD se obtiene que $v.d < \infty$.

Caso 2: Si G contiene ciclos de peso negativo alcanzables desde s entonces, por la Observación 1.17, se tiene que $\delta(s,v) = -\infty$, cumpliéndose que $\delta(s,v) < \infty$ al finalizar BELLMAN-FORD.

$\boxed{\Leftarrow}$ Se supone que al finalizar la ejecución de BELLMAN-FORD se tiene que $v.d < \infty$. Por la propiedad de cota superior (Lema 2.2) se sabe que $\delta(s,v) \leq v.d$ luego, necesariamente, $\delta(s,v) < \infty$ y, por definición, existe un camino con origen s y destino v.

Por todo esto, se puede afirmar que existe un camino con origen s y destino v si y sólo si $v.d < \infty$ al finalizarse la ejecución de BELLMAN-FORD. □

Teorema 3.3 (**Corrección del algoritmo de Bellman-Ford**). *Sea* BELLMAN-FORD *ejecutado sobre un grafo $G = (V, A)$ con vértice fuente s y aplicación peso $w : A \to \mathbb{R}$. Si G no contiene ciclos de peso negativo alcanzables desde s, entonces $v.d = \delta(s,v)$ para todo vértice $v \in V$, el subgrafo de predecesores G_π es un árbol de caminos más cortos enraizado en s y el algoritmo retorna* TRUE. *Por otro lado, en el caso de que G sí contenga, por lo menos, un ciclo de peso negativo alcanzable desde s, entonces el algoritmo retorna* FALSE.

Demostración. Por un lado, se estudia el caso en el que G no contiene ciclos de peso negativo alcanzables desde s. Hay que probar las tres afirmaciones:

1. Se fija un vértice $v \in V$ cualquiera y se dan dos casos. En ambos casos se verifica la igualdad $v.d = \delta(s,v)$:
   Caso 1: si v es alcanzable desde s, por el Lema 3.1 se tiene que $v.d = \delta(s,v)$.
   Caso 2: si por contra v no es alcanzable desde s, no existirá un camino con origen s y destino v, luego, por la propiedad de inexistencia de camino (Corolario 2.3), se tiene que $v.d = \delta(s,v) = \infty$.
   Luego, como v era un vértice cualquiera, se tiene que $v.d = \delta(s,v)$ para todo $v \in V$.

2. Como se ha supuesto que G no contiene ciclos de peso negativo alcanzables desde s, se está en las condiciones para aplicar la propiedad del subgrafo de predecesores (Lema 2.11), obteniéndose de manera inmediata que G_π es un árbol de caminos más cortos enraizado en s.

3. Tras la ejecución del algoritmo BELLMAN-FORD, para toda arista $(u,v) \in A$ se tiene que:
   $$v.d \overset{\text{Afirmación 1.}}{=} \delta(s,v) \overset{\text{D.T.}}{\leq} \delta(s,u) + w(u,v) \overset{\text{Afirmación 1.}}{=} u.d + w(u,v)$$
   de modo que la desigualdad del segundo bucle (línea 8) de este algoritmo nunca se cumple y entonces el algoritmo nunca retorna FALSE y, por como está diseñado, retorna TRUE.

Por otro lado, se estudia el caso en el que G contiene, por lo menos, un ciclo de peso negativo alcanzable desde s. Sea este ciclo $c = \langle v_0, \ldots, v_k \rangle$ con $v_0, \ldots, v_k \in V$, $v_0 = v_k$ y

$$w(c) = \sum_{i=1}^{k} w(v_{i-1}, v_i) < 0.$$

Para este caso hay que probar que el algoritmo BELLMAN-FORD retorna FALSE. Para ello, se supone, por reducción al absurdo, que el algoritmo retorna TRUE. Bajo esta hipótesis, se tendría que $v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$ para cada $i \in \{1, \ldots, k\}$ y, sumando estas k desigualdades se obtiene:

$$\sum_{i=1}^{k} v_i.d \leq \sum_{i=1}^{k} (v_{i-1}.d + w(v_{i-1}, v_i)) = \sum_{i=1}^{k} v_{i-1}.d + \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

Además, como cada vértice del ciclo c aparece exactamente una vez en cada sumatorio, ya que $v_0 = v_k$, se tiene que

$$\sum_{i=1}^{k} v_i.d = \sum_{i=1}^{k} v_{i-1}.d$$

y por el Corolario 3.2 se tiene que $v_i.d < \infty$ para $i \in \{1, 2, \ldots, k\}$. Entonces, sustituyendo esta igualdad en la anterior desigualdad se obtiene que

$$0 \leq \sum_{i=1}^{k} w(v_{i-1}, v_i) = w(c)$$

contradiciendo así la hipótesis de que c es un ciclo de peso negativo. □

3.3. La mejora de Yen para el algoritmo de Bellman-Ford

En esta sección se analiza la mejora propuesta por Yen para el algoritmo de Bellman-Ford, a partir de un problema del Capítulo 22 del libro *Introduction to Algorithms* [6]. Para su desarrollo, también se ha consultado el Capítulo 20 de dicha obra y las referencias [10] y [2].

El algoritmo de Bellman-Ford no especifica el orden en el que las aristas han de ser relajadas durante cada iteración de su bucle principal. Al aplicar este algoritmo al grafo correspondiente a la Figura 3.1 se ha visto que la última iteración del bucle principal no modifica, con respecto a la iteración anterior, los valores de los atributos $v.d$ y $v.\pi$ para ningún vértice v del grafo. En esta sección se propone establecer un orden en el conjunto de aristas del grafo para guiar el proceso de relajación, con el fin de reducir el número de iteraciones del bucle principal del algoritmo de Bellman-Ford necesarias para calcular los caminos más cortos.

En 1970, Jin Y. Yen describió una mejora para el algoritmo de Bellman-Ford. Esta mejora sigue el diseño del algoritmo de Bellman-Ford expuesto al principio de este capítulo. Tras inicializar los valores de los atributos, y antes de realizar la primera iteración del bucle principal del algoritmo, se asigna un orden lineal arbitrario a los vértices del grafo $G = (V, A)$ dado como entrada al algoritmo. Después, Yen propone dividir el conjunto A de aristas del grafo en dos subconjuntos: A_f y A_b. Yen define cada uno de estos subconjuntos de la siguiente manera:

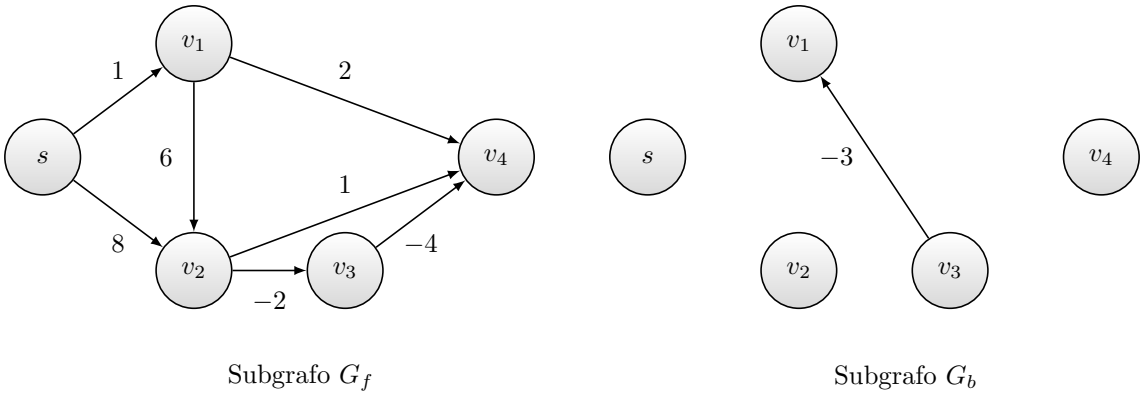
$$A_f = \{(v_i, v_j) \in A : i < j\},$$
$$A_b = \{(v_i, v_j) \in A : i > j\},$$

de modo que $A = A_f \cup A_b$. Además, supone que G no contiene lazos; luego, cada arista del grafo pertenece o bien a A_f o bien a A_b, es decir, se tiene que $A_f \cap A_b = \emptyset$. Conocidos estos subconjuntos, se pueden considerar los subgrafos $G_f = (V, A_f)$ y $G_b = (V, A_b)$.

Ejemplo 3.4. *Para poder dividir las aristas del grafo que aparece en la Figura 3.1 en los dos subconjuntos A_f y A_b definidos por Yen, es necesario considerar que $s = v_0$. Al hacerlo, se obtienen los dos subconjuntos siguientes:*

$$A_f = \{(v_3, v_4), (v_1, v_2), (v_1, v_4), (s, v_1), (v_2, v_3), (s, v_2), (v_2, v_4)\},$$
$$A_b = \{(v_3, v_1)\},$$

junto con los dos subgrafos siguientes:



Subgrafo G_f Subgrafo G_b

Figura 3.2: División del grafo de la Figura 3.1 en dos subgrafos G_f y G_b siguiendo las definiciones para los subconjuntos A_f y A_b tomadas por Yen

Como se puede observar, estos subgrafos son acíclicos, esto es, al realizar la división de las aristas del grafo inicial (el cual contenía el ciclo $\langle v_1, v_2, v_3, v_1\rangle$) en los dos subconjuntos definidos por Yen, el ciclo no aparece en ninguno de los subgrafos correspondientes.

Definición 3.5. *Una **ordenación topológica** para un grafo dirigido acíclico $G = (V, A)$ es una relación de orden total para los vértices de G tal que, dada una arista $(u, v) \in G$, el vértice u aparece antes que el vértice v en la ordenación.*

En el subgrafo G_f del ejemplo anterior las aristas se pueden ordenar topológicamente de una sola forma, esta es, $\langle s, v_1, v_2, v_3, v_4\rangle$. Sin embargo, las aristas del subgrafo G_b cuentan con varias posibles ordenaciones topológicas, como por ejemplo $\langle v_2, s, v_3, v_4, v_1\rangle$ o $\langle v_4, v_3, v_2, v_1, s\rangle$. Nótese que esta última es la inversa de la única ordenación topológica encontrada para las aristas de G_f.

El siguiente lema prueba que los subgrafos G_f y G_b definidos por Yen son siempre acíclicos y que, independientemente de las aristas del grafo, el orden $v_1, v_2, \ldots, v_{|V|}$ va a ser siempre una ordenación topológica.

Lema 3.6. *Sea G un grafo sin lazos, se tiene que los subgrafos correspondientes G_f y G_b son acíclicos, con ordenación topológica $\langle v_1, v_2, \ldots, v_{|V|}\rangle$ y $\langle v_{|V|}, v_{|V|-1}, \ldots, v_1\rangle$ respectivamente.*

Demostración. Primero, se prueba que G_f es acíclico por reducción al absurdo. Se supone que existe un ciclo $\langle v_{i_1}, v_{i_2}, \ldots, v_{i_k}\rangle$ en G_f, donde $v_{i_1}, v_{i_2}, \ldots, v_{i_k} \in V$, $k \in \mathbb{N}$ y $v_{i_1} = v_{i_k}$. Considerando la siguiente secuencia para las aristas que conforman este ciclo: $(v_{i_1}, v_{i_2}), (v_{i_2}, v_{i_3}), \ldots, (v_{i_{k-1}}, v_{i_k})$, y, como todas estas aristas pertenecen a A_f, se cumple la siguiente serie de desigualdades: $i_1 < i_2 < i_3 < \ldots < i_{k-1} < i_k = i_1$. De este modo, se tendría la desigualdad estricta $i_1 < i_1$, llegando a un absurdo. Luego, G_f no puede

contener ciclos y, por definición, es acíclico. Una vez probado que es acíclico, se puede pasar a demostrar que $\langle v_1, v_2, \ldots, v_{|V|}\rangle$ es una ordenación topológica. Esto es inmediato, ya que, como los índices de dicha secuencia están ordenados de forma creciente y como toda arista $(v_i, v_j) \in A_f$ satisface, por definición, que $i < j$, entonces se tiene que v_i aparece antes que v_j en el orden. Así, la secuencia propuesta satisface la definición de ordenación topológica para el grafo dirigido acíclico G_f.

Análogamente se prueba que G_b es acíclico con ordenación topológica $\langle v_{|V|}, v_{|V|-1}, \ldots, v_1\rangle$. Por reducción al absurdo, se supone que existe un ciclo $\langle v_{i_1}, v_{i_2}, \ldots, v_{i_k}\rangle$ en G_b, donde $v_{i_1}, v_{i_2}, \ldots, v_{i_k} \in V$, $k \in \mathbb{N}$ y $v_{i_1} = v_{i_k}$. Considerando la siguiente secuencia para las aristas que conforman este ciclo: $(v_{i_1}, v_{i_2}), (v_{i_2}, v_{i_3}), \ldots, (v_{i_{k-1}}, v_{i_k})$, y, como todas estas aristas pertenecen a A_b, se cumple la siguiente serie de desigualdades: $i_1 > i_2 > i_3 > \ldots > i_{k-1} > i_k = i_1$. De este modo, se volvería a tener la desigualdad estricta $i_1 > i_1$, llegando a un absurdo. Luego, G_b es acíclico y se puede pasar a demostrar que $\langle v_{|V|}, v_{|V|-1}, \ldots, v_1\rangle$ es una ordenación topológica. Esto vuelve a ser inmediato, ya que, como los índices de dicha secuencia están ordenados de forma decreciente y como toda arista $(v_i, v_j) \in A_b$ satisface, por definición, que $i > j$, entonces se tiene que v_i aparece antes que v_j en la ordenación, cumpliéndose así la definición de ordenación topológica. □

Una vez conocidas estas definiciones para los subconjuntos de aristas A_f y A_b, se explica cómo propone Yen que se relajen las aristas en el bucle principal del algoritmo de Bellman-Ford.

Primero, se visitan todos los vértices del grafo siguiendo el orden topológico $v_1, v_2, \ldots, v_{|V|}$, relajándose así las aristas de A_f. Después, se vuelven a visitar todos los vértices pero en el orden topológico inverso, $v_{|V|}, v_{|V|-1}, \ldots, v_1$, relajándose esta vez las aristas de A_b.

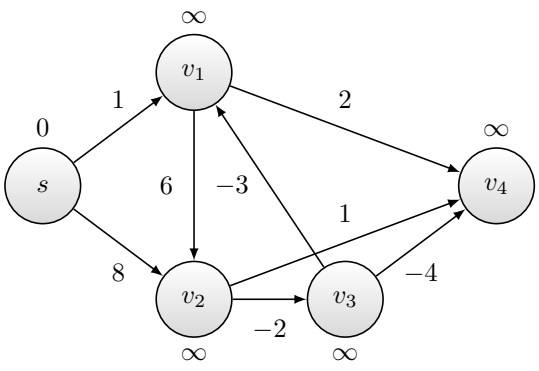
Lema 3.7. *Bajo estas indicaciones, si el grafo G no contiene ciclos de peso negativo que sean alcanzables desde el vértice fuente s, entonces, después de $\lceil |V|/2 \rceil$ iteraciones sobre las aristas, se tiene que $v.d = \delta(s, v)$ para todos los vértices $v \in V$ alcanzables desde s.*

Demostración. Se fija un vértice $v \in V$ cualquiera, que sea alcanzable desde s, y se considera $p = \langle v_{i_1}, v_{i_2}, \ldots, v_{i_k}\rangle$, con $v_{i_1} = s$, $v_{i_k} = v$ y $k \in \mathbb{N}$, un camino más corto con origen s y destino v en el grafo G. En la sección 1.4 se concluyó que el número de aristas para cualquier camino más corto en un grafo con ciclos no alcanzables desde el vértice fuente, en este caso para el camino p, es a lo sumo $|V| - 1$, luego $i_k \leq |V| - 1$.

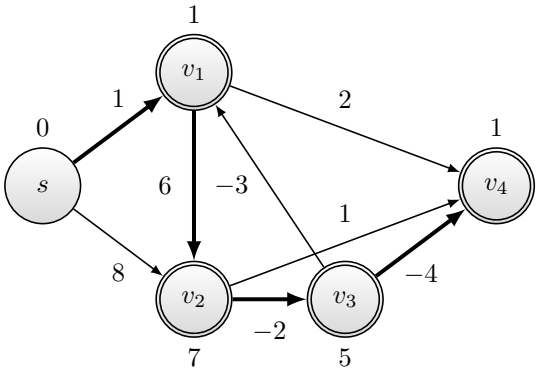
Cada arista $(v_{i_j}, v_{i_{j+1}})$ del camino p, con $j \in \{1, 2, \ldots, k-1\}$, pertenece o bien al conjunto A_f, si $i_j < i_{j+1}$, o bien al conjunto A_b, si $i_j > i_{j+1}$. La secuencia $i_1, i_2, \ldots, i_k$ alterna entre incrementos y decrementos. Cada iteración del bucle principal procesa primero las aristas de A_f y luego las aristas de A_b, tomando secuencias de índices que aumentan o disminuyen, respectivamente. Una secuencia simple de $|V| - 1$ vértices puede cambiar de dirección un máximo de $\lceil |V| - 1/2 \rceil$ veces, ya que cada cambio requiere al menos dos vértices consecutivos. No obstante, si $s = v_{i_1}$ tiene un índice $i_1 > 1$ (si no aparece primero en el orden) , la primera iteración sobre las aristas de A_f puede no inicializar correctamente las estimaciones, requiriendo una iteración adicional. Por lo tanto, se necesitan $\lceil |V|/2 \rceil$ iteraciones para cubrir todos los cambios de dirección y el caso especial de s. Por este motivo, tras las $\lceil |V|/2 \rceil$ iteraciones, al aplicar la propiedad de relajación de caminos (Lema 2.6) se obtiene que $v.d = v_k.d = \delta(s, v_k) = \delta(s, v)$. Como v es arbitrario, el lema queda probado para todos los vértices de G alcanzables desde s. □

Finalmente se muestra un ejemplo de cómo se ejecutaría el algoritmo de Bellman-Ford

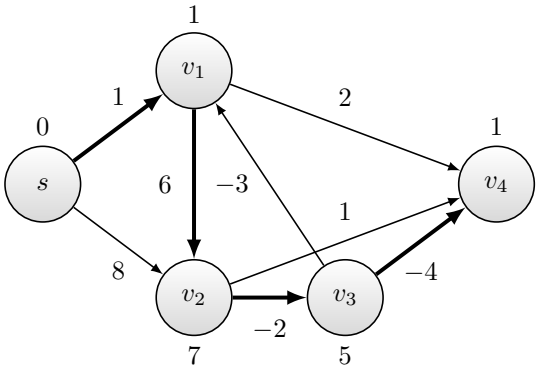
con la mejora de Yen sobre el grafo presentado anteriormente en la Figura 3.1. Aplicando esta mejora, primero se relajan las aristas de A_f siguiendo su ordenación topológica $\langle s, v_1, v_2, v_3, v_4\rangle$ y seguidamente las aristas de A_b siguiendo su ordenación topológica $\langle v_4, v_3, v_2, v_1, s\rangle$. Es decir, el orden de relajación de las aristas en cada iteración del bucle principal es $(s, v_1), (s, v_2), (v_1, v_2), (v_1, v_4), (v_2, v_3), (v_2, v_4), (v_3, v_4), (v_3, v_1)$.



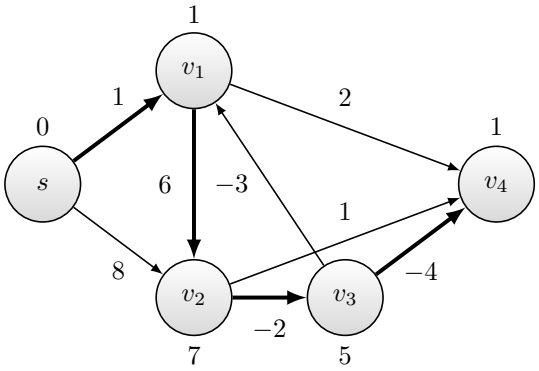
(a) Tras la llamada a INITIALIZE-SINGLE-SOURCE



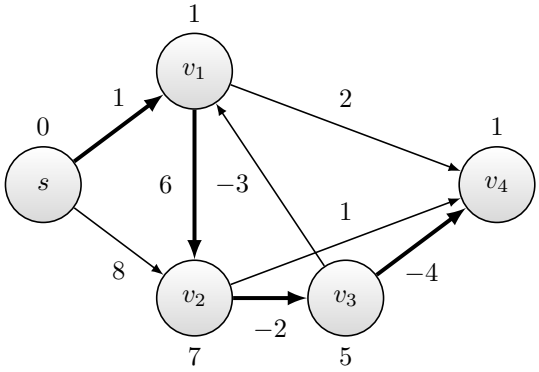
(b) Tras la primera iteración



(c) Tras la segunda iteración



(d) Tras la tercera iteración



(e) Tras la cuarta iteración

Figura 3.3: Ejecución del algoritmo de Bellman-Ford modificado con la mejora de Yen.

Se observa que, con esta mejora, los valores óptimos para las estimaciones del peso del camino más corto y para los predecesores de cada vértice se alcanzan tras la primera iteración, Figura 3.3(b). Además, por la propiedad de convergencia (Lema 2.5) se sabe que estos valores no se verán modificados a lo largo de las iteraciones restantes del bucle.

Se recuerda al lector que, cuando G no tiene ciclos de peso negativo alcanzables desde s, se puede asegurar que las estimaciones óptimas de los pesos de los caminos más cortos se obtienen tras las $|V| - 1$ iteraciones del bucle principal del algoritmo de Bellman-Ford. La mejora de Yen optimiza esto a $\lceil |V|/2 \rceil$ iteraciones.

Capítulo 4

Bellman-Ford en la programación lineal

En este capítulo se muestra la conexión entre la búsqueda en grafos y la programación lineal, tratando un caso especial de programación lineal que se reduce a encontrar caminos más cortos de fuente única.

Primero, se estudia un tipo de problema de programación lineal en el cual todas las restricciones del conjunto de puntos admisibles son de desigualdad. Después, estas restricciones se modelan mediante un grafo de restricciones, cuya estructura permite transformar el problema de resolver el sistema de restricciones en el de buscar el peso de los caminos más cortos. Por último, se explica que el algoritmo de Bellman-Ford que resuelve el problema de los caminos más cortos proporciona una solución para dicho sistema de restricciones.

Se ha tomado como referencia la Sección 22.4 del libro *Introduction to Algorithms* [6]. La autora ha probado por cuenta propia la Observación 4.5, la Observación 4.7, el Lema 4.9 y el Lema 4.10.

4.1. Preliminares de programación lineal

Para empezar, se recuerdan unas nociones básicas con respecto a problemas de optimización, las cuales han sido tomadas de los apuntes de la asignatura Optimización I del Grado en Matemáticas de la Universidad de Cantabria [11].

Se considera la siguiente formulación para un **problema de optimización en dimensión finita**: dados $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ y $K \subset \mathbb{R}^n$, el problema consistente en

$$\text{hallar } \overline{x} \in K \text{ tal que } f(x) \leq f(\overline{x}), \text{ para todo } x \in K;$$

será formulado en la forma

$$(P) \begin{cases} \text{Minimizar } f(x) \\ \\ \text{sujeto a } x \in K \end{cases}$$

De este modo, un problema de optimización se formula a partir de un conjunto de variables independientes, x, sobre las que pueden existir algunas condiciones ($x \in K$), llamadas **restricciones del problema**, y una función dependiendo de las variables que se quiere minimizar o maximizar, llamada **función objetivo**. Al conjunto K se le llama

conjunto de puntos admisibles.

Si $K = \mathbb{R}^n$, entonces se dice que P es un problema de optimización sin restricciones. Sin embargo, en este trabajo se estudian los **problemas de optimización con restricciones**, esto es, aquellos en los que el conjunto de puntos admisibles es de la forma

$$K = \{x \in \mathbb{R}^n : \ h_i(x) = 0, i = 1, \ldots, n_i; \ g_j(x) \leq 0, j = 1, \ldots, n_d\},$$

donde $h_i, g_j : \mathbb{R}^n \longrightarrow \mathbb{R}$. Dentro de este grupo se encuentran los llamados problemas con restricciones lineales, en los que se verifica que K es de la forma:

$$K = \{x \in \mathbb{R}^n : \ c_j^T x = b_j, j = 1, \ldots, n_i; \ c_j^T x \leq b_j, j = n_i + 1, \ldots, n_i + n_d\},$$

donde $c_j \in \mathbb{R}^n$ y $b_j \in \mathbb{R}$.

Notación. Para aclarar la notación, a las matrices se les denotará C, evitando así confundirlas con el conjunto de aristas A de un grafo $G = (V, A)$ con el que se ha estado trabajando a lo largo del trabajo.

Entre los problemas con restricciones lineales, este capítulo se centra en los **problemas de programación lineal**, en los que, además de ser las restricciones lineales, la función objetivo f es lineal, es decir,

$$f(x) = d^T x \ \text{ con } \ d \in \mathbb{R}^n.$$

De este modo, la programación lineal se ocupa de la minimización (o maximización) de una función lineal donde las variables están sujetas a restricciones lineales, que pueden ser tanto de igualdad como de desigualdad.

Observación 4.1. *Un problema de maximización puede formularse en términos de minimización ya que* $\text{máx}\{f(x) : x \in K \subset \mathbb{R}^n\} = -\,\text{mín}\{-f(x) : x \in K \subset \mathbb{R}^n\}$.

Observación 4.2. *En este contexto, el término programación (originario de la traducción directa de mathematical programming) equivale a optimización. La palabra programación ha de entenderse como equivalente a planificación, no a codificación.*

El método más usado para resolver problemas de programación lineal es el método del Simplex, estudiado en la asignatura Optimización I. En este capítulo, se estudia otro método que resuelve los problemas de programación lineal cuyas restricciones son de desigualdad, y concretamente, de la forma $x_i - x_j \leq b_k$ donde $1 \leq i, j \leq n$, $i \neq j$, y $1 \leq k \leq m$ (en inglés, las desigualdades de esta forma son conocidas como *difference constraints*).

Definición 4.3. *Un* ***sistema de difference constraints*** *es aquel en el que cada fila de la matriz de programación lineal C cuenta con un 1 y un -1, y el resto de entradas en C son ceros. De este modo, las restricciones dadas por el sistema $Cx \leq b$ son un conjunto de m difference constraints que contienen n variables.*

Notación. En lo que sigue de capítulo cada vez que se escribe que $Cx \leq b$ es un **sistema de restricciones de desigualdad** se supone que, en concreto, se trata de un sistema de *difference constraints*.

A continuación se plantea el problema consistente en encontrar un vector $x \in \mathbb{R}^5$ que resuelva el siguiente sistema de restricciones de desigualdad cuyo vector $b \in \mathbb{R}^8$ está formado por los valores de los pesos de las aristas del grafo de la Figura 3.1:

$$\begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 1 \\ -3 \\ 8 \\ 6 \\ -2 \\ 2 \\ 1 \\ -4 \end{pmatrix}$$

Este problema es equivalente a encontrar valores para que las variables x_1, x_2, x_3, x_4, x_5 satisfagan las siguientes ocho restricciones de desigualdad:

$$\begin{aligned} x_2 - x_1 &\leq 1, \\ x_2 - x_4 &\leq -3, \\ x_3 - x_1 &\leq 8, \\ x_3 - x_2 &\leq 6, \\ x_4 - x_3 &\leq -2, \\ x_5 - x_2 &\leq 2, \\ x_5 - x_3 &\leq 1, \\ x_5 - x_4 &\leq -4. \end{aligned}$$

Una posible solución para este problema es $x = (-7, -6, 0, -2, -6)$, que se puede verificar comprobando que cumple cada una de las desigualdades. Otra posible solución es $x' = (0, 1, 7, 5, 1)$. Para obtener esta segunda solución no hace falta comprobar que se cumplen las desigualdades, ya que basta con darse cuenta de que cada componente de x' es 7 veces mayor que la correspondiente componente en x.

El siguiente lema verifica por qué se puede afirmar, sin necesidad de comprobaciones, que x' es otra posible solución.

Lema 4.4. *Sea $x = (x_1, x_2, \dots x_n)$ una solución para un sistema $Cx \leq b$ de restricciones de desigualdad, y sea $e \in \mathbb{R}$ una constante cualquiera. Entonces, el vector $x + e = (x_1 + e, x_2 + e, \dots, x_n + e)$ también es una solución para el sistema $Cx \leq b$.*

Demostración. Dadas dos variables x_i, x_j con $i, j \in \{1, \dots, n\}, i \neq j$ y una constante b_k con $k \in \{1, \dots, m\}$ cualesquiera tales que $x_i - x_j \leq b_k$ sea una restricción del sistema, se tiene que $(x_i + e) - (x_j + e) = x_i - x_j \leq b_k$ para cualquier constante $e \in \mathbb{R}$. Luego, si x cumple que $Cx \leq b$, entonces $x + e$ también lo va a cumplir. □

Es importante remarcar que, si en un sistema de restricciones de desigualdad $Cx \leq b$ se cumple que $b_k \geq 0$ para todo $k \in \{1, 2, \dots, m\}$, entonces encontrar una solución para dicho sistema es trivial: basta con tomar un mismo valor para todas las variables x_i, con $i \in \{1, 2, \dots, n\}$. Con esta asignación, todas las restricciones de dicho sistema se satisfacen ya que cumplen que $x_i - x_j = 0 \leq b_k$, donde $1 \leq i, j \leq n$. Por este motivo, el problema de encontrar una posible solución para el sistema de restricciones de desigualdad $Cx \leq b$ sólo es interesante cuando existe al menos un $k \in \{1, 2, \dots, m\}$ tal que $b_k < 0$.

En resumen, en este capítulo se consideran problemas de programación lineal de la forma

$$(PL)_{DC} \begin{cases} \text{Minimizar } f(x) = d^T x \\ \\ \text{sujeto a } x \in K \end{cases}$$

donde $K = \{x \in \mathbb{R}^n : Cx \leq b\}$ con $C \in \mathbb{R}^{m \times n}$ y $b \in \mathbb{R}^m$. Es decir, la entrada para el problema es una matriz C de tamaño $m \times n$ formada por 0, 1 y -1, un vector b de tamaño m y otro vector d de tamaño n.

En muchas ocasiones, la función objetivo no importa, es decir, basta con encontrar cualquier posible solución para el sistema de restricciones, esto es, cualquier vector $x \in \mathbb{R}^n$ que cumpla $Cx \leq b$, o determinar que no existe ninguna posible solución.

Observación 4.5. *El conjunto de restricciones K que se ha definido solo considera restricciones de desigualdad. No obstante, cabe destacar que, si se tiene una restricción de la forma $x_i = x_j + b_k$, esta se puede escribir como dos desigualdades: $x_i - x_j \leq b_k$ y $x_j - x_i \leq -b_k$. Además, se cumplirá necesariamente que $b_k < 0$ $0 - b_k < 0$.*

4.2. Grafos de restricciones

Los sistemas de restricciones de desigualdad se pueden interpretar desde un punto de vista de la teoría de grafos. Dado un sistema de restricciones de desigualdad, la matriz C de tamaño $m \times n$ del problema de programación lineal se puede entender como la traspuesta de una matriz de incidencia para un grafo con m aristas y n vértices.

Definición 4.6. *Dado un sistema de restricciones de desigualdad $Cx \leq b$, el **grafo de restricciones** correspondiente es un grafo $G = (V, A)$, donde $V = \{v_0, v_1, \ldots, v_n\}$ y $A = \{(v_i, v_j) : x_j - x_i \leq b_k, 1 \leq i, j \leq n, 1 \leq k \leq m\} \cup \{(v_0, v_i) : 1 \leq i \leq n\}$.*

El grafo de restricciones incluye un vértice adicional v_0 para garantizar que existe un vértice en el grafo que alcance a todos los demás vértices del grafo. Este vértice adicional se pone incluso cuando ya existe un vértice que cumpla esta condición en el grafo. Por ende, el conjunto de vértices V consiste en un vértice v_i para cada variable x_i, junto con un vértice adicional v_0. Por otro lado, el conjunto de aristas A contiene una arista por cada restricción de desigualdad, junto con una arista (v_0, v_i) para cada incógnita x_i. Si $x_j - x_i \leq b_k$ es una restricción de desigualdad, entonces el peso de la arista $(v_i, v_j) \in A$ es $w(v_i, v_j) = b_k$.

Observación 4.7. *En un grafo de restricciones G, ningún camino más corto cuyo origen es el vértice adicional v_0 puede tener peso positivo.*

Demostración. Para cada vértice $v \in V(G) \setminus \{v_0\}$ existe una arista (v_0, v) cuyo peso cumple que $w(v_0, v) = 0$, por cómo se ha definido este grafo. Luego existe un camino desde el vértice adicional v_0 hasta cualquier otro vértice del grafo con peso cero y, como $\delta(v_0, v)$ es el menor de los pesos de los distintos caminos, no podrá ser mayor que el peso de este camino formado por una arista con peso cero, es decir, $\delta(v_0, v) \leq 0$. □

En la siguiente figura se puede observar el grafo de restricciones correspondiente al sistema de restricciones de desigualdad expuesto como ejemplo en la sección anterior.

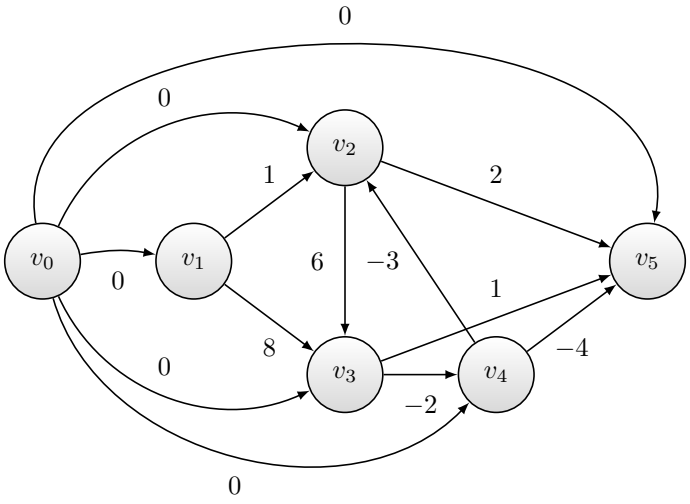


Figura 4.1: Grafo de restricciones correspondiente al sistema de restricciones de desigualdad de la Sección 4.1.

El siguiente teorema indica que cualquier sistema de restricciones de desigualdad se puede resolver, o se puede detectar que no tiene solución en caso de no tenerla, encontrando los pesos de los caminos más cortos en el grafo de restricciones correspondiente.

Teorema 4.8. *Dado un sistema $Cx \leq b$ de restricciones de desigualdad y sea $G = (V, A)$ el grafo de restricciones correspondiente. Si G no contiene ciclos de peso negativo, entonces $x = (\delta(v_0, v_1), \delta(v_0, v_2), \ldots \delta(v_0, v_n))$ es una posible solución para el sistema. Si G contiene un ciclo de peso negativo, entonces no existe ninguna solución para el sistema.*

Demostración. Primero, se prueba el teorema para el caso en el que G no contiene ciclos de peso negativo. Se considera una arista cualquiera $(v_i, v_j) \in A(G)$. Por la desigualdad triangular (Lema 2.1), y como todo vértice de G es alcanzable desde s en un grafo de restricciones, se tiene que $\delta(v_0, v_j) \leq \delta(v_0, v_i) + w(v_i, v_j)$ y, reordenando, $\delta(v_0, v_j) - \delta(v_0, v_i) \leq w(v_i, v_j)$. De este modo, si se define $x_i = \delta(v_0, v_i)$ y $x_j = \delta(v_0, v_j)$, se satisface la restricción de desigualdad $x_j - x_i \leq w(v_i, v_j)$, correspondiendo a la arista (v_i, v_j). Como esta arista es una cualquiera, si se define $x = (\delta(v_0, v_1), \delta(v_0, v_2), \ldots \delta(v_0, v_n))$, se satisfacen todas las restricciones del sistema, esto es, se cumple que $Cx \leq b$, concluyendo que x es una posible solución para el sistema.

Ahora, se estudia qué ocurre cuando G contiene un ciclo de peso negativo. Se considera, sin pérdida de generalidad, el ciclo de peso negativo $c = \langle v_1, \ldots, v_k \rangle$ con $v_1 = v_k$ (este ciclo no puede incluir el vértice v_0 ya que ninguna arista incide en él). Este ciclo se corresponde con las siguientes restricciones de desigualdad:

$$\begin{aligned} x_2 - x_1 &\leq w(v_1, v_2), \\ x_3 - x_2 &\leq w(v_2, v_3), \\ &\vdots \\ x_{k-1} - x_{k-2} &\leq w(v_{k-2}, v_{k-1}), \\ x_k - x_{k-1} &\leq w(v_{k-1}, v_k), \end{aligned}$$

Por reducción al absurdo, se supone que existe un vector $x \in \mathbb{R}^k$ que satisfaga cada una de estas k desigualdades. Este vector x también deberá satisfacer la desigualdad resultante

de sumar las k desigualdades, es decir, debe satisfacer

$$\sum_{i=2}^{k}(x_i - x_{i-1}) \leq \sum_{i=2}^{k} w(v_{i-1}, v_i).$$

En el sumatorio del lado izquierdo de la desigualdad cada variable x_i con $i \in \{2, \ldots, k-1\}$ se suma y se resta una vez, es decir,

$$\sum_{i=2}^{k}(x_i - x_{i-1}) = (x_2 - x_1) + (x_3 - x_2) + \ldots + (x_{k-1} - x_{k-2}) + (x_k - x_{k-1}) = x_k - x_1$$

Además, como en este ciclo $v_1 = v_k$, se tiene que $x_1 = x_k$ y se puede concluir que

$$x_k - x_1 = 0 \leq \sum_{i=2}^{k} w(v_{i-1}, v_i) \leq w(v_1, v_2) + w(v_2, v_3) + \ldots + w(v_{k-1}, v_k) = w(c).$$

Es decir, se obtendría que $0 \leq w(c)$, lo cual contradice la hipótesis de que c sea un ciclo de peso negativo ($w(c) < 0$). Por ende, como no existe ningún vector $x \in \mathbb{R}^k$ satisfaciendo cada una de estas k desigualdades, se concluye que no existe ninguna solución para el sistema. □

4.3. Una herramienta para resolver problemas de programación lineal

Una vez conocidos todos los resultados anteriores, en esta sección se concluye que el algoritmo de Bellman-Ford es un método para resolver un sistema de restricciones de desigualdad $Cx \leq b$ cualquiera.

Sea G el grafo de restricciones correspondiente a dicho sistema, se dan dos casos según cómo sean las aristas de dicho grafo:

Caso 1: Si G contiene uno o varios ciclos de peso negativo alcanzables desde el vértice adicional v_0, el algoritmo BELLMAN-FORD devuelve FALSE y, por el Teorema 4.8, no existe ninguna solución para el sistema $Cx \leq b$.

Caso 2: Si G no contiene ningún ciclo de peso negativo alcanzable desde el vértice adicional v_0, entonces al ejecutar BELLMAN-FORD sobre G se van obteniendo las estimaciones $v.d$ cuyo valor converge, a lo largo de las iteraciones del algoritmo, hasta obtener $v.d = \delta(s, v)$ para cada $v \in V(G)$. De este modo, y como por el Teorema 4.8 se sabe que $x = (\delta(v_0, v_1), \delta(v_0, v_2), \ldots, \delta(v_0, v_n))$ es una de las soluciones para el sistema $Cx \leq b$, se puede afirmar que el algoritmo de Bellman-Ford resuelve dicho sistema.

Al aplicar el algoritmo de Bellman-Ford sobre el grafo de restricciones de la Figura 4.1 se obtienen los siguientes pesos para los caminos más cortos a cada vértice del grafo desde el vértice v_0:

$$\begin{aligned}
\delta(v_0, v_1) &= 0, \\
\delta(v_0, v_2) &= -5, \\
\delta(v_0, v_3) &= 0, \\
\delta(v_0, v_4) &= -2, \\
\delta(v_0, v_5) &= -6.
\end{aligned}$$

Luego, por todo lo probado, se puede afirmar que el vector $x = (0, -5, 0, -2, -6)$ compuesto por estos valores es una solución para el sistema de restricciones de desigualdad presentado en la Sección 4.1.

A continuación, se presentan dos lemas en los que se prueba que el algoritmo de Bellman-Ford también es capaz de maximizar y minimizar, respectivamente, las funciones objetivo de cada uno de los problemas de programación lineal enunciados en estos lemas.

Lema 4.9. *Sea $Cx \leq b$ un sistema de m restricciones de desigualdad con n incógnitas. Entonces, el algoritmo* Bellman-Ford*, al ser ejecutado sobre el grafo de restricciones correspondiente a dicho sistema, resuelve el siguiente problema de programación lineal*

$$(P) \begin{cases} \textit{Maximizar } \sum_{i=1}^{n} x_i \\ \\ \textit{sujeto a } x \in K \end{cases}$$

donde el conjunto de puntos admisibles es $K = \{x \in \mathbb{R}^n : Cx \leq b, x \leq 0\}$.

Demostración. Sea G el grafo de restricciones correspondiente al sistema $Cx \leq b$ dado. Se supone que este grafo no contiene ciclos de peso negativo alcanzables desde el vértice adicional v_0, ya que entonces el sistema no tendría solución. Por el Teorema 4.8 se sabe que una solución del sistema de restricciones de desigualdad es $x_i = \delta(v_0, v_i)$ para todo $i \in \{1, 2, \ldots, n\}$. Además, por la Observación 4.7 se sabe $\delta(v_0, v_i) \leq 0$ para todo $i \in \{1, 2, \ldots, n\}$. Luego, se puede afirmar que $x_i = \delta(v_0, v_i) \leq w(v_0, v_i) = 0$ para todo $i \in \{1, 2, \ldots, n\}$.

A continuación, se demuestra, por reducción al absurdo, que la solución proporcionada por el algoritmo de Bellman-Ford maximiza la función objetivo $\sum_{i=1}^{n} x_i$. Supóngase que exista otra solución $y = (y_1, \ldots, y_n)$, distinta a la que proporciona el algoritmo de Bellman-Ford, que satisface las restricciones del problema y que maximiza la función objetivo. Se fija una variable x_i, con $i \in \{1, 2, \ldots, n\}$, del vector solución que proporciona Bellman-Ford y se recuerda al lector que esta variable cumple que $x_i = \delta(v_0, v_i)$, que es la distancia más corta desde v_0 hasta el vértice v_i. Supóngase también que esta distancia corresponde a la del camino más corto $\langle v_0, v_{i_1}, \ldots, v_{i_k} \rangle$, donde $v_{i_k} = v_i$, el cual está formado por aristas del grafo de restricciones G. Como cada arista de este camino se corresponde con una restricción del sistema $Cy \leq b$, se tienen las siguientes desigualdades:

$$\begin{aligned} y_{i_2} - y_{i_1} &\leq w(v_{i_1}, v_{i_2}), \\ y_{i_3} - y_{i_2} &\leq w(v_{i_2}, v_{i_3}), \\ &\vdots \\ y_{i_{k-1}} - y_{i_{k-2}} &\leq w(v_{i_{k-2}}, v_{i_{k-1}}), \\ y_{i_k} - y_{i_{k-1}} &\leq w(v_{i_{k-1}}, v_{i_k}). \end{aligned}$$

Sumando estas desigualdades, al igual que en la demostración del Teorema 4.8, se cancelan todos aquellos términos que no son y_{i_k} e y_{i_1}, esto es:

$$\sum_{j=2}^{k} (y_{i_j} - y_{i_{j-1}}) = y_{i_k} - y_{i_1} = y_i - y_{i_1} \leq \sum_{j=2}^{k} w(v_{i_{j-1}}, v_{i_j}),$$

El sumatorio de la derecha corresponde al peso del camino desde v_{i_1} hasta $v_{i_k} = v_i$. Sin embargo, el camino completo que se está considerando comienza en v_0. Como la primera

arista de este camino es (v_0, v_{i_1}) con peso $w(v_0, v_{i_1}) = 0$, se tiene que el peso total del camino $\langle v_0, v_{i_1}, \ldots, v_{i_k} = v_i \rangle$ es

$$w(v_0, v_1) + \sum_{j=2}^{k} w(v_{i_{j-1}}, v_{i_j}) = 0 + \sum_{j=2}^{k} w(v_{i_{j-1}}, v_{i_j}) = \delta(v_0, v_i)$$

donde la última igualdad se tiene porque se ha tomado la hipótesis de que este camino es el más corto desde v_0 hasta v_i. Sustituyendo en la desigualdad anterior:

$$y_i - y_{i_1} \leq \delta(v_0, v_i).$$

Reorganizando esta desigualdad y usando que $y_{i_1} \leq 0$ se tiene que

$$y_i \leq y_{i_1} + \delta(v_0, v_i) \leq \delta(v_0, v_i) = x_i.$$

Luego, sumando todas las $i \in \{1, 2, \ldots, n\}$ se obtiene que

$$\sum_{i=1}^{n} y_i \leq \sum_{i=1}^{n} x_i,$$

contradiciendo que exista otra solución, distinta a la que proporciona el algoritmo de Bellman-Ford, que cumpla las restricciones enunciadas y maximice este sumatorio. □

Lema 4.10. *Sea $Cx \leq b$ un sistema de m restricciones de desigualdad con n incógnitas. Entonces, el algoritmo* Bellman-Ford*, al ser ejecutado sobre el grafo de restricciones correspondiente a dicho sistema, resuelve el siguiente problema de programación lineal*

$$(P) \begin{cases} \textit{Minimizar } (\text{máx}\{x_i\} - \text{mín}\{x_i\}) \\ \\ \textit{sujeto a } Cx \leq b \end{cases}$$

donde $i \in \{1, 2, \ldots, n\}$.

Demostración. Por la Observación 4.7 y el Teorema 4.8 se tiene una cota para el máximo, esto es, $\text{máx}\{x_i\} = \text{máx}\{\delta(v_0, v_i)\} \leq 0$. Además, se puede afirmar que no puede ocurrir que para todos los vértices exista un camino más corto con peso negativo. Suponiendo, por reducción al absurdo, que esto fuese así, entonces el predecesor de cada vértice iría a algún vértice distinto del vértice origen. Luego, si se intenta construir un camino más corto para cualquier vértice, nunca llegará al vértice adicional, contradiciendo que es un camino más corto desde el vértice origen hasta dicho vértice.

Falta probar que, al ejecutar el algoritmo de Bellman-Ford sobre el grafo de restricciones correspondiente al sistema $Cx \leq b$, se minimiza el valor de $-\text{mín}\{x_i\}$. Por la Observación 4.1 se tiene que esto es equivalente a maximizar el valor de $\text{mín}\{x_i\}$. Esto se cumple ya que el peso de los caminos más cortos en el grafo de restricciones, que proporciona el algoritmo de Bellman-Ford, es el valor más pequeño para que se satisfagan todas las restricciones; luego, si se aumentase cualquiera de estos valores, se incumplirían algunas de las restricciones. □

En la actualidad, el algoritmo de Bellman-Ford, gracias a su capacidad para manejar grafos con pesos negativos y detectar ciclos de coste negativo, se usa para resolver problemas de diversos campos que preocupan a los seres humanos. En el artículo "A Bellman-Ford Approach to Energy Efficient Routing of Electric Vehicles" [1], se emplea para optimizar rutas energéticamente eficientes en vehículos eléctricos. Por su parte, el artículo

"Comparative analysis of Bellman-Ford and Dijkstra's algorithms for optimal evacuation route planning in multi-floor buildings" [4] lo aplica en la planificación de evacuaciones en edificios de varias plantas. Asimismo, en el artículo "Research on Distribution Path Optimization of Computer-aided Logistics Warehousing Management System" [5], se utiliza para optimizar rutas de distribución en sistemas logísticos. Estos ejemplos muestran cómo el algoritmo es relevante y efectivo para abordar desafíos modernos en movilidad, seguridad y logística.

Referencias

[1] Abousleiman, R. y O. Rawashdeh. A Bellman-Ford Approach to Energy Efficient Routing of Electric Vehicles, *2015 IEEE Transportation Electrification Conference and Expo (ITEC)*, Dearborn, MI, USA, 2015, pp. 1-4. doi:10.1109/itec.2015.7165772

[2] AcademiaLab. Algoritmo de Bellman-Ford, *Enciclopedia*, 2025. Disponible en: `https://academia-lab.com/enciclopedia/algoritmo-de-bellman-ford/`

[3] Arias Figueroa, J. G. Camino más corto: Algoritmo de Bellman- Ford, *Algorithms and More*, 2013. Disponible en: `https://jariasf.wordpress.com/2013/01/01/camino-mas-corto-algoritmo-de-bellman-ford/`

[4] Bhat, R., P. K. Rao, C. R. Kamath, V. Tandon y P. Vizzapu. Comparative analysis of Bellman-Ford and Dijkstra's algorithms for optimal evacuation route planning in multi-floor buildings, *Cogent Engineering*, 11(1). doi: 10.1080/23311916.2024.2319394

[5] Chang, W. y J. Wang. Research on Distribution Path Optimization of Computer-Aided Logistics Warehousing Management System, *2025 IEEE International Conference on Electronics, Energy Systems and Power Engineering (EESPE)*, Shenyang, 2025, pp.1445-1449. doi: 10.1109/EESPE63401.2025.10986819

[6] Cormen, T. H., C. E. Leiserson, R. L. Rivest y C. Stein. *Introduction to Algorithms*, 4th edition, The MIT Press, Cambridge, Massachusetts, 2022.

[7] Fioravanti, M. A. *Matemática Discreta: Teoría de Grafos*, Apuntes de la asignatura, Curso 2021-22, Facultad de Ciencias, Universidad de Cantabria, 2021. Disponible en: `https://personales.unican.es/fioravam/MatDiscreta/Mat-Discreta_3ra-parte_nov2021.pdf`

[8] Google. Ruta en coche desde Santander hasta Logroño, *Google Maps*, 2025. Disponible en: `https://www.google.com/maps/dir/Estaci%C3%B3n+Autobuses+de+Santander,+Plaza+Estaciones,+Santander/Nueva+Estaci%C3%B3n+de+Autobuses+de+Logro%C3%B1o,+Avenida+de+Col%C3%B3n,+Logro%C3%B1o/@42.9429531,-3.8013569,9z/data=!3m1!4b1!4m13!4m12!1m5!1m1!1s0xd494bcb58883bcb:0xf02a497984df339!2m2!1d-3.8096996!2d43.4593554!1m5!1m1!1s0xd5aab07a24c43f7:0x3bb21212e2fcac39!2m2!1d-2.44198!2d42.4575269?entry=ttu&g_ep=EgoyMDI1MDYxMS4wIKXMDSoASAFQAw%3D%3D`

[9] Lipschutz, S. y M. L. Lipson. Grafos dirigidos, *Matemáticas Discretas*, 3rd edition, McGraw-Hill, Madrid, 2009.

[10] Paredes Campos, J. *Modelos de Optimización en Redes*, Universidad San Pedro, Chimbote, 2015, pp. 1-67. Disponible en: `https://1library.co/document/y9r2o0wy-modelosoptimizacionredes.html`

[11] Pola, M. C. *Optimización*, Apuntes de la asignatura, Curso 2023-24, Facultad de Ciencias, Universidad de Cantabria, 2024. Documento sin publicar.

[12] Schrijver, A. On the History of the Shortest Path Problem, *Optimization Stories*, European Mathematical Society Press, Berlín, 2012, pp.155-167. doi: 10.4171/DMS/6/19

[13] Sedgewick, R. y K. Wayne. *Algorithms*, 4th edition, Addison Wesley, Boston, 2021.

[14] Shen, Z. Single-Source Shortest Paths, *CS3221.01 Algorithm Analysis*, Course notes, Spring 2024, Plymouth State University, 2024. Disponible en: `https://turing.plymouth.edu/~zshen/Webfiles/notes/CS322/note24.pdf`